Ο ΚΟΣΜΟΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

.. ΜΙΑ ΟΛΟΚΛΗΡΩΜΕΝΗ ΘΕΩΡΗΣΗ ΤΗΣ ΕΠΙΣΤΗΜΗΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ..

ΤΕΙ ΜΕΣΟΛΟΓΓΙΟΥ

Σχολή:Διοίκησης και Οικονομίας

Τμήμα : Εφαρμογών Πληροφορικής στην Διοίκηση και στην Οικονομία

ΣΠΟΥΔΑΣΤΡΙΑ: ΚΟΝΤΟΥΛΑ ΣΟΦΙΑ

ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: 10344

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΔΡΟΣΟΣ ΛΑΜΠΡΟΣ

ΠΕΡΙΕΧΟΜΕΝΑ

Ενότητα Α: Από τον πρώτο Η/Υ στον σύγχρονο προσωπικό υπολογιστή

Κεφάλαιο 1: Ιστορική αναδρομή των υπολογιστικών συστημάτων

1.1 Μηχανές πρώτης γενεάς (1951-1958)

Από τον Ιούνιο του 1951 και μέχρι το 1958 κατασκευάζονται οι υπολογιστικές μηχανές της Πρώτης Γενεάς, που χαρακτηρίζονται από τη χρησιμοποίηση λυχνιών κενού στην κατασκευή των κυκλωμάτων τους. Πρώτη υπολογιστική μηχανή της κατηγορίας αυτής ήταν η UNIVAC I, (UNIVersal Automatic Computer), που κατασκευάσθηκε από την ίδια ομάδα ατόμων που συμμετείχαν στην κατασκευή της μηχανής ENIAC.

Οι μηχανές της πρώτης γενεάς χρησιμοποιούν στην κεντρική τους μνήμη μαγνητικά τύμπανα. Στην περιφερειακή, η δευτερεύουσα μνήμη χρησιμοποιούν επίσης μαγνητικά τύμπανα, ή μαγνητικές ταινίες. Στη θέση της μονάδας εισόδου έχουν συσκευές ανάγνωσης διατηρημένων καρτών, ή χαρτοταινίες και στη θέση της μονάδας εξόδου συσκευές διάτρησης καρτών ή εκτυπωτικές συσκευές. Η μηχανή UNIVAC I ήταν η πρώτη ηλεκτρονική υπολογιστική μηχανή, που είχε "μαζική" παραγωγή και πώληση. Πωλούνταν από την εταιρεία Remington Rand και η πρώτη πώληση έγινε για λογαριασμό του Γραφείου Απογραφών των Η.Π.Α. Χαρακτηριστικό είναι ότι την εποχή εκείνη, για το ευρύ κοινό, η λέξη "univac" υπονοούσε "υπολογιστική μηχανή".

Η εταιρία International Business Machines (IBM) ήταν, κατά την περίοδο αυτή, μια μικρή σχετικά εταιρία, που δεν είχε ενδιαφέροντα στην κατασκευή υπολογιστικών μηχανών. Από το 1953 όμως, ασχολήθηκε με την κατασκευή υπολογιστικών μηχανών και άρχισε με τη μηχανή IBM-701, που ήταν μια απλή προγραμματιζόμενη μηχανή.

Ακολούθησαν και άλλες μηχανές της σειράς αυτής, όπως η IBM-305 RAMAC και η πολύ επιτυχημένη IBM-650, που κυριάρχησε για μια ολόκληρη πενταετία. Η τελευταία μηχανή της οικογένειας αυτής ήταν η IBM-709, που κατασκευάσθηκε το 1958. Όλες οι μηχανές της σειράς IBM κατασκευάζονται με λυχνίες κενού, που είναι άλλωστε το χαρακτηριστικό της γενεάς αυτής. Η μηχανή IBM-709 πάντως, είναι η τελευταία, που κατασκευάζει η IBM, με λυχνίες κενού. Η μηχανή IBM-305 RAMAC, είναι η πρώτη μηχανή εφοδιασμένη με σύστημα δίσκου.

1.2. Μηχανές δεύτερης γενεάς (1959-1964)

Το έτος 1948 πραγματοποιείται, στα εργαστήρια της εταιρείας Bell, η ανακάλυψη της τριόδου ηλεκτρονικής λυχνίας, γνωστής ως τρανζίστορ

(transistor). Ως γνωστό, οι κατασκευαστές του τρανζίστορ ήταν οι John Bardeen, Walter Brattain και William Shockley, στους οποίους το 1956 απενεμήθη, για το λόγο αυτό, το βραβείο Nobel. Το τρανζίστορ είναι περισσότερο αξιόπιστο, κατά πολύ πιο φθηνό, έχει πολύ χαμηλές απαιτήσεις τροφοδοσίας, παράγει χαμηλή θερμότητα κατά την λειτουργία του και είναι κατά πολύ μικρότερο από τις πιο μικρές λυχνίες κενού.

Μετά την ανακάλυψη του τρανζίστορ αρχίζει η ουσιαστική παραγωγή των υπολογιστικών μηχανών τις Δευτέρας Γενεάς (1959-1964). Οι μηχανές αυτές έχουν ως κύριο χαρακτηριστικό τη χρησιμοποίηση του τρανζίστορ στα κυκλώματα της κεντρικής μνήμης, αντί των μαγνητικών δακτυλίων. Οι υπόλοιπες μονάδες τους έχουν την ίδια τεχνολογία με αυτή της πρώτης γενεάς.

Η πρώτη ηλεκτρονική υπολογιστική μηχανή με τρανζίστορ κατασκευάσθηκε στο MIT και ονομάσθηκε TX-O (Transistorized experiment computer O).Σκοπός της κατασκευής της ήταν ο έλεγχος της μηχανής TX-2, που προοριζόταν για εκμετάλλευση. Τελικά η μηχανή TX-2 ουδέποτε ολοκληρώθηκε. Το 1957, ένας από τους μηχανικούς που συνεργάσθηκαν στην κατασκευή της μηχανής TX-2, ιδρύει την εταιρεία Digital Equipment Corporation (DEC). Η εταιρεία DEC κατασκευάζει αρχικά μια μηχανή παρόμοια με την TX-0, ενώ το έτος 1961 κατασκευάζει την πολύ αξιόλογη μηχανή PDP-1, πρώτη στη σειρά των γνωστών PDP μηχανών. Η μηχανή PDP-1 κόστιζε την εποχή εκείνη 120,000 δολάρια.

Η μηχανή PDP-1 ήταν ο πρώτος υπολογιστής κατηγορίας mini, που κατασκευάσθηκε και επιπλέον είχε σημαντικές πωλήσεις. Μετά την κατασκευή της μηχανής PDP-1, ένα πλήθος καινοτομιών άρχισαν να εμφανίζονται στο χώρο των υπολογιστικών μηχανών. Ανάμεσα στις καινοτομίες αυτές, είναι και η δημιουργία οθονών σε μορφή παρόμοια με αυτή των σημερινών.

Η IBM κατασκεύασε, εν τω μεταξύ, τη μηχανή IBM-7094, που διαδέχτηκε την IBM-709 και περιείχε τρανζίστορ αντί λυχνιών κενού. Η μηχανή IBM-7094 ήταν η ταχύτερη μηχανή, κατά την εποχή εκείνη και το κόστος της μερικά εκατομμύρια δολάρια. Από τη στιγμή που οι μηχανές

άρχισαν να πωλούνται και η κατασκευή τους να έχει εμπορικό ενδιαφέρον, τόσο οι εταιρείες DEC και IBM, όσο και άλλες εταιρείες όπως οι Control Data Corporation και η Univac αλλά και πολλές ακόμη, άρχισαν να κατασκευάζουν και να πωλούν ηλεκτρονικές υπολογιστικές μηχανές.

1.3. Μηχανές τρίτης γενεάς (1965-1970)

Η Τρίτη Γενεά (1965-1970) χαρακτηρίζεται από τη χρησιμοποίηση ολοκληρωμένων κυκλωμάτων μεγάλης ολοκλήρωσης (LSI-Large Scale Integration) στα κυκλώματα των μηχανών. Χαρακτηρίζεται επίσης από την κατασκευή κεντρικών μνημών με μαγνητικούς δακτυλίους. Τα ολοκληρωμένα κυκλώματα επινοήθηκαν από τον Jack Kilby το έτος 1958, στα εργαστήρια της εταιρείας Texas Instruments. Ένα σύνηθες ολοκληρωμένο κύκλωμα αποτελείται από ένα μικρό ορθογώνιο παραλληλεπίπεδο, με διάφορες διαστάσεις. Είναι συνηθισμένη αυτή με 25,4 χιλιοστά μήκος, 0,5 χιλιοστά πλάτος και 0,125 χιλιοστά πάχος. Στο εσωτερικό του περιέχεται ένα μονολιθικό κύκλωμα πυριτίου (τσιπ) με πολλά αλληλοσυνδεδεμένα τρανζίστορ και άλλα στοιχεία. Τα πρώτα ολοκληρωμένα κυκλώματα (Integrated Circuits-IC) περιείχαν περιορισμένο πλήθος στοιχείων, ενώ τα μεταγενέστερα διαρκώς αυξανόμενο πλήθος στοιχείων.

Η τεχνολογία των μαγνητικών δίσκων έχει αναπτυχθεί και αντικαθιστά τις μονάδες μαγνητικής ταινίας, οι οποίες χρησιμοποιούνται πλέον μόνο για διαδικασίες back up. Οι τερματικές συσκευές με πληκτρολόγιο και οθόνη αντικαθιστούν σταδιακά τις μονάδες διάτρησης καρτών και χρησιμοποιούνται ως μηχανισμοί εισόδου και εξόδου αντίστοιχα.

Η τρίτη γενιά είναι δυνατό να υποστηριχθεί ότι χαρακτηρίζεται από την κυριαρχία των μηχανών της IBM και ιδιαίτερα των μηχανών τύπου IBM System/360. Η οικογένεια των μηχανών System/360 περιλάμβανε μια μικρή μηχανή, τη System/360-30, με κύρια χρήση τις εμπορικές εφαρμογές, τη μηχανή System/360-65, ένα ισχυρότερο τύπο μηχανής, καθώς και άλλες μηχανές για εμπορικές και επιστημονικές εφαρμογές. Στις τελευταίες

Γενικά πάντως, η διάκριση μεταξύ των μηχανών τρίτης και τέταρτης γενεάς δεν είναι τόσο εμφανής, όσο μεταξύ των προηγούμενων γενεών.

1.5. Μηχανές πέμπτης γενεάς

Οι μηχανές της Πέμπτης Γενεάς δεν έχουν ακόμη καθορισμένες ιδιότητες, αλλά θεωρείται ότι θα διαφέρουν ουσιαστικά από το μοντέλο της μηχανής του John Von Neumann. Ήδη από τα μέσα της δεκαετίας του 1980 ξεκίνησε στην Ιαπωνία ένα μεγαλόπνοο σχέδιο σχεδίασης και κατασκευής υπολογιστικών μηχανών της αποκαλούμενης πέμπτης γενεάς, οι οποίες αναμένεται να διαφέρουν στη λογική από τις σημερινές μηχανές που ονομάζονται «συμβατικές».

Κεφάλαιο 2: Δομή του προσωπικού υπολογιστή

Με τον όρο δομή ενός αντικειμένου αναφερόμαστε στα μέρη που το αποτελούν καθώς και στον τρόπο με τον οποίο τα μέρη αυτά συνδέονται και συνεργάζονται, ώστε να αποτελέσουν ένα σύνολο. Η εξοικείωση με τη δομή του υπολογιστή είναι σημαντική, για να μπορέσει κανείς να κατανοήσει τη λειτουργία του.

Θα ξεκινήσουμε την περιγραφή της δομής του υπολογιστή χρησιμοποιώντας το παράδειγμα της επεξεργασίας ενός κειμένου για εκτύπωση όπου ακολουθείται η εξής διαδικασία:

- Αρχικά εισάγουμε το κείμενο στον υπολογιστή (χαρακτήρα προς χαρακτήρα).
- Το κείμενο εμφανίζεται στην οθόνη, ενώ ταυτόχρονα αποθηκεύεται προσωρινά στη μνήμη.
- Στη συνέχεια, μπορούμε να δώσουμε στον υπολογιστή εντολές σχετικά με τη μορφοποίηση του κειμένου (π.χ. μέγεθος γραμμάτων, αποστάσεις παραγράφων κτλ.).
- Ο υπολογιστής επεξεργάζεται τις εντολές αυτές και εμφανίζει το νέο (μορφοποιημένο) κείμενο στην οθόνη.
- Ακόμη, μπορούμε να εκτυπώσουμε το κείμενο σε χαρτί χρησιμοποιώντας τον εκτυπωτή.
- Τέλος, αποθηκεύουμε το κείμενο σε ένα αποθηκευτικό μέσο (δίσκο), ώστε να μπορούμε κάποια άλλη στιγμή να το ανακαλέσουμε, για να το ξανατυπώσουμε ή να το διορθώσουμε.

Από το παράδειγμα που αναφέραμε μπορούμε να συμπεράνουμε ότι για τη διαδικασία της επεξεργασίας δεδομένων απαιτούνται τα ακόλουθα τμήματα υλικού:

- Μονάδες με τις οποίες μπορούμε να εισάγουμε δεδομένα στον υπολογιστή (π.χ. πληκτρολόγιο) και αποκαλούνται ***μονάδες εισόδου (input units).***
- Μονάδες με τη βοήθεια των οποίων ο υπολογιστής εμφανίζει τα

αποτελέσματα της επεξεργασίας (π.χ. οθόνη, εκτυπωτής). Οι μονάδες αυτές αποκαλούνται ***μονάδες εξόδου (output units).***

- Μονάδες με τις οποίες ο υπολογιστής μπορεί να πραγματοποιήσει την επεξεργασία. Στην πλειοψηφία των υπολογιστών, χρησιμοποιείται μια τέτοια μονάδα, η οποία αποκαλείται ***Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ, Central Processing Unit, CPU).***
- Μονάδες στις οποίες ο υπολογιστής μπορεί να αποθηκεύσει δεδομένα κατά τη διάρκεια της λειτουργίας του και αποκαλούνται ***κύρια μνήμη (main memory).***
- Μονάδες στις οποίες ο υπολογιστής μπορεί να αποθηκεύσει δεδομένα και στις οποίες θα παραμείνουν και μετά τη λήξη της λειτουργίας του. Οι μονάδες αυτές αποκαλούνται ***μονάδες βοηθητικής μνήμης.*** Τρόπους με τους οποίους επικοινωνούν οι παραπάνω μονάδες μεταξύ τους.
- Η επικοινωνία μεταξύ των μονάδων γίνεται χρησιμοποιώντας τους ***διαδρόμους (buses).***

Πληροφορίες (δεδομένα και σήματα ελέγχου) μπορούν να μεταφερθούν:

- Από τη μονάδα εισόδου προς την ΚΜΕ και την κύρια μνήμη,
- από την ΚΜΕ προς την κύρια μνήμη, και το αντίστροφο,
- από την ΚΜΕ ή την κύρια μνήμη στη μονάδα εξόδου,
- από την ΚΜΕ ή την κύρια μνήμη στη μονάδα αποθήκευσης, και το αντίστροφο,
- από τη μονάδα αποθήκευσης προς την ΚΜΕ και την κύρια μνήμη, και το αντίστροφο.

Είναι πολύ σημαντικό να τονίσουμε ότι κάθε χρονική στιγμή μόνο δύο συσκευές μπορούν να επικοινωνούν μέσω του διαδρόμου. Έτσι, αν κάποια στιγμή επικοινωνεί μέσω του διαδρόμου η ΚΜΕ με τη μνήμη, η μονάδα εισόδου δεν μπορεί να στείλει δεδομένα, αλλά πρέπει να περιμένει να ολοκληρωθεί η επικοινωνία μεταξύ της ΚΜΕ και της κυρίας μνήμης.

Για παράδειγμα, στην περίπτωση του υπολογισμού του αποτελέσματος μιας αριθμητικής πράξης, ο χρήστης εισάγει τα δεδομένα (αριθμούς και σύμβολο πράξης) από το πληκτρολόγιο. Τα δεδομένα μεταφέρονται μέσω της ΚΜΕ στην κύρια μνήμη.

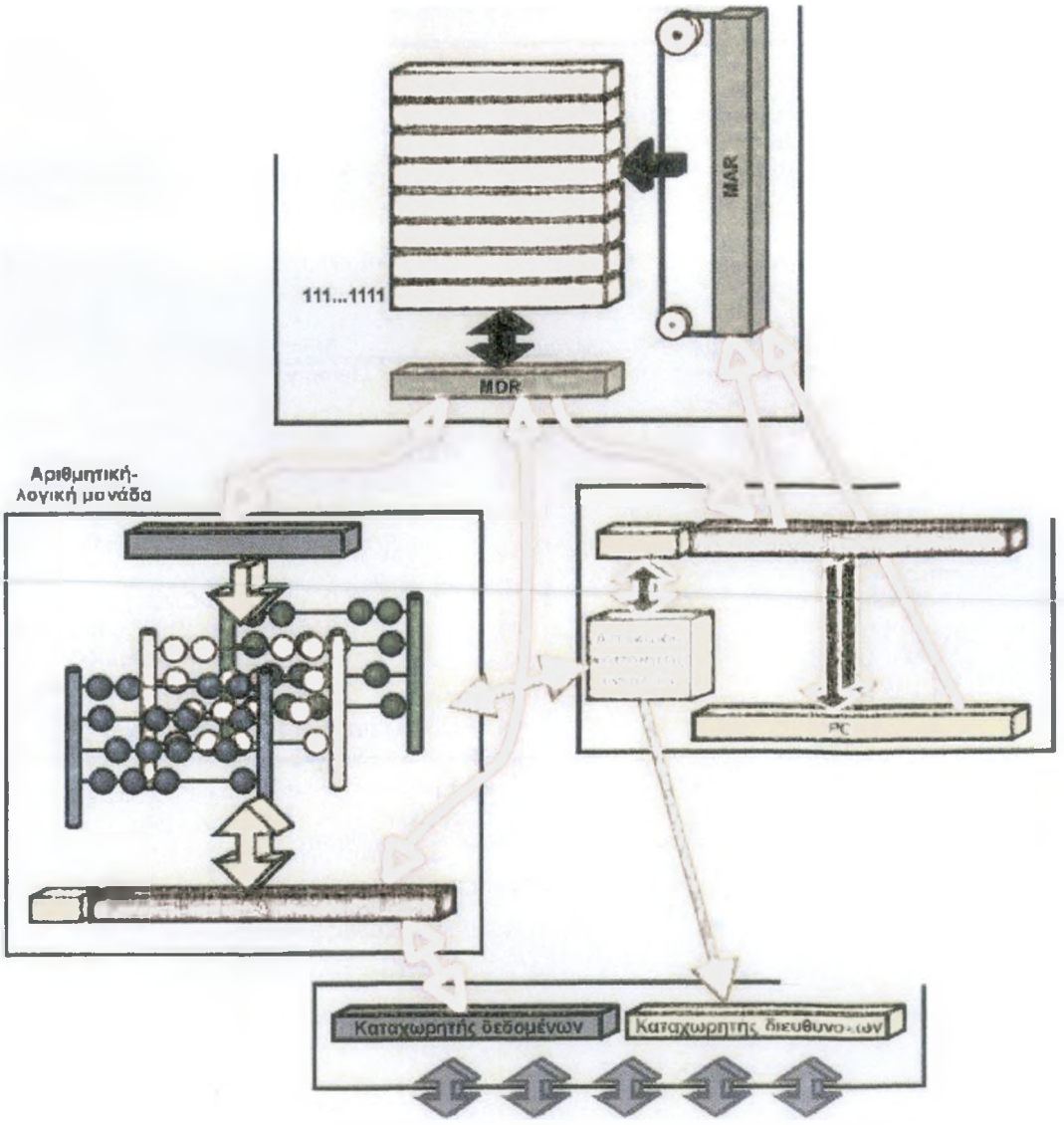
Στη συνέχεια, η ΚΜΕ ανακαλεί τα δεδομένα από την κύρια μνήμη, εκτελεί την αριθμητική πράξη και τοποθετεί το αποτέλεσμα στην κύρια μνήμη. Το αποτέλεσμα της πράξης εμφανίζεται στην οθόνη. Τέλος, είναι δυνατό το αποτέλεσμα της πράξης να αποθηκευτεί για μελλοντική χρήση στο σκληρό δίσκο του υπολογιστή.

2.1. Κεντρική Μονάδα Επεξεργασίας

Η Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ) επεξεργάζεται δεδομένα. Η επεξεργασία των δεδομένων γίνεται σε μια σειρά από βήματα, καθένα από τα οποία ονομάζεται εντολή. Οι εντολές που εκτελούνται από την ΚΜΕ είναι

εντολές σε γλώσσα μηχανής. Μια εντολή σε γλώσσα μηχανής είναι μια σειρά από δυαδικά ψηφία όπου είναι κωδικοποιημένο το είδος της εντολής. Οι εντολές της γλώσσας μηχανής είναι αποθηκευμένες στην κύρια μνήμη, από όπου τις ανακαλεί και τις εκτελεί η ΚΜΕ. Η ΚΜΕ αποτελείται από τρία τμήματα:

- Την αριθμητική και λογική μονάδα (Arithmetic and Logic Unit, ALU), η οποία εκτελεί τις αριθμητικές και λογικές πράξεις.
- Τη μονάδα ελέγχου (Control Unit), η οποία συντονίζει την εκτέλεση των εντολών και την εκτέλεση των πράξεων στην αριθμητική και λογική μονάδα.
- Τους καταχωρητές (registers), οι οποίοι χρησιμεύουν ως χώροι αποθήκευσης.
- Η διαδικασία εκτέλεσης μιας εντολής περιλαμβάνει δύο φάσεις:
- Τη φάση ανάκλησης (fetch cycle), κατά την οποία η μονάδα ελέγχου αποφασίζει ποια εντολή πρέπει να ανακληθεί από τη μνήμη και την ανακαλεί. Στη φάση αυτή, η μονάδα ελέγχου αποκωδικοποιεί την εντολή (βρίσκει, δηλαδή, ποια εντολή είναι).
- Τη φάση εκτέλεσης (execution cycle), κατά την οποία η ΚΜΕ εκτελεί την εντολή. Αν, για παράδειγμα, η εντολή είναι για την εκτέλεση αριθμητικής ή λογικής πράξης, η αριθμητική και λογική μονάδα εκτελεί την πράξη.



Οι εντολές που συναντάμε στους υπολογιστές είναι:

- Ανάγνωση ενός δεδομένου από τη μνήμη.
- Εκτέλεση πράξεων (π.χ. πρόσθεση, πολλαπλασιασμός κ.λ.π.)
- Εισαγωγή ενός δεδομένου από μονάδα εισόδου.
- Εξαγωγή ενός δεδομένου από μονάδα εξόδου.

Για να γίνει κατανοητός ο τρόπος που λειτουργεί ο υπολογιστής, ας υποθέσουμε ότι έχουμε την εκτέλεση μιας εντολής πολλαπλασιασμού δύο αριθμών (π.χ. 2x3) που βρίσκονται στη μνήμη και πως το αποτέλεσμα πρέπει να αποθηκευτεί στη μνήμη. Χρειάζονται οι ακόλουθες εντολές:

1 Ανάγνωση του πρώτου αριθμού και μεταφορά στον καταχωρητή Α.
2 Ανάγνωση του δεύτερου αριθμού και μεταφορά στον καταχωρητή Β.
3 Πολλαπλασιασμός του αριθμού που βρίσκεται στους καταχωρητές Α και Β και αποθήκευση του αποτελέσματος στον καταχωρητή Α.
4 Εγγραφή του αποτελέσματος στη μνήμη.

Η εκτέλεση κάθε εντολής περιλαμβάνει δυο φάσεις:

- Τη *φάση ανάκλησης,* όπου ανακαλείται από τη μνήμη το είδος της εντολής, το οποίο αποθηκεύεται στη μονάδα ελέγχου (CU).
- Τη *φάση εκτέλεσης* όπου η μονάδα ελέγχου δίνει εντολή στην αριθμητική και λογική μονάδα η οποία εκτελεί τον πολλαπλασιασμό μεταξύ των περιεχομένων των καταχωρητών Α και Β και αποθηκεύει το αποτέλεσμα στον καταχωρητή Α όπως φαίνεται από τα βέλη κατεύθυνσης του σχήματος.

2.2. Κεντρική Μνήμη

Η κεντρική μνήμη (main memory) που αναφέρεται επίσης και ως πρωτεύουσα μνήμη (primary memory), δίνει σε ένα υπολογιστικό σύστημα τη δυνατότητα καταχώρησης και ανάκλησης:
α. Των υπό εκτέλεση προγραμμάτων, που καταχωρούνται ως ακολουθίες πράξεων.
β. Των πληροφοριακών δεδομένων επί των οποίων τα εκτελούμενα προγράμματα εφαρμόζονται.

Πιο απλά, η κεντρική μνήμη των υπολογιστικών συστημάτων θεωρούμε ότι χρησιμοποιείται για τους ακόλουθους τέσσερις βασικούς λόγους:

1. Για να καταχωρούνται τα πληροφοριακά δεδομένα, τα οποία εισάγονται στο υπολογιστικό σύστημα για άμεση επεξεργασία.
2. Για να καταχωρούνται τα αποτελέσματα πράξεων που εκτελούνται επί των πληροφοριακών δεδομένων, που ήδη είναι καταχωρημένα στη μνήμη.
3. Για να καταχωρούνται αντίγραφα των πληροφοριακών δεδομένων που ήδη υπάρχουν στην μνήμη, αλλά απαιτείται η αναπαραγωγή τους.
4. Για να καταχωρούνται οι ακολουθίες των πράξεων, που πρόκειται να εκτελεστούν. Οι πράξεις καταχωρούνται σε μια κωδικοποιημένη μορφή που μετά την αποκωδικοποίησή της είναι άμεσα εκτελέσιμη από την αριθμητική και λογική μονάδα του υπολογιστικού συστήματος. Η μορφή με την οποία καταχωρούνται οι πράξεις αποτελεί μια τεχνική γλώσσα. Η γλώσσα αυτή, την οποία θεωρητικά "αντιλαμβάνεται" το υπολογιστικό σύστημα (δηλαδή η μηχανή), ονομάζεται για το λόγο αυτό "γλώσσα μηχανής" (machine language). Τα προγράμματα αποτελούνται από οδηγίες (ή εντολές) προς τη μηχανή, για τις υπό εκτέλεση πράξεις. Αποτελούνται λοιπόν από ακολουθίες εντολών που καθορίζουν τις πράξεις, τις οποίες η μηχανή πρόκειται να εκτελέσει.

Η μνήμη του ηλεκτρονικού υπολογιστικού συστήματος μοιάζει με μια κυψέλη, που αποτελείται από ένα οργανωμένο πλήθος κυψελίδων, ή κελιών ή στοιχείων. Κάθε στοιχείο της μνήμης μπορεί να βρίσκεται σε μία από τις δύο δυνατές καταστάσεις ή μεταφορικά να περιέχει μία από τις δύο δυνατές τιμές.

Παλαιότερα, η υλοποίηση των στοιχείων της μνήμης πραγματοποιούνταν με μικροσκοπικούς μαγνητικούς δακτυλίους. Οι τελευταίοι μαγνητίζονταν είτε δεξιόστροφα, για να αποδώσουν τη μία κατάσταση, είτε αριστερόστροφα για να αποδώσουν την άλλη. Για τη

μαγνήτισή του χρησιμοποιούνταν ηλεκτροφόροι αγωγοί, που περνούσαν από κέντρο τους. Η μαγνήτιση, δεξιόστροφη ή αριστερόστροφη ήταν συνάντηση της φοράς του ηλεκτρικού ρεύματος που διαπερνούσε τους αγωγούς.

Η μνήμη, που αποτελείται από μαγνητικούς δακτυλίους, ονομάζεται μαγνητική μνήμη (core). Η ανάκληση της τιμής που περιέχει ένας μαγνητικός δακτύλιος πραγματοποιείται μέσω ενός δεύτερου αγωγού αλλά αυτό έχει σαν αποτέλεσμα την απώλεια της τιμής του και για το λόγο αυτό απαιτείται ένας επιπλέον αγωγός, για να επαναφέρει στο δακτύλιο τη προηγούμενη τιμή του.

Ο μαγνητισμός των μαγνητικών δακτυλίων ήταν μόνιμος και μπορούσε να μεταβληθεί με αλλαγή της φοράς του ρεύματος του αγωγού που τον διαπερνούσε. Δεν ήταν συνεπώς αναγκαία η διατήρηση της μαγνητικής μνήμης με συνεχή τροφοδοσία ηλεκτρικού ρεύματος. Απλώς, η μαγνητική μνήμη ήταν δυνατόν να παραμένει αμετάβλητη και να κρατά συνεπώς πληροφοριακά δεδομένα για όσο διάστημα ήταν επιθυμητό. Αυτό είχε σαν αποτέλεσμα την επάνοδο στην προηγούμενη κατάσταση, στην περίπτωση που η μηχανή, για οποιοδήποτε λόγο και κυρίως μετά τη διακοπή της τροφοδοσίας της με ηλεκτρικό ρεύμα, διέκοπτε την λειτουργία της. Για το λόγο αυτό η μαγνητική μνήμη και γενικά κάθε είδος μνήμης που δεν απαιτεί τροφοδοσία για να διατηρήσει τα στοιχεία που περιέχει, ονομάζεται "σταθερή μνήμη" (non volatile storage).

Σήμερα, τα στοιχεία μνήμης υλοποιούνται μέσω ημιαγωγικών (semi-conductor) ολοκληρωμένων κυκλωμάτων πυριτίου (τσιπ) που έχουν σαν ελάχιστο στοιχείο το flip-flop. Το χαμηλό δυναμικό (τάση) υλοποιεί την μία κατάσταση και το υψηλό την άλλη. Το flip-flop θεωρείται, όπως βέβαια και ο μαγνητικός δακτύλιος, ως το ελάχιστο στοιχείο μνήμης.

Βασική διαφορά των flip-flops από τους μαγνητικούς δακτυλίους, είναι η μεγάλη τους ταχύτητα και ο μικρός τους όγκος, που συνεχώς μειώνεται λόγω της αύξησης της ολοκλήρωσης. Ωστόσο, η ημιαγωγική μνήμη δεν είναι σταθερή και για να διατηρηθεί απαιτεί διαρκή τροφοδοσία με ηλεκτρικό ρεύμα. Είναι δηλαδή ασταθής μνήμη (volatile storage). Σήμερα χρησιμοποιούνται διάφορες τεχνολογίες ημιαγωγικών μνημών, από τις οποίες οι περισσότερο γρήγορες βρίσκονται στην αριθμητική και λογική μονάδα των ηλεκτρονικών υπολογιστικών συστημάτων, ενώ οι λιγότερο γρήγορες που κατασκευάζονται από MOS (Metal-Oxide Semiconductor, ημιαγωγούς οξειδίων μετάλλου) βρίσκονται στη κεντρική μνήμη.

Τα ολοκληρωμένα κυκλώματα που χρησιμοποιούνται στην κεντρική μνήμη των υπολογιστικών συστημάτων, διακρίνονται στα δυναμικά και τα στατικά και η αντίστοιχη μνήμη σε δυναμική (dynamic memory) και στατική (static memory). Τα στοιχεία από τα οποία αποτελείται η δυναμική μνήμη περιέχουν ένα τρανζίστορ, που ενεργεί σαν διακόπτης και ένα πυκνωτή στον οποίο καταχωρείται ηλεκτρικό φορτίο. Ο διακόπτης κάθε στοιχείου, δηλαδή το τρανζίστορ, καθορίζει αν ο πυκνωτής έχει φορτίο ή όχι, όπως περίπου ένας διακόπτης μιας ηλεκτρικής λυχνίας καθορίζει αν η λυχνία φωτίζει ή όχι. Η δυναμική μνήμη είναι ασταθής μνήμη, εξαιτίας της απώλειας του φορτίου του

πυκνωτή και απαιτεί περιοδική αναπαραγωγή του φορτίου (refresh, φρεσκάρισμα) για να διατηρείται. Η στατική μνήμη είναι και αυτή ασταθής, αλλά δεν απαιτεί αναπαραγωγή των τιμών που περιέχουν τα στοιχεία της. Απαιτεί όμως τροφοδοσία με ρεύμα. Τα στοιχεία της στατικής μνήμης είναι περισσότερο πολύπλοκα, περιέχουν περισσότερα από ένα τρανζίστορ και είναι μεγαλύτερα σε όγκο. Για το λόγο αυτό, η στατική μνήμη χρησιμοποιείται σε ειδικές περιπτώσεις, ενώ η δυναμική μνήμη χρησιμοποιείται κυρίως στη κεντρική μνήμη των ηλεκτρονικών υπολογιστικών συστημάτων.

2.3. Οργάνωση της Μνήμης

Οι δύο διαφορετικές καταστάσεις, στις οποίες είναι δυνατόν να βρίσκεται το ελάχιστοι στοιχείο μνήμης, συμβολίζονται με τα δυαδικά ψηφία 0 και 1. Ο συμβολισμός είναι ιδιαίτερα πρακτικός και έχει επικρατήσει. Θεωρούμε ότι τα δυαδικά ψηφία 0 και 1, που αποτελούν το σύνολο {0,1} είναι ένα αλφάβητο δύο συμβόλων. Με το αλφάβητο αυτό είναι δυνατό να κωδικοποιηθεί οποιοδήποτε πληροφοριακό δεδομένο και να εισαχθεί και καταχωρηθεί σε ένα υπολογιστικό σύστημα.

Bit:

Συνεπώς, το ελάχιστο στοιχείο μνήμης του ηλεκτρονικού υπολογιστή, μπορεί να αποδώσει και να αποθηκεύσει δύο απλές καταστάσεις που αντιστοιχούν σε ένα δυαδικό ψηφίο. Για το λόγο αυτό το ελάχιστο στοιχείο μνήμης ονομάζεται bit. Το όνομα bit προέρχεται από το 1ο και τα δύο τελευταία γράμματα του όρου Binary digIT, που σημαίνει δυαδικό ψηφίο.

Λέξη:

Μια λέξη (word) είναι η ακολουθία από bit. Το πλήθος N των bit που περιέχονται σε μια λέξη, είναι χαρακτηριστικό για κάθε είδος μηχανής. Κάθε ένα από τα bit μιας λέξης μπορεί να βρίσκεται στη κατάσταση 0 ή στη κατάσταση 1.

Για το λόγο αυτό, μια λέξη μπορεί να αποδώσει μέχρι 2^N το πολύ διαφορετικές καταστάσεις, οι οποίες αντιστοιχούν στους δυνατούς συνδυασμούς των καταστάσεων των N bit. Με τον τρόπο αυτό μια λέξη είναι δυνατόν να παραστήσει και 2^N διαφορετικά πληροφοριακά στοιχεία, αν κάθε ένα αποδοθεί με ένα διαφορετικό συνδυασμό καταστάσεων των bit της λέξης

Ο χαρακτηριστικός αριθμός N, που αποδίδει το πλήθος των bit ανά λέξη, ποικίλλει στους διάφορους τύπους μηχανών. Υπάρχουν μηχανές διαφόρων κατηγοριών με 4, 8, 12, 16, 18, 24, 32, 36, 48, 64 bit ανά λέξη.

Byte:

Μια τυπική υποδιαίρεση των λέξεων, που αποτελεί την πιο συνηθισμένη ομαδοποίηση των bit, είναι μια ακολουθία από 8 bit, που ονομάζεται byte. Σύμφωνα με αυτό, μια λέξη των 8 bit αποτελείται από 1 byte, μια λέξη των 16 bit αποτελείται από 2 bytes, ενώ μια λέξη των 32 bit αποτελείται από 4 bytes.

Η χρήση του όρου «byte» έχει ελαχιστοποιήσει τη χρήση του όρου «λέξη», που δεν είναι σαφώς καθορισμένη και το μέγεθός της ποικίλλει στους διάφορους τύπους μηχανών. Η έννοια της λέξης θεωρείται επίσης ως περισσότερο κατανοητή στους ειδικούς, απ' ότι στους απλούς χρήστες των υπολογιστικών συστημάτων. Αντίθετα, το byte έχει επικρατήσει ως όρος και αποδίδει μια ακολουθία από 8 bit.

Διεύθυνση θέσης μνήμης

Είναι γνωστό, ότι η μνήμη των υπολογιστικών συστημάτων χρησιμοποιείται για την καταχώρηση των εντολών των προγραμμάτων και των πληροφοριακών δεδομένων. Για να είναι όμως οι καταχωρήσεις αυτές χρήσιμες, θα πρέπει να είναι δυνατή η ανάκληση τους και η μεταφορά τους στην Κεντρική Μονάδα Επεξεργασίας, ή σε άλλη μονάδα του υπολογιστικού συστήματος. Θα πρέπει συνεπώς να παρέχεται η δυνατότητα προσδιορισμού των θέσεων της μνήμης, που περιέχουν τις υπό εκτέλεση εντολές καθώς και τα πληροφοριακά δεδομένα.

Είναι λογικό, ότι η δυνατότητα αυτή απαιτεί οργάνωση της μνήμης με τρόπο, που να επιτρέπει τον προσδιορισμό των θέσεων στις οποίες πρόκειται να γίνουν καταχωρήσεις, αλλά και των θέσεων από τις οποίες πρόκειται να γίνουν ανακλήσεις. Για το λόγο αυτό, κάθε θέση μνήμης προσδιορίζεται με μια συγκεκριμένη διεύθυνση. Η διεύθυνση αυτή είναι αριθμητική και προσδιορίζει την κάθε θέση.

Κάθε byte στην μνήμη μιας υπολογιστικής μηχανής έχει μια διεύθυνση με την οποία μπορεί να αναζητηθεί. Η διεύθυνση ενός byte αποδίδεται με μία διάταξη των στοιχείων {0,1} ανά k. Είναι δηλαδή ένας δυαδικός αριθμός με k ψηφία. Ο αριθμός k είναι ακέραιος και χαρακτηριστικός για κάθε τύπο μηχανής. Ο χαρακτηριστικός αριθμός k είναι καθοριστικός για το δυνατό μέγεθος της μνήμης.

Για παράδειγμα, αν k=2, τότε οι δυνατές διατάξεις και συνεπώς διευθύνσεων είναι 4 (δηλαδή 2^2). Οι διατάξεις αυτές είναι οι ακόλουθες:
00, 01, 10, 11
Αν k=3 οι διατάξεις έχουν πλήθος 8 (2^3) και είναι οι εξής:
000, 001, 010, 011, 100, 101, 110, 111.

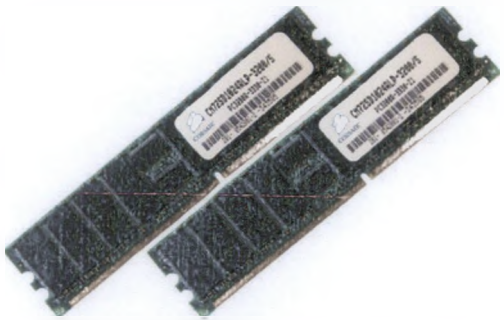
Γενικά, ο μέγιστος αριθμός των διαφορετικών διευθύνσεων, που ισοδυναμεί με το πλήθος των διαφορετικών διατάξεων, είναι 2^k. Αυτό δεν σημαίνει πάντως ότι κάθε υπολογιστικό σύστημα στο οποίο χρησιμοποιούνται k ψηφία, για την απόδοση των διευθύνσεων, έχει μνήμη χωρητικότητας 2^k

bytes. Απλώς σημαίνει ότι δεν είναι δυνατό να αποκτήσει περισσότερη μνήμη, δεδομένου ότι δεν έχει δυνατότητα διεύθυνσης των επιπλέον θέσεων.

Οι διευθύνσεις της μνήμης είναι διαδοχικοί ακέραιοι αριθμοί εκφρασμένοι στο δυαδικό σύστημα αριθμήσεως. Η διεύθυνση του πρώτου byte της μνήμης είναι πάντα ο αριθμός μηδέν. Η διεύθυνση του τελευταίου byte της μνήμης είναι M-1, αν το σύνολο των θέσεων της μνήμης περιέχει M bytes.

Ένα υπολογιστικό σύστημα με διεύθυνση μνήμης μήκους 16 bit έχει δυνατότητα διεύθυνσης 2^16 = 65536 bytes. Αυτό σημαίνει, ότι το υπολογιστικό σύστημα αυτό μπορεί να διαθέτει μνήμη μικρότερη ή ίση με 65536 bytes, αλλά όχι μεγαλύτερη. Η πρώτη θέση της μνήμης έχει τη διεύθυνση μηδέν, που αποδίδεται με μια ακολουθία 16 ψηφίων 0, ενώ η τελευταία θέση έχει τη διεύθυνση 65536 και αποδίδεται με 16 ψηφία 1. Επίσης, μια μηχανή με 24 bit διεύθυνση μπορεί να διευθύνει μνήμη μέχρι 16777216 bytes και οι διευθύνσεις της πρώτης και της τελευταίας θέσης αποδίδονται με 24 ψηφία μηδέν και 24 ψηφία 1 αντιστοίχως.

Οι προσπελάσεις (access) σε θέσεις της μνήμης για ανάκληση ή για καταχώρηση πληροφοριακών δεδομένων, είναι δυνατόν να γίνονται σε ολόκληρες λέξεις ή σε bytes της μνήμης. Οι προσπελάσεις σε χαμηλότερο επίπεδο, π.χ. bit είναι δυνατές, αλλά πραγματοποιούνται πάντα μέσω του byte.



2.4. Μέτρηση της χωρητικότητας της μνήμης

Το byte είναι η μονάδα μέτρησης της χωρητικότητας της μνήμης. Πολλαπλάσιο του byte είναι ο αριθμός 1024, που αποδίδεται με το γράμμα K και προφέρεται Kilo. Το πολλαπλάσιο K, που δεν σημαίνει 1000, αλλά 1024, προκύπτει από την ύψωση του αριθμού 2 στη δεκάτη δύναμη (2^10). Το γράμμα K τοποθετείται πριν από τη λέξη byte (Kbyte), ή πριν από το γράμμα b (Kb) και προφέρεται "kilobyte".

Το γράμμα M (Mega), που σε άλλες περιπτώσεις χρησιμοποιείται για να αποδώσει εκατομμύρια, όταν αναφέρεται σε μεγέθη μνήμης υπολογιστικών συστημάτων δεν σημαίνει 1.000 x 1.000, αλλά 1024 χ 1024, δηλαδή 2^20 ή

1.048.576. Όπως και στην περίπτωση του K έτσι και με . το M, τα πολλαπλάσια του byte γράφονται Mbyte, ή Mb και προφέρονται "Megabyte".

Τέλος, το γράμμα G είναι το πρώτο γράμμα της λέξης Giga (1 Giga = 1.024 Mega = 1.073.741.824) και σημαίνει 10243, ή 1024 Mega και χρησιμοποιείται όταν γίνεται αναφορά σε μνήμες πολύ μεγάλης χωρητικότητας, όπως οι περιφερειακές μνήμες π.χ. μαγνητικοί δίσκοι η άλλα μέσα. Τα πολλαπλάσια του byte γράφονται Gbyte, ή Gb και προφέρονται "Gigabyte".

Σύμφωνα με αυτά η χωρητικότητα μνήμης 65.536 byte γράφεται 65 Kb ή 65 Kbyte. Αντιστοίχως, η χωρητικότητα μνήμης 16.777.216 byte γράφεται 16 Mb, ή 16 Mbyte.

2.5. Περιεχόμενα της μνήμης

Κάθε πληροφοριακό δεδομένο, που καταχωρείται σε μια θέση της μνήμης και αποτελείται από ένα ή περισσότερα bytes, παραμένει αμετάβλητο μέχρι να καταχωρηθεί ένα νέο πληροφοριακό δεδομένο στην ίδια θέση. Είναι επίσης δυνατή, σ' οποιαδήποτε στιγμή, η ανάκληση και χρησιμοποίηση ενός πληροφοριακού δεδομένου, από τη θέση της μνήμης που το περιέχει, αρκεί να είναι γνωστή η διεύθυνση της. Μετά την ανάκληση του πληροφοριακού δεδομένου, το περιεχόμενο της θέσης μνήμης που το περιέχει παραμένει αμετάβλητο, επειδή κατά την ανάκληση χρησιμοποιείται μόνο ένα αντίγραφο του πληροφοριακού δεδομένου.

Στην κεντρική μνήμη ενός υπολογιστικού συστήματος καταχωρούνται ποικίλα πληροφοριακά δεδομένα, τα οποία μπορεί να είναι αριθμοί, γράμματα, συνδυασμοί αυτών σε διάφορες μορφές, άλλοι χαρακτήρες ή σύμβολα, καθώς ακόμη και εντολές στη γλώσσα μηχανής. Όλα αυτά τα πληροφοριακά δεδομένα, καθώς και οι εντολές, είναι κωδικοποιημένα στη γλώσσα των δύο στοιχείων, στη γλώσσα δηλαδή του 0 και του 1. Συνεπώς, στην κεντρική μνήμη των υπολογιστικών συστημάτων περιέχονται και συνυπάρχουν τα προγράμματα, τα δεδομένα των προγραμμάτων, τα αποτελέσματα των ενδιαμέσων υπολογισμών καθώς και τα τελικά αποτελέσματα, κωδικοποιημένα πάντα με ακολουθίες των ψηφίων 0 και 1.

Σύμφωνα με τα προηγούμενα, η κεντρική μνήμη κάθε υπολογιστικού συστήματος έχει συγκεκριμένη χωρητικότητα. Είναι συνεπώς αντιληπτό, ότι δεν είναι δυνατή η καταχώρηση στη μνήμη όλων των χρησιμοποιουμένων πληροφοριακών δεδομένων και προγραμμάτων, αλλά μόνον εκείνων που είναι, σε κάθε περίπτωση, απολύτως απαραίτητα.

Απαραίτητη είναι η παραμονή των προγραμμάτων κατά τη διάρκεια της εκτέλεσης τους, καθώς επίσης και ενός συνόλου προγραμμάτων, που πρέπει να είναι μονίμως εγκατεστημένα στη μνήμη όσο το υπολογιστικό σύστημα λειτουργεί. Τέλος, στη μνήμη παραμένουν πληροφοριακά δεδομένα των εκτελουμένων προγραμμάτων, καθώς και μερικά ακόμη στοιχεία από τα οποία εξαρτάται η λειτουργία του υπολογιστικού συστήματος.

Οτιδήποτε άλλο, όπως π.χ. τα προγράμματα των χρηστών όταν δεν εκτελούνται, σύνολα πληροφοριακών δεδομένων και διάφορα άλλα προγράμματα, βρίσκονται αποθηκευμένα σε εφεδρικές μνήμες, που συνήθως ονομάζονται δευτερεύουσες ή περιφερειακές μονάδες μνήμης. Τυπικές περιφερειακές μονάδες μνήμης αποτελούν σήμερα οι μαγνητικοί δίσκοι και οι μαγνητικές ταινίες.

2.6. Συσκευασίες μνήμης

Η μνήμη είναι ένα ολοκληρωμένο κύκλωμα, το οποίο πρέπει να έχει κατάλληλη μορφή, για να μπορεί να χρησιμοποιηθεί σε μια ηλεκτρονική συσκευή όπως ο προσωπικός υπολογιστής. Συνήθως, περικλείεται από ένα περίβλημα το οποίο σε συγκεκριμένα σημεία του έχει πλήθος από μεταλλικές επαφές, για τη σύνδεση της μνήμης με το υπόλοιπο ηλεκτρονικό κύκλωμα. Αυτό αποτελεί τη **συσκευασία** της μνήμης (memory package). Η συσκευασία αυτή, εκτός από το ολοκληρωμένο κύκλωμα της μνήμης, μπορεί να περιέχει και άλλα ηλεκτρονικά κυκλώματα που είναι απαραίτητα για τη λειτουργία της. Στη συνέχεια θα μελετήσουμε διάφορες συσκευασίες μνήμης που χρησιμοποιούνται στους προσωπικούς υπολογιστές.

Η απλούστερη συσκευασία μνήμης είναι η συσκευασία DIP (Dual Inline Package - συσκευασία διπλής σειράς επαφών). Σε αυτή το ολοκληρωμένο κύκλωμα της μνήμης βρίσκεται μέσα σε ένα παραλληλεπίπεδο πλαστικό περίβλημα, στις δύο μεγαλύτερες πλευρές του οποίου υπάρχουν από μία σειρά επαφών.

Οι σύγχρονοι προσωπικοί υπολογιστές χρησιμοποιούν μνήμη σε συσκευασία SIMM (Single Inline Memory Module -Εξάρτημα μνήμης μονής σειράς επαφών) ή συσκευασία DIMM (Dual Inline Memory Module - Εξάρτημα μνήμης διπλής σειράς επαφών). Η μνήμη SIMM/DIMM είναι στην πράξη μια μικρή πλακέτα, πάνω στην οποία είναι τοποθετημένα αρκετά ολοκληρωμένα κυκλώματα μνήμης. Η πλακέτα αυτή έχει στη μία της ακμή ένα πλήθος επαφών, που της επιτρέπουν να τοποθετείται πάνω σε ειδικές βάσεις, ώστε να επιτυγχάνεται η σύνδεση της με το υπόλοιπο ηλεκτρονικό κύκλωμα.

Η σχεδίαση της μνήμης SIMM/DIMM μας δίνει τη δυνατότητα να επιλέξουμε το πλήθος και το είδος των ολοκληρωμένων κυκλωμάτων μνήμης που θα τοποθετήσουμε στην πλακέτα της. Έτσι, μπορούμε να κατασκευάσουμε μνήμες SIMM/DIMM διαφορετικών χωρητικοτήτων, αλλά με τις ίδιες ακριβώς επαφές. Το είδος και η θέση των επαφών μιας μνήμης SIMM/DIMM είναι σαφώς καθορισμένο έτσι, ώστε σε μια βάση για μνήμη SIMM ή DIMM να μπορεί να τοποθετηθεί μνήμη SIMM ή DIMM αντίστοιχα οποιασδήποτε χωρητικότητας. Έχουμε, επομένως, τη δυνατότητα να καθορίσουμε τη συνολική χωρητικότητα της βασικής μνήμης ενός προσωπικού υπολογιστή επιλέγοντας μνήμες SIMM/DIMM κατάλληλης χωρητικότητας.

Υπάρχουν δύο είδη μνήμης SIMM, αυτή των 30 επαφών και αυτή των 72 επαφών.

Στην κάτω πλευρά της συσκευασίας αυτής υπάρχει μια σειρά από 30 επαφές. Στην πίσω πλευρά της μνήμης SIMM υπάρχει άλλη μια σειρά από 30 επαφές, οι οποίες όμως είναι όμοιες μία προς μία με τις επαφές της σειράς.

Η μνήμη SIMM 72 επαφών είναι παρόμοια με τη μνήμη SIMM 30 επαφών. Διαφέρει από την τελευταία στις διαστάσεις, αφού είναι λίγο μεγαλύτερη, και στο πλήθος των επαφών, αφού κάθε σειρά, αντί για 30, έχει 72 επαφές.

Η μνήμη DIMM 168 επαφών είναι η πιο διαδεδομένη συσκευασία μνήμης. Σε αντίθεση με τη SIMM, η μνήμη αυτή έχει στη μία ακμή της δύο διαφορετικές σειρές από 84 επαφές, μία σε κάθε πλευρά, δηλαδή συνολικά 168 διαφορετικές επαφές, αντί για 30 ή 72 "διπλές" επαφές (μία επαφή από τη μία πλευρά και μία από την άλλη) που έχουν οι μνήμες SIMM.

2.7. Χαρακτηριστικά συσκευασιών μνήμης

Στους πρώτους προσωπικούς υπολογιστές χρησιμοποιήθηκε ως κύρια μνήμη σε συσκευασία DIP. Η συσκευασία αυτή όμως έχει δύο βασικά μειονεκτήματα, μικρή χωρητικότητα, επομένως, απαιτείται μεγάλος αριθμός ολοκληρωμένων κυκλωμάτων, ώστε να επιτύχουμε την επιθυμητή συνολική χωρητικότητα της βασικής μνήμης του προσωπικού υπολογιστή, και επιπλέον τόσο το πλήθος όσο και το είδος των επαφών της διαφέρουν ανάλογα με τη χωρητικότητα της. Έτσι, είναι αδύνατη η χρησιμοποίηση μνήμης DIP ως κυρίας μνήμης στους σύγχρονους προσωπικούς υπολογιστές αφού η μνήμη αυτή πρέπει να έχει μεγάλη χωρητικότητα και παράλληλα πρέπει να υπάρχει η δυνατότητα μεταβολής της συνολικής χωρητικότητας της μνήμης τους. Στους σύγχρονους προσωπικούς υπολογιστές σε συσκευασία DIP χρησιμοποιείται μόνο η μνήμη ROM, που περιέχει το BIOS, ενώ ως κύρια μνήμη η μνήμη σε συσκευασία SIMM ή DIMM.

Η μνήμη SIMM 30 επαφών που χρησιμοποιείται στους προσωπικούς υπολογιστές είναι απλή μνήμη DRAM με χρόνο προσπέλασης 60 ή 70nsec. Η χωρητικότητα της επαφών μπορεί να είναι 256 KBytes, 1 MByte, 4 MBytes ή 16 MBytes. Η οργάνωση της είναι σε λέξεις των 8 bits. Επομένως, όταν ο προσωπικός υπολογιστής διαθέτει έναν επεξεργαστή που έχει εύρος διαδρόμου δεδομένων 16 bits, όπως οι επεξεργαστές 80286 και 80386SX, πρέπει να χρησιμοποιούνται ζευγάρια μνήμης SIMM 30 επαφών, ώστε να προκύψει το εύρος των 16 bits του επεξεργαστή. Αντίστοιχα, όταν ο προσωπικός υπολογιστής διαθέτει έναν επεξεργαστή που έχει εύρος διαδρόμου δεδομένων 32 bits, όπως οι επεξεργαστές 80386DX και i486DX, τότε πρέπει να χρησιμοποιούνται ομάδες των τεσσάρων τέτοιων μνημών. Εύκολα βρίσκουμε ότι για έναν επεξεργαστή με εύρος διαδρόμου δεδομένων 64 bits, όπως οι επεξεργαστές Pentium και Pentium II, θα έπρεπε να χρησιμοποιούνται οι μνήμες SIMM 30 επαφών σε ομάδες των οκτώ, γεγονός το οποίο, όμως, δεν είναι ιδιαίτερα βολικό.

Η μνήμη SIMM 72 επαφών που χρησιμοποιείται στους προσωπικού

υπολογιστές μπορεί να είναι μνήμη DRAM, FPM DRAM ή EDO RAM. Η χωρητικότητα της μπορεί να είναι 1 MByte, 2 MBytes, 4 MBytes, 8 MBytes 16 MBytes, 32 MBytes, 64 MBytes ή 128 MBytes με χρόνο προσπέλαση 60nsec.

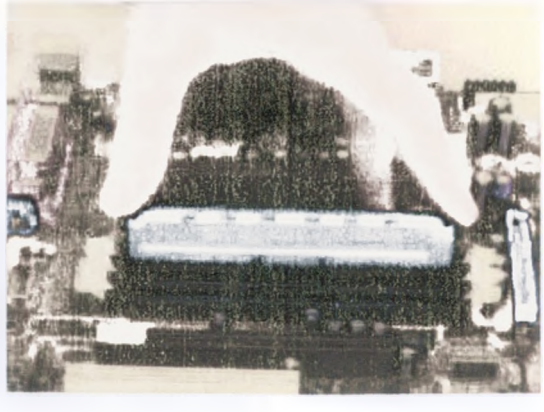
Η μνήμη SIMM 72 επαφών διαφέρει από τη μνήμη SIMM 30 επαφών προς την οργάνωση, η οποία είναι σε λέξεις των 32 bits. Για το λόγο αυτό κάθε τέτοια μνήμη μπορεί να χρησιμοποιηθεί αντί τεσσάρων μνημών 30 επαφών. Έτσι, σε επεξεργαστές με εύρος διαδρόμου δεδομένων 32 bits, όπως ο 80386DX και ο i486DX, μπορούν να χρησιμοποιηθούν μόνες τους ενώ σε επεξεργαστές με εύρος διαδρόμου δεδομένων 64 bits, όπως ο Pen tium και ο Pentium II, πρέπει να χρησιμοποιούνται σε ζεύγη. Βέβαια η μνήμη SIMM 72 επαφών δεν μπορεί να χρησιμοποιηθεί σε επεξεργαστές με εύρος διαδρόμου δεδομένων μικρότερο από 32 bits, όπως οι επεξεργαστές 8028 και 80386SX.

Η μνήμη DIMM 168 επαφών που χρησιμοποιείται στους προσωπικούς υπολογιστές είναι σχεδόν αποκλειστικά μνήμη SDRAM. Η χωρητικότητα της μνήμης αυτής μπορεί να είναι 8 MBytes, 16 MBytes, 32 MBytes, 64 MBytes 128 MBytes ή 256 MBytes.

Η οργάνωση της μνήμης DIMM 168 επαφών είναι σε λέξεις των 64 bits πράγμα που την καθιστά ιδανική για επεξεργαστές με εύρος διαδρόμου δεδομένων 64 bits, όπως οι επεξεργαστές Pentium και Pentium II. Αντίθετα δεν μπορούν να χρησιμοποιηθούν σε επεξεργαστές με μικρότερο εύρος διαδρόμου δεδομένων όπου αναγκαστικά χρησιμοποιούνται μνήμες 30 και 72 επαφών.

Συσκευασία	Είδος	Οργάνωση	Επεξεργαστές
SIMM 30 επ.	DRAM	8 bits	80386,80486,5*86
SIMM 72 επ.	DRAM,EDO,RAM, FPM DRAM	32 bits	80486,5*86, Pentium, K5
SIMM 168 επ.	SDRAM	64 bits	Pentium, K5 Pentium Pro Pentium || Pentium||| Celeron

2.8. Τοποθέτηση μνήμης



Κάθε συσκευασία μνήμης έχει διαφορετικό τρόπο σύνδεσης με το υπόλοιπο υπολογιστικό σύστημα. Έτσι, η μνήμη DIP μπορεί είτε να κολληθεί απευθείας πάνω στην πλακέτα του υπολογιστή είτε να τοποθετηθεί πάνω σε ειδική βάση, η οποία με τη σειρά της είναι κολλημένη πάνω στην πλακέτα. Η βάση αυτή μας επιτρέπει να τοποθετούμε και να αφαιρούμε εύκολα τη μνήμη DIP- Στη μία

άκρη της μνήμης DIP υπάρχει πάντοτε ένα σημάδι, συνήθως μια μικρή βούλα ή μια εγκοπή. Το σημάδι αυτό δείχνει την πλευρά της μνήμης που υπάρχει η επαφή 1 της μνήμης. Αντίστοιχο σημάδι υπάρχει στη βάση της μνήμης DIP και μας υποδεικνύει το σωστό τρόπο τοποθέτησης της. Προσοχή πρέπει να δοθεί κατά την εισαγωγή ή την εξαγωγή της μνήμης στη βάση, γιατί υπάρχει κίνδυνος να στραβώσει ή να αποκοπεί κάποια από τις επαφές της.

Οι μνήμη SIMM τοποθετείται πάντα πάνω στην ειδική βάση. Στην αριστερή πλευρά της μνήμης αυτής υπάρχει μια εγκοπή, η οποία δείχνει την πλευρά στην οποία βρίσκεται η επαφή 1 της μνήμης. Αντίστοιχα, στη βάση της υπάρχει ένα εξόγκωμα στην ίδια πλευρά, το οποίο εμποδίζει τη λανθασμένη τοποθέτηση της. Η βάση αυτή διαθέτει επίσης στις άκρες της δύο ελάσματα, τα οποία χρησιμεύουν για την ακινητοποίηση της μνήμης μετά την τοποθέτηση της. Η διαδικασία τοποθέτησης είναι απλή: Κρατώντας τη μνήμη υπό γωνία περίπου 45ο ως προς τη βάση, ακουμπάμε τις επαφές της στις αντίστοιχες επαφές της βάσης, φροντίζοντας η εγκοπή της να βρίσκεται από την πλευρά που υπάρχει το εξόγκωμα της βάσης. Στη συνέχεια, ανορθώνουμε τη μνήμη μέχρι την κατακόρυφη θέση, οπότε τα ελάσματα της βάσης την "κλειδώνουν" στη θέση της. Για να αφαιρέσουμε τη μνήμη SIMM από τη βάση της, ακολουθούμε την αντίστροφη πορεία, τραβώντας πρώτα τα ελάσματα της βάσης προς τα έξω ώστε να ελευθερώσουν τη μνήμη.

Παρόμοια είναι και η διαδικασία τοποθέτησης της μνήμης DIMM στην αντίστοιχη βάση. Η μνήμη DIMM 168 επαφών έχει στο κάτω μέρος της δύο εγκοπές, μία στο κέντρο και μία ασύμμετρα τοποθετημένη προς την πλευρά που βρίσκεται η επαφή 1, ώστε να αποκλείεται η λανθασμένη τοποθέτηση της στη βάση. Η τοποθέτηση γίνεται ως εξής: Ευθυγραμμίζουμε τη μνήμη DIMM πάνω από τη βάση, φροντίζοντας οι εγκοπές της να βρίσκονται πάνω από τα αντίστοιχα εξογκώματα της βάσης. Στη συνέχεια «συρταρώνουμε» τη μνήμη στη βάση τραβώντας παράλληλα τα άκρα των δύο μοχλών που διαθέτει η βάση προς τα πάνω. Τα εξογκώματα που έχουν οι μοχλοί αυτοί μπαίνουν μέσα στις αντίστοιχες εγκοπές που έχει η μνήμη DIMM στα πλαϊνά της άκρα, ώστε να την «κλειδώσουν» στη θέση της. Για να αφαιρέσουμε τη μνήμη DIMM από τη βάση της, αρκεί να σπρώξουμε τους δύο μοχλούς προς τα κάτω, οπότε η μνήμη DIMM ελευθερώνεται.

2.9. Μνήμες

Οι μνήμες χρησιμοποιούνται στους υπολογιστές για την προσωρινή ή τη μόνιμη αποθήκευση του κώδικα του προγράμματος που εκτελείται και των δεδομένων του. Οι βασικότερες κατηγορίες μνήμης είναι δυο, η μνήμη Ram (Random Access Memory- μνήμη τυχαίας προσπέλασης) και η μνήμη Rom (Read Only Memory- μνήμη μόνο ανάγνωσης).

2.10. Διάδρομοι περιφερειακών και κάρτες επέκτασης

Ένας προσωπικός υπολογιστής, εκτός από τον επεξεργαστή και τη μνήμη, διαθέτει ακόμα ένα πλήθος περιφερειακών μονάδων. Οι περιφερειακές

μονάδες είναι κυκλώματα και συσκευές που είναι απαραίτητα για τη λειτουργία του προσωπικού υπολογιστή. Τέτοιες είναι τα κυκλώματα για τη διασύνδεση της οθόνης και του πληκτρολογίου, τα κυκλώματα για τη διασύνδεση των αποθηκευτικών μέσων, οι θύρες εισόδου / εξόδου κλπ. Είναι αυτονόητο ότι για να λειτουργήσει ο προσωπικός υπολογιστής πρέπει οι περιφερειακές μονάδες να επικοινωνούν με τον επεξεργαστή και τη μνήμη. Επειδή όμως οι περιφερειακές μονάδες είναι συνήθως αργές συγκριτικά με τον επεξεργαστή και δεν μπορούν να λειτουργήσουν στις μεγάλες συχνότητες που λειτουργεί ο επεξεργαστής και η μνήμη, οι προσωπικοί υπολογιστές σχεδιάζονται με διαδρόμους ειδικά για τις περιφερειακές μονάδες. Οι διάδρομοι αυτοί ονομάζονται **διάδρομοι περιφερειακών.** Λειτουργούν σε χαμηλότερες συχνότητες από αυτές του διαδρόμου του επεξεργαστή και αυτούς συνδέονται οι περιφερειακές μονάδες. Οι διάδρομοι περιφερειακών συνδέονται με τον διάδρομο του επεξεργαστή μέσω ειδικών κυκλωμάτων που ονομάζονται κυκλώματα ελέγχου του διαδρόμου περιφερειακών.

Η βασική δομή ενός προσωπικού υπολογιστή περιλαμβάνει συνήθως λίγες περιφερειακές μονάδες. Ο προσωπικός υπολογιστής όμως πρέπει να είναι επεκτάσιμος, να υπάρχει, δηλαδή, η δυνατότητα να προστεθούν αργότερα νέες περιφερειακές μονάδες, που να του δίνουν επιπλέον δυνατότητες. Τέτοιες περιφερειακές μονάδες μπορεί να είναι συσκευές modem, κυκλώματα παραγωγής ήχου ή κυκλώματα για την καταγραφή ψηφιακής εικόνας. Οι νέες αυτές περιφερειακές μονάδες πρέπει να συνδέονται σε κάποιο διάδρομο περιφερειακών. Για το σκοπό αυτό, οι διάδρομοι περιφερειακών του προσωπικού υπολογιστή διαθέτουν ειδικές υποδοχές, οι οποίες ονομάζονται **υποδοχές επέκτασης** (expansion slots).

Τα κυκλώματα αυτών των περιφερειακών μονάδων είναι τοποθετημένα πάνω σε ειδικές πλακέτες, οι οποίες ονομάζονται **κάρτες επέκτασης** (expansion cards). Οι κάρτες επέκτασης έχουν στη μια τους πλευρά κατάλληλες επαφές, ώστε να μπορούν να συνδέονται στις υποδοχές επέκτασης των διαδρόμων περιφερειακών του προσωπικού υπολογιστή. Οι κάρτες επέκτασης είναι αυτές που κάνουν τον προσωπικό υπολογιστή πολύ ευέλικτο αφού έχουμε τη δυνατότητα να του προσθέσουμε (με τη μορφή καρτών επέκτασης) κυκλώματα τα οποία δεν υπάρχουν στη βασική του δομή. Με τη μορφή καρτών επέκτασης όμως κατασκευάζονται και ,;~,ποια κυκλώματα που είναι απολύτως απαραίτητα για τη λειτουργία του προσωπικού υπολογιστή, όπως είναι η κάρτα γραφικών. Η κάρτα αυτή περιέχει τα απαραίτητα κυκλώματα για τη σύνδεση της οθόνης με το υπόλοιπο υπολογιστικό σύστημα. Τα κυκλώματα αυτά θα ήταν αναμενόμενο να συμπεριλαμβάνονται στη βασική δομή του προσωπικού υπολογιστή. Αντ' αυτού όμως έχουν τη μορφή κάρτας επέκτασης, πράγμα που επιτρέπει στον κατασκευαστή του προσωπικού υπολογιστή να επιλέγει την κάρτα γραφικών που θα χρησιμοποιήσει μεταξύ πολλών καρτών γραφικών διαφόρων δυνατοτήτων. Η σχεδίαση αυτή προσδίδει μεγάλη ευελιξία στον προσωπικό υπολογιστή, οι δυνατότητες του οποίου μπορούν να προσαρμοστούν στις ανάγκες του χρήστη με κατάλληλη επιλογή καρτών επέκτασης.

2.11. Σύγκριση διαδρομών περιφερειακών

Στο χώρο των προσωπικών υπολογιστών υπάρχει ένα πλήθος διαδρόμων περιφερειακών, καθένας με τα δικά του χαρακτηριστικά και, επομένως, τη δική του χρήση. Άλλοι είναι αργοί αλλά απλή στη σχεδίαση τους (όπως ο διάδρομος ISA), ενώ άλλοι είναι γρήγοροι αλλά απαιτούν πολλά επιπλέον κυκλώματα (όπως ο διάδρομος AGP). Άλλοι είναι εξειδικευμένοι (όπως ο διάδρομος AGP που χρησιμοποιείται αποκλειστικά για τη σύνδεση καρτών γραφικών), ενώ άλλοι είναι γενικής χρήσης (όπως ο διάδρομος PCI).

Διάδρομοι	Εύρος δεδομένων	Συχνότητα λειτουργίας	Ταχύτητα μεταφοράς δεδομένων
ISA 8 bits	8 bits	Έως 8,33 MHz	4 MBytes/sec
ISA 16 bits	16 bits	Έως 8,33 MHz	8 MBytes/sec
VL-BUS	32 bits	Ίδια με τον επεξεργαστή	160 MBytes/sec στα 40 MHz
PCI	32 bits	33 MHz	133 MBytes/sec
AGP	32 bits	66 MHz	266MBytes/sec (1x) 533 MByte/sec (2x) 1066 MBytes (4x)

Χαρακτηριστικά διαδρόμων προσωπικών υπολογιστών

Ο διάδρομος ISA παρ' όλο που είναι ο πιο αργός απ' όλους τους διαδρόμους περιφερειακών, χρησιμοποιείται ευρέως, ακόμα και στους" σύγχρονους υπολογιστές, γιατί είναι πολύ απλός και, επομένως, η κατασκευή καρτών επέκτασης ISA είναι πολύ χαμηλού κόστους. Έτσι, σε μορφή καρτών επέκτασης ISA κατασκευάζονται περιφερειακές μονάδες, για τις οποίες η χαμηλή ταχύτητα του διαδρόμου ISA δεν αποτελεί περιορισμό. Τέτοιες περιφερειακές μονάδες είναι, για παράδειγμα, συσκευές modem, των οποίων η ταχύτητα είναι έτσι κι αλλιώς πολύ χαμηλή.

Ο διάδρομος PCI είναι ο καταλληλότερος διάδρομος περιφερειακών για μονάδες που απαιτούν μεγάλη ταχύτητα μεταφοράς δεδομένων με τον επεξεργαστή και τη μνήμη. Τέτοιες περιφερειακές μονάδες είναι οι ελεγκτές αποθηκευτικών μέσων, τα κυκλώματα ψηφιοποίησης και αναπαραγωγής εικόνας video και οι κάρτες γραφικών. Αυτές οι περιφερειακές μονάδες έχουν μεγάλες απαιτήσεις ταχύτητας, οπότε η χρήση του διαδρόμου PCI είναι απαραίτητη.

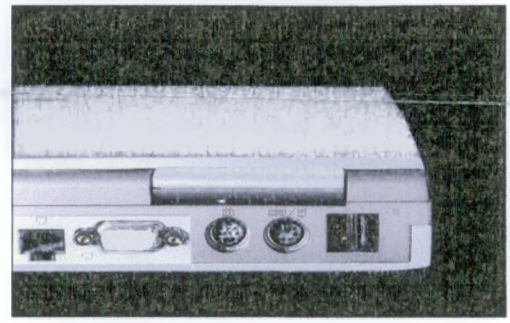
Οι πιο απαιτητική σε ταχύτητα κάρτα επέκτασης σε έναν προσωπικό υπολογιστή είναι η κάρτα γραφικών. Η πληροφορία που μεταφέρεται από τη μνήμη και τον επεξεργαστή προς αυτήν είναι πολύ μεγάλη. Επομένως, για τη γρήγορη μεταφορά της απαιτείται ένας διάδρομος περιφερειακών αυξημένων δυνατοτήτων. Αυτήν την ανάγκη καλείται να καλύψει ο διάδρομος AGP, καθώς η ταχύτητα μεταφορά δεδομένων μπορεί να φτάσει και τα 1066 MBytes/sec.

Ο διάδρομος VL-BUS δεν χρησιμοποιείται πια στους προσωπικούς υπολογιστές, γιατί από πλευράς κόστους και ταχύτητας βρίσκεται μεταξύ του διαδρόμου ISA και του διαδρόμου PCI. Έτσι, όταν απαιτείται χαμηλό κόστος,

αντ' αυτού χρησιμοποιείται ο διάδρομος ISA, ενώ όταν απαιτείται υψηλή ταχύτητα, χρησιμοποιείται ο διάδρομος PCI. Επιπλέον, επειδή οι κάρτες επέκτασης VL-BUS παρουσιάζουν δυσκολίες στην κατασκευή, τη ρύθμιση και τη λειτουργία τους, αποφεύγεται η χρήση τους.

Οι σύγχρονοι προσωπικοί υπολογιστές διαθέτουν συνήθως ένα διάδρομο ISA για τις αργές και χαμηλού κόστους περιφερειακές μονάδες, ένα διάδρομο PCI για τις γρήγορες περιφερειακές μονάδες και ένα διάδρομο AGP για την κάρτα γραφικών.

2.12. Θύρες επικοινωνίας



Ένα μεγάλο μέρος της λειτουργικότητας των προσωπικών υπολογιστών οφείλεται στην ποικιλία των περιφερειακών συσκευών που μπορούμε να συνδέσουμε σε αυτούς. Εξάλλου η μεγάλη ποικιλία περιφερειακών συσκευών επιτρέπει τη χρήση των προσωπικών υπολογιστών σε μεγάλο αριθμό διαφορετικών εφαρμογών.

Ο προσωπικός υπολογιστής διαθέτει θύρες (ports), για να επικοινωνεί με τις περιφερειακές συσκευές ή με άλλες ηλεκτρονικές συσκευές και υπολογιστές.

Οι θύρες επικοινωνίας του προσωπικού υπολογιστή είναι τυποποιημένες. Με άλλα λόγια, ο τρόπος σύνδεσης, η ταχύτητα, τα σήματα που ανταλλάσσονται και γενικώς τα χαρακτηριστικά μίας θύρας επικοινωνίας του υπολογιστή είναι αυστηρώς προδιαγεγραμμένα. Αυτό δίνει τη δυνατότητα να μπορούν να συνδέονται πάνω στην ίδια θύρα διαφορετικές περιφερειακές συσκευές και μάλιστα διαφορετικών κατασκευαστών, αρκεί να είναι σχεδιασμένες σύμφωνα με το πρότυπο που απαιτεί η θύρα αυτή. Έτσι, για παράδειγμα, στην παράλληλη θύρα του υπολογιστή μπορούμε να συνδέσουμε έναν εκτυπωτή ή ένα σαρωτή (scanner) ή ακόμη και μια ψηφιακή κάμερα.

Υπάρχουν τρεις τυποποιημένες θύρες, που έχουν επικρατήσει στο χώρο των προσωπικών υπολογιστών:

1. Η παράλληλη θύρα
2. Η σειριακή θύρα και
3. Η θύρα USB.

Καθεμιά από αυτές διαθέτει ένα ειδικό προσαρμοστικό κύκλωμα (interface) για την επικοινωνία με την ΚΜΕ μέσω του διαδρόμου του υπολογιστή.

Τα προσαρμοστικά κυκλώματα παλιότερα ήταν σε μορφή κάρτας επέκτασης και συνδέονταν πάνω σε μια υποδοχή επέκτασης του

υπολογιστή. Εδώ και αρκετό καιρό για λόγους οικονομίας χώρου τα προσαρμοστικά κυκλώματα έχουν συμπεριληφθεί στα κυκλώματα υποστήριξης της μητρικής πλακέτας. Βεβαία διατίθενται ακόμα κάρτες με προσαρμοστικά κυκλώματα θυρών, στην περίπτωση που θέλουμε να προσθέσουμε επιπλέον θύρες πέρα από αυτές που διαθέτει η μητρική, ή να αντικαταστήσουμε μια κατεστραμμένη θύρα.

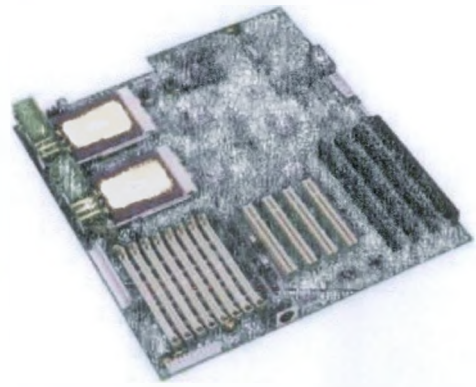
Η επικοινωνία της ΚΜΕ με το προσαρμοστικό κύκλωμα γίνεται μέσω διευθύνσεων θύρας (port address). Μέσω των διευθύνσεων αυτών η ΚΜΕ μπορεί:

1. Να δέχεται και να στέλνει δεδομένα μέσω της θύρας.
2. Να ελέγχει την ταχύτητα και γενικότερα τα χαρακτηριστικά της επικοινωνίας της θύρας.

Στο προσαρμοστικό κύκλωμα της θύρας διαθέτει επίσης μία διακοπή, με την οποία ειδοποιεί την ΚΜΕ, για παράδειγμα, ότι έχει έτοιμα δεδομένα, ή ότι υπάρχει κάποιο λάθος κατά την αποστολή των δεδομένων κ.ο.κ.

2.13. Η μητρική πλακέτα

Η μητρική πλακέτα (motherboard) είναι μια παραλληλόγραμμη πλακέτα διαστάσεων περίπου 22x25 cm, πάνω στην οποία είτε είναι τοποθετημένες είτε συνδέονται με τη βοήθεια καλωδίων όλες οι μονάδες του προσωπικού υπολογιστή. Στην ουσία είναι ένα σχετικά μεγάλο τυπωμένο κύκλωμα, πάνω στο οποίο είναι κολλημένα ολοκληρωμένα κυκλώματα που υλοποιούν τις διάφορες μονάδες του υπολογιστή. Επίσης, πάνω στη μητρική πλακέτα είναι τοποθετημένες οι υποδοχές επέκτασης (expansion slots) για τη σύνδεση άλλων περιφερειακών μονάδων.



Η αρχιτεκτονική της μητρικής πλακέτας έχει άμεση σχέση με το είδος του επεξεργαστή του προσωπικού υπολογιστή. Έτσι, για παράδειγμα, η μητρική πλακέτα πρέπει να έχει διαφορετικά κυκλώματα, όταν χρησιμοποιείται ένας επεξεργαστής της οικογένειας Pentium, και άλλα, όταν χρησιμοποιείται ένας επεξεργαστής της οικογένειας Pentium II. Επίσης, διαφορετική είναι η βάση πάνω στην οποία τοποθετείται ο επεξεργαστής καθώς και το είδος της μνήμης που χρησιμοποιείται. Επομένως, η επιλογή της μητρικής πλακέτας του προσωπικού υπολογιστή γίνεται έχοντας πάντα υπόψη τον επεξεργαστή που θα χρησιμοποιηθεί. Συνήθως οι μητρικές πλακέτες σχεδιάζονται, ώστε να μπορούν να τοποθετηθούν σε αυτές διάφοροι επεξεργαστές, οι οποίοι όμως πρέπει να έχουν παρόμοια αρχιτεκτονική. Έτσι, σε μια μητρική πλακέτα ενδεχομένως να μπορεί να τοποθετηθεί ένας από τους επεξεργαστές της οικογένειας Pentium II ή Celeron, αλλά δεν μπορεί να τοποθετηθεί ένας επεξεργαστής Pentium ή 80486, γιατί έχουν διαφορετική αρχιτεκτονική.

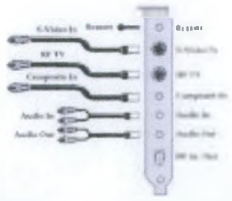
2.14. Το κουτί της κεντρικής μονάδας

Τα τμήματα της κεντρικής μονάδας ενός προσωπικού υπολογιστή (η μητρική πλακέτα, το τροφοδοτικό, οι σκληροί δίσκοι, οι οδηγοί δισκέτας και οι κάρτες επέκτασης) τοποθετούνται σε ειδικά διαμορφωμένους χώρους σ' ένα μεταλλικό κουτί.

Κατά καιρούς έχουν κυκλοφορήσει διάφοροι τύποι κουτιών με διαφορετικές ιδιότητες και σχεδιασμένα για διαφορετικές χρήσεις. Για παράδειγμα, υπάρχουν κουτιά που τοποθετούνται οριζόντια και άλλα που τοποθετούνται κάθετα, υπάρχουν κουτιά που χρησιμοποιούνται σε: βιομηχανικό περιβάλλον και άλλα που δε χρησιμοποιούν καθόλου βίδες για τη στήριξη της μητρικής πλακέτας και των περιφερειακών.

Ενότητα Β: Από τον κωδικοποιητή της πληροφορικής στην ολοκληρωμένη αρχιτεκτονική των υπολογιστικών συστημάτων

Κεφάλαιο 3: Κωδικοποίηση της πληροφορικής, θεωρήματα και πράξεις για τα διάφορα συστήματα κωδικοποίησης



3.1. Αριθμητικά συστήματα

3.1.1. Συστήματα Αρίθμησης

Ονομάζουμε σύστημα αρίθμησης τη μέθοδο παράστασης των αξιών που επιδέχονται μέτρηση, έτσι ώστε σε κάθε μια από αυτές να αντιστοιχεί μία και μοναδική "εικόνα", την οποία ονομάζουμε παράσταση της αριθμητικής αξίας στο σύστημα αρίθμησης. Τα συστήματα αρίθμησης διακρίνονται σε θεσιακά και μη θεσιακά.

Ένα από τα μεγαλύτερα επιτεύγματα του πολιτισμού μας θεωρείται η επινόηση των θεσιακών συστημάτων αρίθμησης. Στα θεσιακά συστήματα αρίθμησης (Positional Number Systems) μια αριθμητική αξία παριστάνεται σαν μια αλυσίδα από ψηφία έτσι ώστε η θέση κάθε ψηφίου μέσα στην αλυσίδα να έχει το δικό της αριθμητικό βάρος.

Δυαδικό, Οκταδικό και δεκαεξαδικό σύστημα αρίθμησης

Το δυαδικό σύστημα έχει βάση R=2 και κάθε αριθμός σε αυτό μπορεί να παρασταθεί σαν μια αλυσίδα από ψηφία τα οποία ανήκουν στο σύνολο [0,1]. Το δυαδικό είναι το σύστημα με τη μικρότερη βάση που μπορεί να μας φανεί χρήσιμο. Πράγματι, συστήματα με βάση 1 και 0 μόνο θεωρητικά μπορούν να υπάρξουν.

Η παράσταση 1011_2 είναι η παράσταση ενός αριθμού στο δυαδικό σύστημα. Ο δείκτης 2 είναι απαραίτητος όταν δεν είναι σαφές ότι δουλεύουμε σ' ένα μόνο σύστημα. Και αυτό γιατί οι παραστάσεις 1011_{10} και 1011_2, αν έχουν την ίδια εικόνα-παράσταση, παριστάνουν εντελώς διαφορετικές αξίες.

Το οκταδικό σύστημα έχει σαν βάση το 8 και τα ψηφία του ανήκουν στο σύνολο [0,1,2,3,4,5,6,7], ενώ το δεκαεξαδικό έχει βάση το 16 και τα ψηφία του ανήκουν στο σύνολο [0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F]. Η παράσταση 623_8 είναι η παράσταση ενός αριθμού στο οκταδικό σύστημα ενώ η $6AF3.6AB_{16}$ είναι η παράσταση ενός αριθμού στο δεκαεξαδικό.

Το δυαδικό χρησιμοποιείται για την άμεση παράσταση των αριθμών μέσα στον υπολογιστή. Τα άλλα δύο συστήματα (οκταδικό και δεκαεξαδικό) χρησιμοποιούνται για τη συνοπτική παράσταση των δυαδικών ποσοτήτων εκτός του υπολογιστή. Και αυτό γιατί έχουν βάσεις που είναι δυνάμεις του δύο. Αυτή η ιδιότητα διευκολύνει, όπως θα δούμε παρακάτω, την μετατροπή της παράστασης ενός αριθμού από το ένα σύστημα στο άλλο.

Παράδειγμα παρουσιάζουμε τους αριθμούς από το 1 ως 16 του δεκαδικού συστήματος και τις αντίστοιχες παραστάσεις τους στο δυαδικό, οκταδικό και δεκαεξαδικό. Παρατηρούμε ότι τα ψηφία του οκταδικού συστήματος χρειάζονται 3 bit για να παρασταθούν στο δυαδικό σύστημα ενώ τα ψηφία του δεκαεξαδικού χρειάζονται 4. Κρατήστε στο μυαλό σας την αντιστοιχία του οκταδικού με το 3 και του δεκαεξαδικού με το 4. Θα χρειαστεί στις μετατροπές των παραστάσεων από το ένα σύστημα στο άλλο. Οι δύο τελευταίες στήλες του πίνακα 1.2 μας δείχνουν τη δυνατότητα παράστασης αριθμών στα 3 και 4 bits αντίστοιχα. Παρατηρήστε ότι το 8 δεν μπορεί να παρασταθεί στα 3 bits. Αυτό σημαίνει ότι, ένας υπολογιστή με μήκος θέσης 3 bits δεν μπορεί να δεχθεί σαν δεδομένο τον αριθμό 8 γιατί απλούστατα δεν μπορεί να τον αποθηκεύσει.

Δεκαδικό	Δυαδικό	Οκταδικό	Δεκαεξαδικό	3 Bits	4Bits
0	0	0	0	000	0000
1	1	1	1	001	0001
2	10	2	2	010	0010
3	11	3	3	011	0011
4	100	4	4	100	

0100				
5	101	.5	5	101
0101				
6	110	.6	6	110
0110				
7	111	7	7	111
0111				
8	1000	10	8	-
1000				
9	1001	11	9	-
1001				
10	1010	12	A	-
1010				
11	1011	13	B	-
1011				
12	1100	14	C	-
1100				
13	1101	15	D	-
1101				
14	1110	16	E	-
1110				
15	1111	17	F	-
1111				
16	10000	20	10	-

Αντιστοιχίες συστημάτων

3.2. Κωδικοποίηση της πληροφορικής

3.2.1. Πληροφορία

Στην καθημερινή μας ζωή χρησιμοποιούμε σχεδόν αποκλειστικά το δεκαδικό σύστημα αρίθμησης. Είναι λοιπόν φυσικό να έχουμε ταυτίσει την αξία (value) μιας ποσότητας, με τη παράσταση της στο σύστημα αυτό. Στη πραγματικότητα δεν είναι έτσι. Οι αξίες μπορούν να παρασταθούν με πολλούς τρόπους. Η αξία παραμένει η ίδια, εκείνο που αλλάζει κάθε φορά είναι το σύστημα αρίθμησης. Η μέθοδος παράστασης των αριθμητικών ποσοτήτων, ονομάζεται αριθμητικό σύστημα αρίθμησης (Number System).

Στους υπολογιστές, η χρήση του δεκαδικού συστήματος θα δημιουργούσε προβλήματα κόστους και πολυπλοκότητας, γιατί θα έπρεπε να κατασκευαστούν στοιχειώδης διακόπτες με δέκα διαφορετικές καταστάσεις. Θα έπρεπε για παράδειγμα, να κατασκευαστούν τρανζίστορ με δέκα

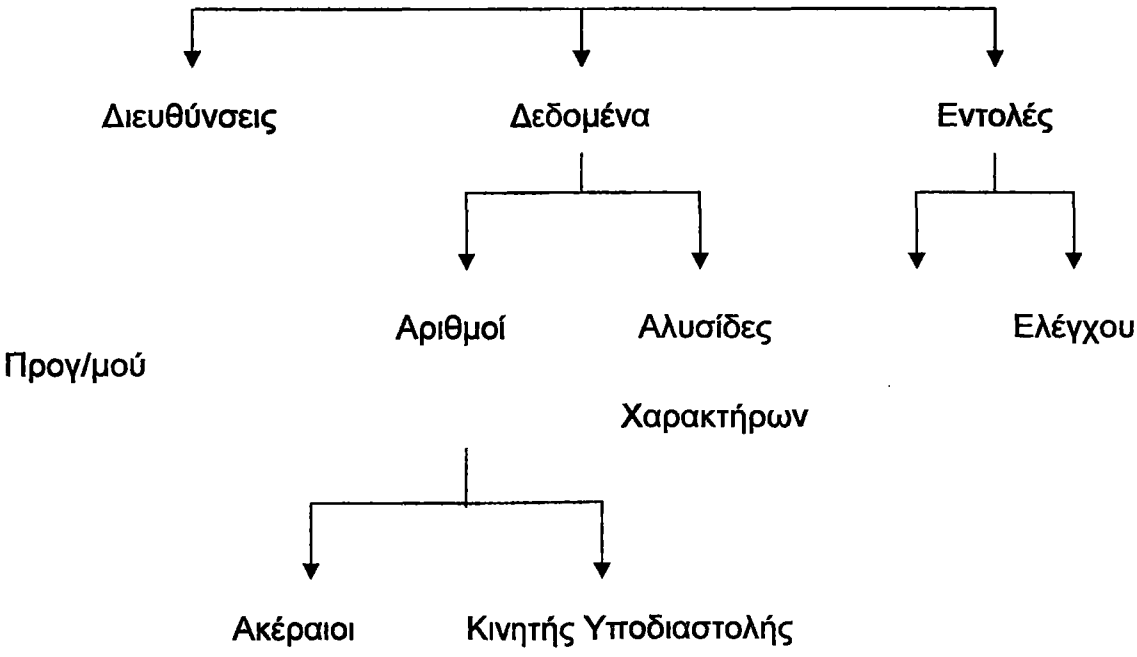
διαφορετικές καταστάσεις αντί για δύο. Για τον λόγο αυτό προτιμήθηκε το δυαδικό σύστημα, όπου η κατασκευή μηχανισμών με δύο καταστάσεις (on, off) ήταν πιο εύκολη και φθηνή.

Για παράδειγμα,

Από έναν αγωγό περνά ή δεν περνά ρεύμα , ένας πυκνωτής είναι φορτισμένος ή δεν είναι, ένας μαγνητικός δακτύλιος είναι μαγνητισμένος προς την μια ή την άλλη διεύθυνση, μια μονάδα είναι απασχολημένη ή δεν είναι, μία λυχνία ή ένα τρανζίστορ επιτρέπουν ή όχι την διέλευση ρεύματος κ.λ.π.

Όλες οι πληροφορίες που συνυπάρχουν μέσα σε ένα υπολογιστή αναλύονται σε μια ακολουθία από δυαδικά ψηφία. Οι διαφορετικοί τύποι πληροφορίας παρουσιάζονται επόμενο σχήμα και αφορούν εντολές, δεδομένα και διευθύνσεις.

Αν και η έννοια της πληροφορίας συνδέεται (λανθασμένα) με τα δεδομένα που επεξεργάζεται ένας υπολογιστής, πληροφορία είναι και οι εντολές που δέχεται ο υπολογιστής από το περιβάλλον. Με τις εντολές αυτές ο υπολογιστής ενεργεί πάνω στα δεδομένα και παράγει νέα. Οι εντολές είναι και τα σήματα ελέγχου που εκπορεύονται από την μονάδα ελέγχου προς τις υπόλοιπες μονάδες. Άλλη μια μορφή πληροφορίας είναι οι διευθύνσεις των δεδομένων και των εντολών μέσα στον υπολογιστή. Σε κάθε δεδομένο ή εντολή αντιστοιχεί πάντα και μια διεύθυνση. Αυτό είναι φυσικό, και έτσι συμβαίνει και στη καθημερινή μας ζωή. Μια γνωριμία σε ένα πάρτι είναι άχρηστη αν δεν συνδέεται και με ένα αριθμό τηλεφώνου.



Τύποι Πληροφορίας

Υπάρχουν δύο τύποι δεδομένων. Οι αριθμοί κάθε τύπου (ακέραιοι, πραγματικοί ή κινητής υποδιαστολής, κ.λπ.) και οι αλυσίδες χαρακτήρων. Ονομάζουμε αλυσίδα χαρακτήρων (String) μια συνεχή ακολουθία από απλούς χαρακτήρες (Characters)

Οι βασικές μονάδες πληροφορίας δεδομένων, όπως άλλωστε και κάθε άλλο είδος πληροφορίας, πρέπει να μεταφερθούν στο δυαδικό σύστημα

αρίθμησης, αφού είναι το μοναδικό σύστημα αρίθμησης που "καταλαβαίνει" ο υπολογιστής. Η μεταφορά αυτή γίνεται κατά την φάση της εισόδου των δεδομένων. Πράγματι, τα δεδομένα πληκτρολογούνται στο δεκαδικό σύστημα αρίθμησης μέσο του πληκτρολογίου και πριν ο υπολογιστής τα επεξεργαστεί, το λειτουργικό σύστημα τα μεταφέρει στο δυαδικό σύστημα αρίθμησης. Το ίδιο γίνεται, αλλά με την αντίθετη λογική, κατά τη φάση της εξόδου των δεδομένων που παράχθηκαν από τη "μηχανή". Αν σας παραξενεύει ο όρος μηχανή που δίνουμε στον υπολογιστή, σας πληροφορώ ότι είναι πολύ διαδεδομένος στην αργκό των ανθρώπων των υπολογιστών. Πιστεύω ότι είναι μια ασυναίσθητη ενέργεια υποβιβασμού του υπολογιστή, η οποία πρέπει να πηγάζει από την καταπίεση που νοιώθουμε από αυτό το θαύμα της ανθρώπινης τεχνολογίας.

Είναι γνωστό από προηγουμένως ότι η πιο μικρή πληροφορία που μπορεί να παρασταθεί ηλεκτρονικά, μαγνητικά ή ηλεκτρομηχανικά είναι το bit (binary digit). Κάθε μορφής πληροφορία παριστάνεται εσωτερικά σαν μια αλυσίδα από διαδοχικά bits. Ένα πλήθος από bits συγκροτούν μεγαλύτερες μονάδες πληροφορίας, πιο περιεκτικές σε αξία. Μια τέτοια μονάδα πληροφορίας, από τις πιο βασικές θα έλεγα, είναι ο χαρακτήρας (αν ο χαρακτήρας σχηματίζεται από 8 bits , ονομάζεται byte). Το μήκος ενός χαρακτήρα σε bits εξαρτάται βασικά από την αρχιτεκτονική του υπολογιστή, αλλά η πιο συνηθισμένη περίπτωση είναι το οκτώ (άλλωστε byte σημαίνει οκτάδα). Ένα πλήθος από bytes συγκροτούν μια αμέσως επόμενη μονάδα, η οποία ονομάζεται λέξη (Word). Και εδώ το πλήθος των Bytes ανά λέξη ποικίλουν από αρχιτεκτονική σε αρχιτεκτονική (επόμενος πίνακας).

Οικογένεια	bits
VAX	16
IBM S/370	32
CDC Cyber	60
INTEL 8080	8
INTEL 80XX6	16
INTEL PENTIUM	32
MC6800	8
	MC680XX
16/32	
BERCLEY Risc II	32
BURROUGHS B5X00	48

Μήκος λέξης σε bits

Επειδή οι εντολές, τα δεδομένα και οι διευθύνσεις αναφέρονται στη μνήμη του υπολογιστή θα κάνουμε μια μικρή αναφορά σε αυτήν. Η κεντρική μνήμη δεν είναι τίποτε άλλο παρά μια αποθήκη που αποτελείται από ομοιόμορφους χώρους. Κάθε χώρος ονομάζεται θέση (Location) και στη θέση αυτή αντιστοιχεί μια διεύθυνση (Address). Κάθε θέση σχηματίζεται από ένα πλήθος από bits τα οποία υποδιαιρούνται σε bytes. Ένα πράγμα που πρέπει να συγκρατήσει κανείς είναι ότι κάθε θέση μνήμης μπορεί να περιέχει είτε ένα

δεδομένο, είτε μία εντολή. Επίσης σε κάθε θέση μνήμης αντιστοιχούν δύο ποσότητες, το περιεχόμενο και η διεύθυνση. Αν κάποιος "κοιτάξει" σε μια δεδομένη στιγμή το περιεχόμενο μια τυχαίας διεύθυνσης θα "δει" μια σειρά από "0" και "1". Σε καμιά περίπτωση δεν θα μπορεί διαγνώσει αν πρόκειται για δεδομένο ή εντολή. Αυτό μπορεί να το κάνει αν εφοδιαστεί και με άλλες πληροφορίες.

Πέρα όμως από το δυαδικό σύστημα χρησιμοποιούνται, αλλά σε βοηθητικό ρόλο, το οκταδικό και το δεκαεξαδικό σύστημα. Μοιραία λοιπόν θα συγκεντρώσουμε τη προσοχή μας σ' αυτά τα τρία συστήματα. Η μελέτη αυτών των συστημάτων θα μας βοηθήσει να φτάσουμε στον στόχο αυτού του κεφαλαίου, που είναι η κατανόηση του τρόπου με τον οποίο παριστάνονται στους ψηφιακούς υπολογιστές οι αριθμοί και οι αλυσίδες χαρακτήρων (αλφαριθμητικές ποσότητες). Στα επόμενα κεφάλαια θα μάθουμε πως παριστάνονται εσωτερικά οι εντολές και οι διευθύνσεις.

3.2.2. Μετατροπές παραστάσεων

Όταν δουλεύει κανείς με διαφορετικά αριθμητικά συστήματα χρειάζεται να γνωρίζει τη μέθοδο μετάβασης από το ένα σύστημα στο άλλο. Με άλλα λόγια να μπορεί να μετατρέπει παραστάσεις. Υπάρχουν δύο μέθοδοι μετατροπής. Οι αριθμητικές και οι μη αριθμητικές. Οι πρώτες πραγματοποιούνται με τη βοήθεια αριθμητικών πράξεων και εφαρμόζονται σ' όλες τις περιπτώσεις, ενώ οι δεύτερες εφαρμόζονται σε ειδικές περιπτώσεις και βασίζονται σε κάποιους εριστικούς κανόνες.

Αριθμητικές μέθοδοι μετατροπής

Αν Q είναι η βάση του συστήματος μετάβασης και R ή βάση του συστήματος αναχώρησης ισχύει η σχέση :

$$(X_R)_Q = \sum_{N-1}^{0} X_i R^i + \sum_{-1}^{-M} X_i R^i$$

όπου οι πράξεις στο δεύτερο μέλος της γίνονται στο σύστημα μετάβασης Q, αφού προηγουμένως τα X_i και το R έχουν μεταφερθεί από το σύστημα R στο Q.

Παραδείγματα :

α) Q = 10, R = 2

$(110.111_2)_{10} = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 6.875_{10}$

β) Q = 10, R = 8

$(3507_8)_{10} = 3 \times 8^3 + 5 \times 8^2 + 0 \times 8^1 + 7 \times 8^0 = 1863_{10}$

γ) Q = 10, R = 6 $(5.325_6)_{10} = 5 \times 6^0 + 3 \times 6^{-1} + 2 \times 6^{-2} + 5 \times 6^{-3} = 15.578703703..._{10}$

Παρατηρούμε ότι ένας αριθμός μπορεί σε ένα σύστημα να έχει παράσταση με πεπερασμένο πλήθος ψηφίων και σ' ένα άλλο να μην έχει.

δ) Q = 10, R = 16

$(3AB8.C_{16})_{10} = 3 \times 16^3 + A \times 16^2 + B \times 16^1 + 8 \times 16^0 + C \times 16^{-1} =$

$= 3 \times 16^3 + 10 \times 16^2 + 11 \times 16^1 + 8 \times 16^0 + 12 \times 16^{-1} =$

$= 12288 + 2560 + 176 + 8 + 0.75 = 14856.75_{10}$

Στο δεύτερο βήμα της μετατροπής μετατρέψαμε τα ψηφία του δεκαεξαδικού αριθμού στο σύστημα μετάβασης, δηλαδή στο δεκαδικό.

ε) Q = 2, R = 10

$(31_{10})_2 = 3_{10} \times 10^1 + 1_{10} \times 10^0$ = (ανάπτυγμα τύπου 2)

$= (11_2) \times (1010_2)^1 + 1_2 \times 1_2$ = (μετατροπή της βάσης 10 στο δυαδικό)

$= 11110_2 + 1_2 = 11111_2$ (πρόσθεση στο δυαδικό)

Επαναλαμβάνουμε ότι οι πράξεις έγιναν στο σύστημα μετάβασης, δηλαδή στο δυαδικό. Επίσης τα ψηφία 3 και 1 του δεκαδικού αριθμού 31_{10} μεταφέρθηκαν στο δυαδικό σύστημα.

Ας υποθέσουμε ότι πρέπει να πραγματοποιήσουμε την μετατροπή $(2345.89_{10})_{16}$. Θα πρέπει σύμφωνα με τον κανόνα, μετά την εφαρμογή του τύπου, να κάνουμε τις πράξεις στο δεκαεξαδικό σύστημα. Τι γίνεται όμως αν δεν ξέρουμε να κάνουμε πράξεις στο σύστημα αυτό; Ήδη στο παράδειγμα ε) υποθέσαμε ότι γνωρίζουμε να κάνουμε πράξεις στο δυαδικό σύστημα, αν και αυτό δεν είναι αλήθεια. Υπάρχει λοιπόν πρόβλημα όταν το σύστημα μετάβασης έχει άγνωστη σε μας αριθμητική

Θα παρουσιάσουμε τώρα μια μέθοδο, όπου θα μπορούμε να μεταβαίνουμε από ένα σύστημα με γνωστή αριθμητική σ' ένα σύστημα με άγνωστη αριθμητική χρησιμοποιώντας την αριθμητική του συστήματος αναχώρησης. Θα αντιμετωπίσουμε το πρόβλημα χωρίζοντας τη μέθοδο σε δύο υπομεθόδους. Η πρώτη, ασχολείται με τη μετατροπή του ακεραίου μέρους της παράστασης του αριθμού και η δεύτερη με τη μετατροπή του κλασματικού μέρους.

Αλγόριθμος διαδοχικών διαιρέσεων

B0: Αρχικός διαιρετέος = Παράσταση στο σύστημα με βάση R
B1: Διαιρούμε (διαίρεση στο R) τον διαιρετέο με την παράσταση της βάσης Q στο σύστημα R .
B2: Μετατρέπουμε το υπόλοιπο από το σύστημα R στο Q και το σημειώνουμε.

Β3: ΑΝ το πηλίκο είναι μηδέν ΤΟΤΕ πηγαίνουμε στο βήμα Β4
ΔΙΑΦΟΡΕΤΙΚΑ τοποθετούμε το πηλίκο στη θέση του διαιρέτη και πηγαίνουμε στο βήμα Β1.

Β4: Σχηματίζουμε τη νέα παράσταση στο Q, με LSD το πρώτο από τα υπόλοιπα των διαδοχικών διαιρέσεων που σημειώσαμε και MSD το τελευταίο.

Παραδείγματα :

α) Μετατροπή του 35_{10} στο δυαδικό σύστημα.

Διαίρεση	πηλίκο	υπόλοιπο	
$35_{10} : 2$	17	1	
$17_{10} : 2$	8	1	
$8_{10} : 2$	4	0	
$4_{10} : 2$	2	0	
$2_{10} : 2$	1	0	
$1_{10} : 2$	0	1	MSB 100011_2 LSB

β) Μετατροπή του 353_{10} στο δυαδικό.

Διαίρεση	πηλίκο	υπόλοιπο	
$353_{10} : 2$	176	1	
$176_{10} : 2$	88	0	
$88_{10} : 2$	44	0	
$44_{10} : 2$	22	0	
$22_{10} : 2$	11	0	
$11_{10} : 2$	5	1	
$5_{10} : 2$	2	1	
$2_{10} : 2$	1	0	
$1_{10} : 2$	0	1	MSB 101100001_2 LSB

γ) Μετατροπή του 1863_{10} στο οκταδικό.

Διαίρεση	πηλίκο	Υπόλοιπο	
$1863_{10} : 8$	232	7	
$232_{10} : 8$	29	0	
$29_{10} : 8$	3	5	
$3_{10} : 8$	0	3	MSD 3507_8 LSD

δ) Μετατροπή του 2567_{10} στο δεκαεξαδικό.

Διαίρεση	πηλίκο	Υπόλοιπο10	Υπόλοιπο16	
$2567_{10} : 16$	160	7_{10}	7_{16}	
$160_{10} : 16$	10	0_{10}	0_{16}	
$10_{10} : 16$	0	10_{10}	A_{16}	MSD $A07_{16}$ LSD

Παρατηρούμε ότι τα υπόλοιπα πρέπει να μεταφερθούν στο δεκαεξαδικό σύστημα.

Η μέθοδος που αφορά τη μετατροπή του κλασματικού μέρους μιας παράστασης ενός αριθμού από το R στο Q είναι γνωστή σαν μέθοδος

διαδοχικών πολ/μών με τη βάση του συστήματος μετάβασης και με αριθμητική του συστήματος αναχώρησης και πραγματοποιείται από τον παρακάτω αλγόριθμο.

Αλγόριθμος διαδοχικών πολλαπλασιασμών

B0: Αρχικός πολλαπλασιαστέος = Παράσταση στο R

B1: Πολλαπλασιάζουμε (Πολ/μός στο R) τον πολ/στέο με το Q .

B2: Μετατρέπουμε το ακέραιο μέρος του γινομένου από το R στο Q και το σημειώνουμε

B3: ΑΝ το κλασματικό μέρος του γινομένου είναι μηδέν ΤΟΤΕ πηγαίνουμε στο βήμα B4 ΔΙΑΦΟΡΕΤΙΚΑ τοποθετούμε το κλασματικό μέρος του γινομένου στη θέση του πολ/στέου και πηγαίνουμε στο βήμα B1

B4: Σχηματίζουμε τη νέα παράσταση στο Q, με MSD μετά το βασικό σημείο το, πρώτο από ακέραια μέρη των διαδοχικών πολ/μών και LSD το τελευταίο.

Παραδείγματα :

α) Μετατροπή του $.65625_{10}$ στο δυαδικό.

.65625	.3125	.625	.25	.5
x 2	x 2	x 2	x 2	x 2
1.3125	0.6250	1.250	0.5	1.0
↓	↓	↓	↓	↓
1	0	1	0	1

Άρα $0.65625_{10} = 0.10101_2$

Υπάρχει περίπτωση οι συνεχείς πολ/μοί να μη δώσουν ποτέ γινόμενο με μηδενικό κλασματικό μέρος. Τότε κατά τη κρίση μας σταματάμε του πολ/μούς έχοντας υπόψη ότι όσους περισσότερους πολ/μούς εκτελέσουμε τόσο μεγαλύτερη ακρίβεια επιτυγχάνουμε.

β) Μετατροπή του 0.1_{10} στο δυαδικό σύστημα.

0.1	0.2	0.4	0.8	0.6	0.2	0.4	0.8	0.6	0.2
x 2	x 2	x 2	x 2	x 2	x 2	x 2	x 2	x 2	x 2
0.2	0.4	0.8	1.6	1.2	0.4	0.8	1.6	1.2	0.4
0	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0 ...**

Άρα $0.1_{10} = (0.0001100110...)2$

Πολλές φορές λοιπόν η μεταφορά από ένα σύστημα στο άλλο δημιουργεί παραστάσεις που δεν είναι ισοδύναμες. Πράγματι, αν για παράδειγμα σταματήσουμε μετά από 10 πολ/μούς όπως κάναμε στη περίπτωση β) του παραδείγματος και δεχθούμε σαν απάντηση τον 0.0001100112, τότε λέμε ότι έχουμε μια μετατροπή κατά προσέγγιση. Αν ακολουθήσουμε την αντίστροφη πορεία, δηλαδή να μετατρέψουμε τον 0.0001100112 στο δεκαδικό σύστημα θα έχουμε:

$$0.000110011_2 = 0 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5} + 0 \times 2^{-6} +$$
$$+ 0 \times 2^{-7} + 1 \times 2^{-8} + 1 \times 2^{-9} = 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} =$$
$$= 1/16 + 1/32 + 1/256 + 1/512 = 0.099509375_{10}$$

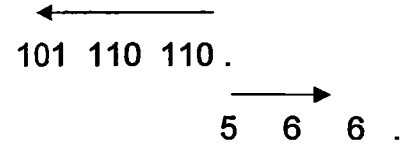
Παρατηρούμε λοιπόν ότι η μετατροπή από το ένα σύστημα στο άλλο προκάλεσε απώλεια στην ακρίβεια της αρχικής αριθμητικής ποσότητας. Ουσιαστικά δημιουργήθηκε μια νέα ποσότητα "πολύ" κοντά στην αρχική. Στην παρατήρηση αυτή στηρίζεται το γεγονός ότι οι πράξεις με Η/Υ έχουν πεπερασμένη ακρίβεια η οποία εξαρτάται από το μέγεθος σε bits μιας θέσης μνήμης ή των καταχωρητών εργασίας που γίνονται οι πράξεις.

Μη αριθμητικές μέθοδοι μετατροπής

Οι μέθοδοι αυτές αφορούν μετατροπές παραστάσεων αριθμών από το δυαδικό, στο οκταδικό, ή δεκαεξαδικό και αντίστροφα και στηρίζονται στη σχέση που υπάρχει μεταξύ των βάσεων των τριών συστημάτων, όπου 8 = 23 και 16=24.

Μετατροπή από το δυαδικό στο οκταδικό σύστημα. Χωρίζουμε τη δυαδική παράσταση του αριθμού σε ομάδες των 3 bits. Αν ο αριθμός είναι ακέραιος, αρχίζουμε την ομαδοποίηση από τα δεξιά προς τα αριστερά. Αν ο αριθμός έχει ακέραιο και κλασματικό μέρος, η ομαδοποίηση γίνεται προς δύο κατευθύνσεις. Η μία από το δυαδικό σημείο και προς τα δεξιά και η άλλη από το δυαδικό σημείο και προς τα αριστερά. Μετά την ομαδοποίηση μετατρέπουμε τις τριάδες σε οκταδικά ψηφία. **Παραδείγματα:**

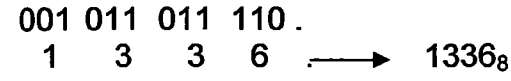
α) $101110110_2 = (;)_8$



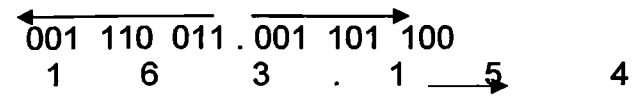
566_8

Αν τα ψηφία του δυαδικού αριθμού δεν είναι αρκετά για να συμπληρώσουμε ακέραιο αριθμό από τριάδες, τότε συμπληρώνουμε την ελλιπή τριάδα με μη σημαντικά μηδενικά.

β) $1011011110_2 = (;)_8$



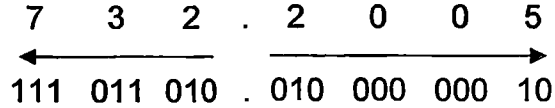
γ) $1110011.0011011_2 = (;)_8$



163.154_8

Μετατροπή από το οκταδικό στο δυαδικό: Αν έχουμε έναν αριθμό με παράσταση στο οκταδικό σύστημα αναλύουμε ένα προς ένα τα ψηφία του σε ομάδες των τριών bit, διατηρώντας τη θέση του βασικού σημείου. Τα ψηφία του δυαδικού συνενώνονται και σχηματίζουν την παράσταση του αριθμού στο δυαδικό. Τα μη σημαντικά ψηφία (μηδενικά) απομακρύνονται.

Παράδειγμα : $732.2005_8 = (;)_2$



Άρα $732.2005_8 = 111011010.010000000101_2$

Μετατροπή από το δυαδικό στο δεκαεξαδικό: Ακολουθούμε την ίδια μέθοδο με αυτήν της μετατροπής από το δυαδικό στο οκταδικό με μια μόνο διαφορά. Η ομαδοποίηση γίνεται σε ομάδες των τεσσάρων Bits αντί των τριών.

Παραδείγματα :

α) $101110110_2 = (;)_{16}$

0001 0111 0110.

1 7 6 ⟶ 176_8

Παρατηρούμε και πάλι, ότι συμπληρώνουμε τη αρχική παράσταση με μηδενικά από δεξιά (για το κλασματικό μέρος) και από αριστερά (για το ακέραιο μέρος) για να σχηματίσουμε πλήρης τετράδες.

β) $111011.101101_{12} = (;)_{16}$

0111 1011 . 1011 0110

7 B . B 6 ⟶ $7BB6_{16}$

Μετατροπή από το δεκαεξαδικό στο δυαδικό. Η μέθοδος λειτουργεί όπως η αντίστοιχη του οκταδικού, με μόνη τη διαφορά που επισημάναμε πιο πριν.

Παράδειγμα :

$2BED.A1_{16} = (;)^2$

2 B E D . A 1

0010 1011 1110 1101 . 1010 0001

Άρα $2BED.A1_{16} = 101011111011.10100001_2$

Παρατηρούμε ότι απομακρύνουμε τα μηδενικά που βρίσκονται αριστερά ή δεξιά και δεν επηρεάζουν την αριθμητική τιμή της παράστασης του αριθμού.

Μετατροπή από το δεκαεξαδικό στο οκταδικό και αντίστροφα

Οι κλασσικές μέθοδοι μετατροπής είναι επίπονες, γιατί δεν είμαστε εξοικειωμένοι με τις αριθμητικές των δύο συστημάτων. Η ενδιάμεση μέθοδος που θα δείξουμε τώρα έχει δυο βήματα. Στο πρώτο βήμα γίνεται η μετατροπή της δεκαεξαδικής παράστασης τού αριθμού στο δυαδικό σύστημα με την μέθοδο των τετράδων. Στο δεύτερο βήμα γίνεται η μετατροπή της δυαδικής

παράστασης, που δημιουργήθηκε από το πρώτο βήμα, στο οκταδικό με τη μέθοδο των τριάδων. Η μέθοδος λειτουργεί και αντίστροφα.

Παράδειγμα : $2BED_{16} = (;)_8$

2 B E D

0010 1011 1011 1101

10101110111101_2

010 101 110 111 101

2 5 6 7 5 25675_8

Οι παραπάνω μέθοδοι είναι ελκυστικές γιατί δεν απαιτούν υπολογισμούς.

3.2.3. Πρόσθεση και αφαίρεση μη προσημασμένων ακεραίων

Οι βασικές πράξεις σ' ένα σύστημα αρίθμησης είναι η πρόσθεση και αφαίρεση. Οι άλλες πράξεις προκύπτουν από συνδυασμό των βασικών πράξεων. Η πρόσθεση παραστάσεων αριθμών στα θεσιακά συστήματα αρίθμησης, γίνεται πάντα με την ίδια μέθοδο και είναι ανεξάρτητη της βάσης του συστήματος. Εκείνο που αλλάζει από σύστημα σε σύστημα είναι ο πίνακας πρόσθεσης του συστήματος. Σε κάθε σύστημα αρίθμησης αντιστοιχεί και ένας τέτοιος πίνακας. Επίσης σε κάθε σύστημα αρίθμησης αντιστοιχεί και ένας πίνακας αφαίρεσης. Ο πίνακας μιας βασικής πράξης περιέχει όλους τους συνδυασμούς των ψηφίων του συστήματος ανά δύο.

Πρόσθεση

Ο πίνακας της πρόσθεσης ορίζει τη μονοψήφια πρόσθεση. Στο δεκαδικό σύστημα αντιστοιχεί ο παρακάτω πίνακας .

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

Πίνακας πρόσθεσης δεκαδικού συστήματος

Η μονάδα, μπροστά από το LSD, όπου υπάρχει (με έντονα γράμματα), ονομάζεται κρατούμενο (carry). Σε κάθε ζεύγος ψηφίων του δεκαδικού συστήματος αντιστοιχεί ένα ψηφίο του συστήματος και ένα κρατούμενο (τα μηδενικά κρατούμενα παραλείπονται). Το κρατούμενο είναι το μηδέν ή το ένα.

Η πράξη της πρόσθεσης μεταξύ δύο παραστάσεων με N ψηφία γίνεται ως εξής: Γίνονται τόσες μονοψήφιες προσθέσεις όσα και τα ψηφία των προσθετέων. Κάθε μονοψήφια πρόσθεση δέχεται ένα κρατούμενο (κρατούμενο εισόδου) από τη προηγουμένη και παράγει ένα κρατούμενο (εξόδου) για την επομένη. Το κρατούμενο εισόδου προστίθεται στο αποτέλεσμα της μονοψήφιας πρόσθεσης. Η πρόσθεση των LSD των δύο προσθετέων έχει αρχικό κρατούμενο 0. Το κρατούμενο της τελευταίας πρόσθεσης είναι και το MSD του αποτελέσματος. Ο παραπάνω κανόνας δεν εξαρτάται από το σύστημα αρίθμησης.

Η ίδια λογική ισχύει και για την δυαδική πρόσθεση της οποίας ο πίνακας παρουσιάζεται παρακάτω.

+	0	1
0	0	1
1	1	10

Πίνακας Πρόσθεσης δυαδικού συστήματος

Η μονοψήφια δυαδική πρόσθεση πραγματοποιείται από τον full-adder, όπως περιγράψαμε στην παράγραφο περί συνδυαστικών κυκλωμάτων.

Ο μηχανισμός πρόσθεσης παρουσιάζεται στο επόμενο σχήμα.

Ο πίνακας αλήθειας του ακολουθιακού κυκλώματος μηχανισμού της μονοψήφιας πρόσθεσης δίνεται στον παρακάτω πίνακα

C-in	X	Y	S = (X+Y+C-in)	C-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0

0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Πίνακας αλήθειας μονοψήφιας πρόσθεσης

Παράδειγμα :

Κρατούμενα 1 1 1 1 1 0 1 **0** ← αρχικό κρατούμενο

Προσθετέος 1 0 1 1 1 0 1
Προσθετέος + 0 1 0 0 1 0 1

Άθροισμα 1 0 0 0 0 0 1 0

Αφαίρεση

Όπως αναφέραμε και νωρίτερα, ο μηχανισμός της αφαίρεσης δεν εξαρτάται από το σύστημα αρίθμησης. Εκείνο που αλλάζει είναι ο πίνακας της αφαίρεσης. Η μονάδα εμπρός από το LSB (έντονα γράμματα) ονομάζεται δανεικό ψηφίο (Borrow). Η πράξη της αφαίρεσης πραγματοποιείται με την υπόθεση ότι ο μειωτέος είναι μεγαλύτερος από τον αφαιρετέο και γίνεται ως εξής:

-	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	19	18	17	16	15	14	13	12	11
1	1	0	19	18	17	16	15	14	13	12
2	2	1	0	19	18	17	16	15	14	13
3	3	2	3	0	19	18	17	16	15	14
4	4	3	2	1	0	19	18	17	16	15
5	5	4	3	2	1	0	19	18	17	16
6	6	5	4	3	2	1	0	19	18	17
7	7	6	5	4	3	2	1	0	19	18
8	8	7	6	5	4	3	2	1	0	19
9	9	8	7	6	5	4	3	2	1	0

Πίνακας αφαίρεσης δεκαδικού συστήματος

Η διαδικασία αποτελείται από ένα πλήθος από μονοψήφιες αφαιρέσεις όσες και το πλήθος των ψηφίων του μειωτέου (ή αφαιρετέου). Κάθε μια από τις μονοψήφιες αφαιρέσεις δέχεται ένα δανεικό ψηφίο εισόδου B-in από την προηγουμένη και παράγει ένα δανεικό ψηφίο εξόδου B-out για τη επομένη. Το δανεικό ψηφίο εισόδου μιας αφαίρεσης προστίθεται στον αφαιρετέο πριν γίνει η αφαίρεση. Η διαδικασία των διαδοχικών αφαιρέσεων αρχίζει από το LSB

προς το MSB με δανεικό ψηφίο της πρώτης αφαίρεσης το μηδέν.

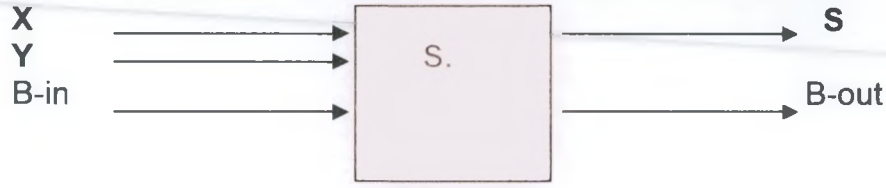
Στο δυαδικό σύστημα ο πίνακας της αφαίρεσης δίνεται στον παρακάτω πίνακα.

Αφαιρεταιος

-	0	1
0	0	**11**
1	1	0

Πίνακας αφαίρεσης στο δυαδικό σύστημα

Ο μηχανισμός αφαίρεσης παρουσιάζεται στο επόμενο σχήμα.



Μηχανισμός Αφαίρεσης

Ο πίνακας αλήθειας του ακολουθιακού κυκλώματος μηχανισμού της μονοψήφιας αφαίρεσης δίνεται στον παρακάτω πίνακα

bin	X	Y	S = X-(Y+B-in)	b-out
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Πίνακας αλήθειας μονοψήφιας αφαίρεσης

Σημειώνουμε ότι αν ο υπολογισμός της παράστασης X-(Y+B-in) παρουσιάζει δυσκολίες τότε χρησιμοποιούμε την ισοδύναμη X-Y-B-in. Προτείνουμε να κατασκευάσετε τους πίνακες αλήθειας της πρόσθεσης και της αφαίρεσης.

Παράδειγμα :

```
Δανεικά          0  1  0  0  0  1  1  0   ←— αρχικό
-------------------------------------------  δανεικό
Μειωτέος            1  0  1  1  1  0  0
Αφαιρετέος          0  1  0  1  0  0  1
-------------------------------------------
```

Διαφορά 0 1 1 0 0 1 1

Πίνακες μονοψήφιας πρόσθεσης και αφαίρεσης μπορούν να αναπτυχθούν για το οκταδικό και για το δεκαεξαδικό σύστημα αρίθμησης. Παρ' όλα αυτά λίγοι είναι εκείνοι που αποστηθίζουν τέτοιους πίνακες για ένα σωρό λόγους. Ακόμα και αν χρειαστεί να κάνουμε πράξεις σ' αυτά τα συστήματα είναι προτιμότερο ν' αγοράσουμε έναν υπολογιστή τσέπης που έχει τη δυνατότητα να κάνει τέτοιου είδους πράξεις. Επειδή όμως και αυτοί είναι πιθανόν κάποτε να μας προδώσουν καλό είναι να γνωρίζουμε μια πρακτική μέθοδο πρόσθεσης και αφαίρεσης σ' αυτά τα συστήματα. Η μέθοδος είναι απλή. Μεταφέρουμε όλα τα ψηφία στο δεκαδικό σύστημα όπου γίνονται οι πράξεις κατά στήλη. Ένα κρατούμενο παράγεται στην πρόσθεση κάθε φορά που το άθροισμα στήλης γίνει ίσο ή μεγαλύτερο από τη βάση του συστήματος αναχώρησης. Επίσης ένα δανεικό παράγεται στην αφαίρεση κάθε φορά που ο μειωτέος στήλης είναι μικρότερος από τον αφαιρετέο στήλης.

Παραδείγματα :

Κρατούμενα 0 1 1 0 0	
X	$19B9_{16}$
+ Y	$C7E6_{16}$
X +Y	$E19F_{16}$

0	1	1	0	0
	1	9	11	9
+ 12	7	14	6	
	14	17	25	15
	14	16+1	16+9	15
	E	1	9	F

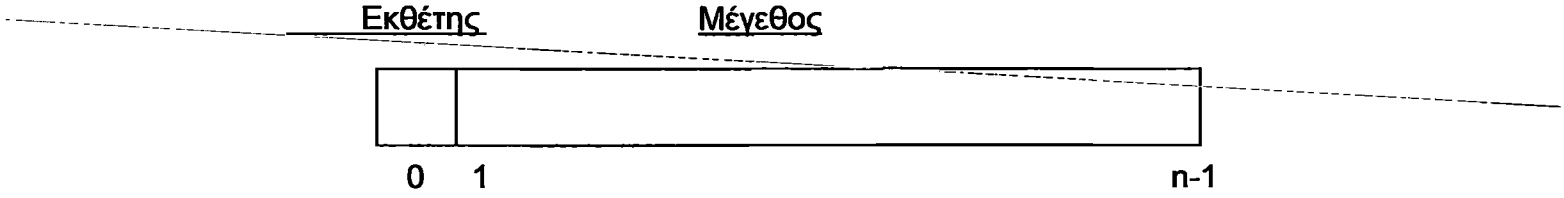
3.2.4. Παράσταση προσημασμένων ακεραίων

Ο πιο συνηθισμένος τρόπος να παραστήσει κανείς ένα προσημασμένο ακέραιο αριθμό είναι να τοποθετήσει το σύμβολο του πρόσημου (+ ή -) μπροστά από το μέγεθος (απόλυτη τιμή) του αριθμού. Η μέθοδος αυτή ονομάζεται πρόσημο-μέγεθος και δεν είναι άλλος από την αλγεβρική μέθοδο. Όταν όμως τις πράξεις τις κάνουν τα λογικά κυκλώματα ή μέθοδος αυτή δεν ο ιδανικός τρόπος να παραστήσει κανείς προσημασμένους αριθμούς. Πέρα από την ευκολία στη παράσταση πρέπει να οδηγεί και σε αριθμητική, η οποία υλοποιείται από απλά και φτηνά λογικά κυκλώματα. Για τους παραπάνω λόγους το σύστημα πρόσημο-μέγεθος έχει εγκαταλειφθεί και έχει αντικατασταθεί από άλλα πιο ευέλικτα συστήματα παράστασης. Στις παραγράφους που ακολουθούν θα καλύψουμε τρία από τα πιο γνωστά. Τα συστήματα αυτά, όπως είναι φυσικό ισχύουν για οποιαδήποτε βάση. Εμείς θα επικεντρώσουμε το ενδιαφέρον μας μόνο στο δυαδικό σύστημα

αρίθμησης.

Παράσταση Πρόσημο – Μέγεθος

Στο σύστημα πρόσημο-μέγεθος (Signed-Magnitude Representation), αν ο χώρος αποθήκευσης ενός ακεραίου αριθμού έχει μήκος n bits, τότε η παράσταση του χωρίζεται σε δύο τμήματα. Το πρώτο τμήμα αποτελείται από το bit 0 και το δεύτερο τμήμα από τα bits 1 έως η-1. Το πρώτο τμήμα περιέχει το πρόσημο του αριθμού και το δεύτερο το μέγεθος. Αν το πρόσημο περιέχει το ψηφίο 0 τότε αριθμός θα θεωρείται θετικός, αν περιέχει το 1 ο αριθμός θα θεωρείται αρνητικός.



Σύστημα παράστασης πρόσημο-μέγεθος

Η αρίθμηση των bits γίνεται από αριστερά προς τα δεξιά. Έτσι το MSB του μεγέθους βρίσκεται στη θέση 1 και το LSB στη θέση n-1. Το σύστημα αυτό της παράστασης των αρνητικών και κατ' επέκταση όλων των ακεραίων ονομάζεται πρόσημο-μέγεθος.

Παραδείγματα :

Αν n = 8 τότε:

$00101011_2 = +43_{10}$	$11111111_2 = -127_{10}$
$10101011_2 = -43_{10}$	$00000000_2 = +0_{10}$
$01111111_2 = +127_{10}$	$10000000_2 = -0_{10}$

Παρατηρούμε ότι υπάρχουν δύο παραστάσεις του μηδενός. Επίσης ο μεγαλύτερος αριθμός, κατ' απόλυτη τιμή, που μπορεί να παρασταθεί είναι ο $2^{7-1} = 127$. Γενικά σε έναν υπολογιστή μήκος θέσης η bits οι ακέραιοι αριθμοί περιέχονται μεταξύ των αριθμών $-(2^{n-1}-1)$ και $(2^{n-1}-1)$ συμπεριλαμβανομένων. Επομένως, ένας υπολογιστής με μήκος θέσης n bits μπορεί, στο σύστημα πρόσημο-μέγεθος, να κάνει πράξεις μόνο με ακεραίους προσημασμένους αριθμούς που περιλαμβάνονται στα παραπάνω όρια. Στο σημείο αυτό μας δίνεται η δυνατότητα να εκτιμήσουμε τη σημασία πού έχει το μήκος θέσης για την δυνατότητα του υπολογιστή. Στο σύστημα αυτό μπορούν να κωδικοποιηθούν 2^n διαφορετικοί προσημασμένοι ακέραιοι, από τους οποίους οι $2^{n-1}-1$ είναι αρνητικοί, οι $2^{n-1}-1$ είναι θετικοί και υπάρχουν δύο παραστάσεις του μηδενός.

Ας υποθέσουμε τώρα ότι έχουμε ένα μικροϋπολογιστή με επεξεργαστή 8086. Όπως ξέρουμε αυτός ο επεξεργαστής έχει μήκος θέσης μνήμης 16 bits , άρα ο μεγαλύτερος προσημασμένος ακέραιος πρέπει να είναι ο $2^{15-1} = 32767$. Είναι προφανές ότι ο υπολογιστής αυτός επεξεργάζεται και μεγαλύτερους ακέραιους. Τι γίνεται τότε ; Απλούστατα , προγραμματίζουμε τον υπολογιστή, έτσι ώστε να τοποθετεί τους προσημασμένους ακεραίους σε δύο συνεχόμενες θέσεις της μνήμης. Αμέσως αυξάνεται το μέγεθος των αριθμών που μπορούμε ν' αποθηκεύσουμε, σε βάρος όμως της χωρητικότητας της μνήμης. Δηλαδή η μνήμη θα χωρά τώρα λιγότερους "μεγάλους ακέραιους" από προηγουμένως. Στην περίπτωση του 8086 , ο μεγάλος ακέραιος θα έχει μέγιστο μέγεθος 2^{31-1}.

Οι πράξεις της πρόσθεσης και της αφαίρεσης ακολουθούν τους κλασσικούς αλγεβρικούς κανόνες. Συγκεκριμένα ισχύουν οι παρακάτω αλγόριθμοί :

Πρόσθεση :

ΑΝ τα πρόσημα είναι ίδια ΤΟΤΕ προσθέτουμε τα μεγέθη και δίνουμε στο αποτέλεσμα το κοινό πρόσημο.

ΔΙΑΦΟΡΕΤΙΚΑ

αφαιρούμε το μικρότερο μέγεθος από το μεγαλύτερο και δίνουμε στο αποτέλεσμα το πρόσημο του μεγαλύτερου.

Αφαίρεση :

Αλλάζουμε το πρόσημο του αφαιρετέου και κάνουμε πρόσθεση με τον παραπάνω αλγόριθμο πρόσθεσης.

Συμπλήρωμα μη προσημασμένου αριθμού ως προς βάση R: Θα υποθέσουμε και πάλι ότι έχουμε έναν υπολογιστή με μήκος θέσης η bit. Ένας ακέραιος δυαδικός αριθμός X με η ψηφία παριστάνεται από το διάνυσμα $(X_{\eta-1}\ X_{\eta-2}\ \ldots\ X_0)$.

Ονομάζουμε συμπλήρωμα ως προς τη βάση του συστήματος παράστασης του X, έναν ακέραιο αριθμό X' τέτοιον ώστε $X + X' = R^{\eta}$. Αν $R = 2$ τότε $X + X' = 2^{\eta}$. Το X' ονομάζεται συμπλήρωμα ως προς 2 του X. Το συμπλήρωμα του X ως προς 2 θα το συμβολίζουμε με **Σ2(X)**

Ο υπολογισμός του Σ2(X) γίνεται ως εξής:

$$\Sigma 2(X) = 2^{\eta} - X = (1\ 0\ 0\ \ldots 0) - (X_{\eta-1}\ X_{\eta-2}\ \ldots\ X_0).$$

$\eta+1$ όροι η όροι

$$\Sigma 2(X) = (1\ 1\ 1\ \ldots 1) + (0\ 0\ 0\ \ldots 1) - (X_{\eta-1}\ X_{\eta-2}\ \ldots\ X_0).$$

η όροι η όροι

$$\Sigma 2(X) = (1-X_{\eta-1}\ \ 1-X_{\eta-2}\ \ \ldots\ 1-X_0) + (0\ 0\ 0\ \ldots 1)$$

Από την τελευταία σχέση προκύπτει ο παρακάτω απλός κανόνας για τον υπολογισμό του συμπληρώματος ενός αριθμού.

Κανόνας: ***Το συμπλήρωμα ως προς 2 ενός δυαδικού αριθμού Χ, υπολογίζεται αν αντιστρέψουμε ένα προς ένα τα ψηφία του και προσθέσουμε τη μονάδα (δυαδική πρόσθεση).***

Παραδείγματα:

Αν n = 8 τότε

$X = 17_{10} = 00010001_2$

11101110_2 Αντιστροφή

$+ \quad 1$ Πρόσθεση

$\Sigma 2(17) = 11101111_2 = 239_{10}$

$X = 127_{10} = 01111111_2$

10000000_2

$+ \quad 1$

$\Sigma 2(127) = 10000001_2 = 129_{10}$

$X = 99_{10} = 01100011_2$

10011100_2

$+ \quad 1$

$\Sigma 2(99) = 10011101_2 = 157_{10}$

Ένας δεύτερος πρακτικός κανόνας είναι: "Αντιστρέφουμε τα ψηφία ένα προς ένα αρχίζοντας από αριστερά προς τα δεξιά μέχρι να συναντήσουμε την δεξιότερη μονάδα.

Σύστημα παράστασης: Με βάση τον ορισμό του συμπληρώματος ως προς 2 ορίζεται το ομώνυμο σύστημα παράστασης των προσημασμένων ακεραίων αριθμών σε υπολογιστή με μήκος θέσης n bits. Στο σύστημα όλοι οι μη αρνητικοί (θετικοί ή μηδέν) αριθμοί που είναι μικρότεροι ή ίσοι από 2^{n-1}-1 παριστάνονται όπως ακριβώς στην παράσταση πρόσημο-μέγεθος. Οι αρνητικοί αριθμοί από -2^{n-1} μέχρι -1 συμπεριλαμβανομένων παριστάνονται με το συμπλήρωμα ως προς 2 της απολύτου τιμής του Χ. Επειδή το συμπλήρωμα ως προς 2 του -2^{n-1} είναι 2^{n-1} και του -1 είναι 2^n-1 συμπεραίνουμε ότι το MSB θα είναι πάντοτε 1. Αυτή η τελευταία παρατήρηση σημαίνει ότι αν θέλουμε ν' ανιχνεύσουμε την αρνητικότητα μιας παράστασης στο δεν έχουμε παρά ν' ανιχνεύσουμε το MSB της παράστασης. Σε καμία περίπτωση όμως το MSB δεν μπορεί να θεωρηθεί πρόσημο.

οι ακέραιοι που μπορούν να παρασταθούν σε μια θέση των η bits πρέπει να βρίσκονται στο διάστημα $(-2^{n-1}, 2^{n-1}-1)$ των ορίων συμπεριλαμβανομένων.

Στο σύστημα αυτό μπορούν να κωδικοποιηθούν 2^n διαφορετικοί προσημασμένοι ακέραιοι, από τους οποίους οι 2^{n-1} είναι αρνητικοί, οι 2^{n-1}-1 είναι θετικοί και υπάρχει και μια παράσταση του μηδενός.

Το χρησιμοποιείται σε όλους τους υπολογιστές μεσαίου μεγέθους και τους μικροϋπολογιστές. Χρησιμοποιείται επίσης και στους περισσότερους μεγάλους υπολογιστές που βρίσκονται εν λειτουργία σήμερα.

Στο σημείο αυτό θα πρέπει να πούμε ότι η παράσταση ενός αριθμού στο δεν εξαρτάται μόνο από τον αριθμό , αλλά και από το πλήθος των bits της θέσης της μνήμης που θα αποθηκευθεί ο αριθμός.

Παράδειγμα :

Να βρεθεί η παράσταση του (-7) στο ΣΤ2 σε υπολογιστή με μήκος θέσης 4 ψηφία και 8 ψηφία.

Σύμφωνα με τα παραπάνω, η παράσταση του -7 στο ΣΤ2 για n=4 είναι :

7 = 0111 ⟶ αντιστροφή 1000

\+ 1

Σ2(-7) = 1001

Επίσης, η παράσταση του -7 στο ΣΤ2 είναι το Σ2(7) για n=8 είναι:σε 8

7 = 00000111 ⟶ αντιστροφή 11111000

\+ 1

Σ2(-7) = 11111001

Ο πίνακας παρακάτω πίνακας δείχνει τις διαφορετικές παραστάσεις του -8 όταν μεταβάλλεται το μήκος n της θέσης αποθήκευσης.

n	Παράσταση στο ΣΤ2
2	Δεν υπάρχει
3	Δεν υπάρχει
4	1000
5	11000
6	111000
7	1111000
8	11111000

Παραστάσεις το –8 στο ΣΥΣΤΗΜΑ

Άσκηση :

Δίνεται ένας η παράσταση ενός αριθμού $X = (X_3\ X_2\ X_1\ X_0)_{10}$ όπου X_0, X_1, X_2, X_3 ανήκουν στο σύνολο {0,1,2,..9}. Να βρεθεί το συμπλήρωμα του αριθμού X ως προς βάση 10.

Λύση:

Αν ονομάσουμε X' το συμπλήρωμα του X ως προς 10 τότε

$X + X' = 10^4$ (υποθέτουμε ότι έχουμε 4 ψηφία) , άρα

$X' = 10^4 - X$

$X' = 9999_{10} + 0001_{10} - (X_3\ X_2\ X_1\ X_0)_{10}$

$X' = (9\text{-}X_3\ \ 9\text{-}X_2\ \ 9 - X_1\ \ 9\text{-}X_0)_{10} + 0001_{10}$

Ο κανόνας λοιπόν εύρεσης του ΣΤ10 διαμορφώνεται ως εξής:

"Αφαιρούμε ένα προς ένα τα ψηφία του αριθμού από το 9 και προσθέτουμε τη μονάδα".

Αν για παράδειγμα $X = 1849_{10}$ τότε:

$$\begin{array}{rl} & 8150_{10} \\ + & \quad 1 \\ \hline \Sigma 10(X) = \quad X' = & 8151_{10} \end{array}$$

Πρόσθεση: Πριν συζητήσουμε τον αλγόριθμο της πρόσθεσης στο ΣΤ2 παρουσιάζουμε τον επόμενο πίνακα, ο οποίος περιέχει τα τρία πλέον γνωστά συστήματα παράστασης προσημασμένων ακεραίων αριθμών. Στον πίνακα αυτό δείχνουμε πόσους και ποιους αριθμούς μπορούν να παραστήσουν τα τρία συστήματα σ' ένα υπολογιστή με μήκος θέσης 4 bits..

Δεκαδικοί	ΣΤ2	ΣΤ1	S-M
-8	1000	-	-
-7	1001	1000	1111
-6	1010	1001	1110
-5	1011	1010	1101
-4	1100	1011	1100
-3	1101	1100	1011
-2	1110	1101	1010
-1	1111	1110	1001
		1111	**1000**
0	**0000**	**ή**	**ή**
		0000	**0000**
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	0101	0101
6	0110	0110	0110
7	0111	0111	0111

Πρέπει να γίνει σαφής η έννοια της χωρητικότητας μιας θέσης μήκους n bits. Ονομάζουμε έτσι το μέγιστο πλήθος των προσημασμένων ακεραίων που μπορούν να παρασταθούν με n bits. Όπως φαίνεται από τον παραπάνω πίνακα η χωρητικότητα εξαρτάται και από το σύστημα παράστασης. Αλλά ο κρίσιμος παράγοντας για την επιλογή του ενός ή του άλλου συστήματος δεν είναι η χωρητικότητα, αλλά η "ευκολία" πραγματοποίησης των αριθμητικών πράξεων. Η "εύκολη αριθμητική" είναι φανερό πως έχει άμεση επίπτωση στην πολυπλοκότητα των λογικών κυκλωμάτων και κατ' επέκταση στο κόστος του υπολογιστή. Σε λίγο θα διαπιστώσουμε γιατί το ΣΤ2 είναι το περισσότερο δημοφιλές σύστημα.

Αν αρχίσουμε το μέτρημα από το $1000_2 = (-8)_{10}$ μέχρι το $0111_2 = (+7_{10})$ θα παρατηρήσουμε ότι κάθε αριθμός προκύπτει από τον προηγούμενο του αν προσθέσουμε τη μονάδα, και αγνοήσουμε κάθε κρατούμενο πέρα από το 4 ψηφίο. Επειδή λοιπόν η "πρόσθεση" δεν είναι τίποτε άλλο παρά μια διαδοχική πρόσθεση της μονάδας ή μια επέκταση της μέτρησης, δύο αριθμοί στο σύστημα παράστασης ΣΤ2 προστίθενται σύμφωνα με τον παρακάτω κανόνα :

Κανόνας πρόσθεσης:

Ακολουθούμε τους κανόνες της πρόσθεσης μη προσημασμένων δυαδικών Αριθμών και αγνοούμε κάθε κρατούμενο πέρα από το MSB.

Παραδείγματα :

Αν η=4 τότε έχουμε:

+3	0011		-2	1110	
+ +4	+0100		+ -6	+ 1010	
+7	0111	NO overflow	-8	11000	NO overflow
+6	0110		+4	0100	
+ -3	+1101		+ -7	+1001	
+3	10011	NO overflow	-3	1101	NO overflow

Εάν μια πράξη πρόσθεσης παράγει ένα αποτέλεσμα το οποίο είναι έξω από τα όρια του συστήματος παράστασης τότε λέμε ότι έχουμε **υπερχείλιση (Overflow)**. Με άλλα λόγια υπερχείλιση είναι το φαινόμενο όπου αριθμοί n bits μετά την διαδικασία μιας πράξης δίνουν αποτέλεσμα που δεν παριστάνεται με n bits. Όπως θα φανεί στα επόμενα παραδείγματα, στη πρόσθεση, αριθμοί με διαφορετικό πρόσημο δεν παρουσιάζουν υπερχείλιση ενώ αντίθετα αριθμοί με το ίδιο πρόσημο παρουσιάζουν.

Παραδείγματα :

-3	1101	+5	0101

```
+ -6   + 1010                    + +6  + 0110
  -9    10111 **overflow**        +11   1011 **overflow**
   ↓
  -8     1000                      +7   0111
+ -8   + 1000                    + +7  + 0111
 -16    10000 **overflow**        +14   1110 **overflow*
   ↓
```

κανόνας υπερχείλισης για τη πρόσθεση

ΑΝ οι αριθμοί είναι εταιρόσημοι ΤΟΤΕ δεν υπάρχει υπερχείλιση
ΔΙΑΦΟΡΕΤΙΚΑ

ΑΝ το MSB του ενός προσθετέου είναι το ίδιο
με το MSB του αποτελέσματος ΤΟΤΕ δεν υπάρχει υπερχείλιση
ΔΙΑΦΟΡΕΤΙΚΑ
υπάρχει υπερχείλιση

Ο μηχανισμός ελέγχου της υπερχείλισης σε άλλες μηχανές ελέγχεται από το λογικό και σε άλλες από το υλικό.

Αφαίρεση: Δύο αριθμοί μπορούν να αφαιρεθούν σαν να ήταν μη προσημασμένοι δυαδικοί αριθμοί. Εν τούτοις όμως σπάνια γίνεται αυτό. Συνήθως εφαρμόζεται μια μέθοδος η οποία βασιζόμενη στις ιδιότητες του συστήματος, μετατρέπει την αφαίρεση σε πρόσθεση. Η μέθοδος αυτή εξοικονομεί λογικά κυκλώματα και περιγράφεται από τον αλγόριθμο:

Κανόνας αφαίρεσης

Β0 : Αντιστρέφουμε τα ψηφία του αφαιρετέου
Β1 : Προσθέτουμε τον αριθμό που θα προκύψει
από το βήμα 1 στον μειωτέο με αρχικό κρατούμενο 1 αντί 0.

Παραδείγματα :

```
                                        1 αρχικό
κρατούμενο.   +4      0100                      0100
- +3        - 0011   ──────────►   + 1100
                      Αντιστροφή
 +1                                  10001  NO overflow
                                          ↙
                                        1 αρχικό
κρατούμενο.
```

```
  +3          0011                      0011
- -4        - 1100   ──────────►     + 0011
 +7                                    0111  NO overflow

                                        1 αρχικό κρατούμενο.
  +3          0011                      0011
- +4         -0100   ──────────►     + 1011
```

+1			1111	NO overflow
			1	αρχικό κρατούμενο.
+3	0011		0011	
- -6	- 1010	⟶	+ 0101	
+9			1001	**overflow**

Η υπερχείλιση στην αφαίρεση ανιχνεύεται όπως ακριβώς στη πρόσθεση. Ελέγχουμε τα MSB του μειωτέου και του αντιστρόφου του αφαιρετέου.

Συμπλήρωμα αριθμού ως προς μειωμένη βάση: Ονομάζουμε συμπλήρωμα ως προς μειωμένη βάση R ενός μη προσημασμένου αριθμού $X = (X_{\eta-1}\ X_{\eta-2}\ \dots\ X_0)_R$ έναν άλλον αριθμό X' τέτοιον ώστε $X + X' = R^{n}.-1$ Για το δυαδικό σύστημα το συμπλήρωμα αυτό ονομάζεται συμπλήρωμα ως προς 1 ή Σ1(X) και ισχύει $X+X'=2^{n}-1$.

Ο υπολογισμός του X' γίνεται ως εξής:

$$X' = (2^{n}-1) - X = (1\ 1\ \dots 1) - (X_{\eta-1}\ X_{\eta-2}\ \dots\ X_0)$$

ΚΑΙ η όροι $X' = (1-X_{\eta-1}\ \ 1-X_{\eta-2}\ \dots\ 1-X_0)$

Ερμηνεύοντας την τελευταία σχέση προκύπτει ο παρακάτω απλός κανόνας για τον υπολογισμό του συμπληρώματος ως προς 1 ενός αριθμού X.

Κανόνας: ***Το συμπλήρωμα ως προς 1 ενός μη προσημασμένου δυαδικού ακεραίου αριθμού X υπολογίζεται αν αντιστρέψουμε ένα προς ένα τα ψηφία του.***

Παράδειγμα : Αν η=8 τότε:

$X = 17_{10} = 00010001_2$ Σ1(17) $= 11101110_2$

$X = 119_{10} = 01110111_2$ Σ1(119) $= 10001000_2$

$X = 0_{10} = 00000000_2$ Σ1(0) $= 11111111_2$

$X = 99_{10} = 01100011_2$ Σ1(99) $= 10011100_2$

$X = 127_{10} = 01111111_2$ Σ1(127) $= 10000000_2$

Σύστημα παράστασης: Στο σύστημα παράστασης οι προσημασμένοι αριθμοί που μπορούν να χωρέσουν σε μία θέση μνήμης μήκους n bits ορίζονται ως εξής:

max = (e-127)max = emax-127 = 254-127 = 127

Fmax = (1.f)max = $1.1111....1111_2 = 1.999999999.._{10}$

Rmax = 2^{127} x 1.99999.... ≅ 2^{128} ≅ 3.4×10^{38}

Υπολογισμός Rmin:

Rmin = 2^{Emin} x Fmin

Emin = (e-127)min = emin-127 = 1-127. Όλοι οι μη αρνητικοί αριθμοί (θετικοί και μηδέν) που είναι μικρότεροι από το $2^{\eta-1}-1$ συμπεριλαμβανομένου, παριστάνονται όπως ακριβώς στο σύστημα πρόσημο-μέγεθος. Οι αριθμοί X από $-(2^{\eta-1}-1)$ μέχρι και 0 παριστάνονται με το

συμπλήρωμα ως προς 1 της απολύτου τιμής του Χ. Έχουμε δύο παραστάσεις του μηδενός, την (0 0 ... 0) και την (1 1 ... 1).

Όπως και στο MSB θα είναι πάντοτε 1 για τους αρνητικούς και 0 για του θετικούς (γιατί;).

Τέλος σε μια θέση των n bits μπορούν να παρασταθούν οι ακέραιοι που βρίσκονται μεταξύ των ορίων $-(2^{n-1}-1)$ ως $(2^{n-1}-1)$ συμπεριλαμβανομένων και κωδικοποιεί τούς ίδιους ακριβώς αριθμούς με το σύστημα πρόσημο-μέγεθος, δηλαδή 2^n.

Το κύριο πλεονέκτημα του ΣΤ1 είναι η συμμετρία του και η ευκολία της εύρεσης του συμπληρώματος ως προς 1. Η πρόσθεση όμως στο σύστημα αυτό είναι πιο πολύπλοκη από αυτήν του ΣΤ2. Χρησιμοποιήθηκε στο παρελθόν από μερικούς κατασκευαστές αλλά σήμερα σπάνια περιλαμβάνεται στους νέους σχεδιασμούς.

Πρόσθεση Αν παρατηρήσουμε προσεκτικά τον πίνακα που περίεχει τα συστήματα και συγκεκριμένα που, θα πάρουμε μια ιδέα πως πρέπει να γίνεται η πρόσθεση σ' αυτό το σύστημα. Αρχίζοντας από το 1000_2 (-7_{10}) και προχωρώντας προς τα κάτω, θα παρατηρήσουμε ότι κάθε αριθμός προκύπτει από τον προηγούμενο αν προσθέσουμε τη μονάδα, εκτός από τη μεταφορά από το 1111_2 (-0) στο 0001_2 $(+1_{10})$. Και αυτό γιατί ενδιάμεσα παρεμβάλλεται το 0000 (+0). Για να καλύψουμε αυτήν την "ανωμαλία" προσθέτουμε το 2 αντί για το 1 όταν θέλουμε να μεταφερθούμε από το 1111_2 στο 0001^2. Συνοψίζοντας, μπορούμε να φτιάξουμε τον κανόνα " Για να διασχίσουμε τον πίνακα αυξάνουμε κατά 1 εκτός από τη μετάβαση μας από 1111 στο 0001 όπου αυξάνουμε κατά 2". Και ο κανόνας της πρόσθεσης γίνεται:

Κανόνας πρόσθεσης

Εκτελούμε την δυαδική πρόσθεση,
αν υπάρχει κρατούμενο πέρα από το MSB το προσθέτουμε στο αποτέλεσμα
Η μέθοδος αυτή είναι γνωστή και σαν End-around carry.

Παραδείγματα :

```
   +3     0011          +4     0100
 + +4   + 0100        + -7   + 1000
   +7     0111  NO overflow   -3     1100  NO overflow

   -2     1101          +6     0110
 + -5   +1010         + -3   +1100
   -7   1¦0111
+3  1¦0010
          + 1                  + 1
        1000  NO overflow    0011  NO overflow
```

+5	0101		-4	1011	
+ -5	+ 1010		+ -4	+ 1011	
0	1111	NO overflow	-8	1¦0110	
				+ 1	
				0111	** overflow

Αφαίρεση: ο πιο εύκολος τρόπος να κάνει κανείς αφαίρεση, είναι να βρει το αντίστροφο του αφαιρετέου και να το προσθέσει στον μειωτέο.

Το τέχνασμα αυτό έχει τα παρακάτω βήματα:

Κανόνας αφαίρεσης

Β0 : Αντιστρέφουμε τον αφαιρετέο
Β1 : Προσθέτουμε στον αριθμό
που θα προκύψει από το βήμα 0, στον μειωτέο.

Παραδείγματα :

+4	0100		0100	
- +3	- 0011	αντιστροφή →	+ 1100	
+1			1¦0000	
			+ 1	
			0001	NO overflow

+2	0010		0010	
- -2	- 1101	αντιστροφή →	+ 0010	
+4			0100	NO overflow

+4	0100		0100	
- -5	- 1010	αντιστροφή →	+ 0101	
+9			1001	overflow

Όπως έγινε αντιληπτό οι κανόνες για την υπερχείλιση είναι οι ίδιοι με αυτούς που περιγράψαμε στο σύστημα ΣΤ2.

3.2.5. Πολλαπλασιασμός μη προσημασμένων ακεραίων

Η μέθοδος πολλαπλασιασμού μη προσημασμένων ακεραίων είναι απλή και δεν είναι άλλη από αυτή που μάθαμε στο δημοτικό σχολείο. Δηλαδή, βρίσκουμε ένα άθροισμα το οποίο αποτελείται από μετατοπισμένα γινόμενα, τα οποία προέκυψαν από τον πολ/μό των ψηφίων του πολλαπλασιαστή με τον πολλαπλασιαστέο.

Παράδειγμα :

1011_{10}		1011_2	Πολλαπλασιαστέος
x 1100_{10}		x 1100_2	Πολλαπλασιαστής
0000		0000	
0000		0000	Μετατοπισμένα
1011		1011	γινόμενα.
+ 1011		+ 1011	
1112100_{10}		10000100_2	Γινόμενο.

Όπως θα μπορούσε να παρατηρήσει κανείς τα μετατοπισμένα γινόμενα στο δυαδικό σύστημα προκύπτουν πάρα πολύ εύκολα, γιατί οι πιθανές τιμές των ψηφίων με τα οποία πολ/ζεται ο πολ/στέος είναι 0 και 1.

Μια άλλη μέθοδος που βολεύει τον Η/Υ, είναι αυτή όπου κάθε μετατοπισμένο γινόμενο προστίθεται αμέσως σ' ένα μερικό γινόμενο το οποίο μεταβάλλεται σε τελικό όταν τελειώσει ο πολ/μός.

Παράδειγμα :

1011	
x 1100	
00000000	1ο μερικό άθροισμα
+ 0000	2ο μετατοπισμένο γινόμενο.
00000000	2ο μερικό άθροισμα.
+ 0000	2ο μετατοπισμένο γινόμενο.
00000000	3ο μερικό άθροισμα.
+ 1011	3ο μετατοπισμένο γινόμενο.
00101100	4ο μερικό άθροισμα.
+1011	4ο μετατοπισμένο γινόμενο.
10000100	τελικό γινόμενο.

Παρατηρούμε ότι χρησιμοποιήσαμε ένα καταχωρητή μήκους 2^n bits για να αθροίσουμε Τα μερικά γινόμενα τα οποία θα μας οδηγήσουν στο τελικό γινόμενο. αποθηκεύσουμε το γινόμενο.

	n bits	Πολλαπλασιαστέος
X	n bits	Πολλαπλασιαστής
A μέρος γινομ.	B μέρος γινομ.	Γινόμενο
n bits	n bits	

Ο έλεγχος υπερχείλισης γίνεται ως εξής :

ΑΝ το Α μέρος του γινομένου(τα n αριστερά bits) είναι μηδέν
ΤΟΤΕ δεν υπάρχει υπερχείλιση
ΔΙΑΦΟΡΕΤΙΚΑ υπάρχει υπερχείλιση

Πολλαπλασιασμός προσημασμένων ακεραίων αριθμών

Ας δούμε τώρα πως γίνεται ο πολ/μος στη περίπτωση προσημασμένων ακεραίων αριθμών.

Μέγεθος Σύστημα Πρόσημο

Πολ/ζουμε τις απόλυτες τιμές των παραγόντων του πολ/μού και μετά χαρακτηρίζουμε το γινόμενο θετικό, αν οι παράγοντες είναι ομόσημοι και αρνητικό αν είναι ετερόσημοι.

Σύστημα: Για να κάνουμε πολ/μό στο σύστημα αυτό ακολουθούμε τον παρακάτω αλγόριθμο:

Β0 : Υπολογίζουμε το μέγεθος του πολλαπλασιαστέου
και του πολλαπλασιαστή
Β1 : Υπολογίζουμε το γινόμενο των μεγεθών
Β2 : Βρίσκουμε το πρόσημο του γινομένου
Β3 : ΑΝ το πρόσημο είναι αρνητικό ΤΟΤΕ υπολογίζουμε το συμπλήρωμα ως προς 2 του γινομένου των μεγεθών.

Παράδειγμα :

Δεκαδικό			μεγέθη	
-4	1100	μέγεθος →	0100	θέση 4 bits
x +2	x 0010		x 0010	θέση 4 bits
-8			00000000	θέση 8 bits
			+ 0000	
			00000000	
			+ 0100	
			00001000	Σ → 1111000 =-8_{10}

Αν θέλουμε να κάνουμε έλεγχο υπερπλήρωσης αυτό πρέπει να γίνει στο μέγεθος του αποτελέσματος δηλαδή με τον αριθμό 00001000. Σύμφωνα με τα παραπάνω το αποτέλεσμα είναι δεκτό.

Σύστημα 1: Ισχύουν ακριβώς ότι και για το σύστημα 2

Διαίρεση μη προσημασμένων ακεραίων

Ο αλγόριθμος της διαίρεσης στο δυαδικό σύστημα ακολουθεί, όπως θα περίμενε κανείς, τους ίδιους κανόνες, μ' αυτούς που χρησιμοποιούμε στη γνωστή μας από το σχολείο δεκαδική διαίρεση. Για του λόγου το αληθές θα εκτελέσουμε μια διαίρεση

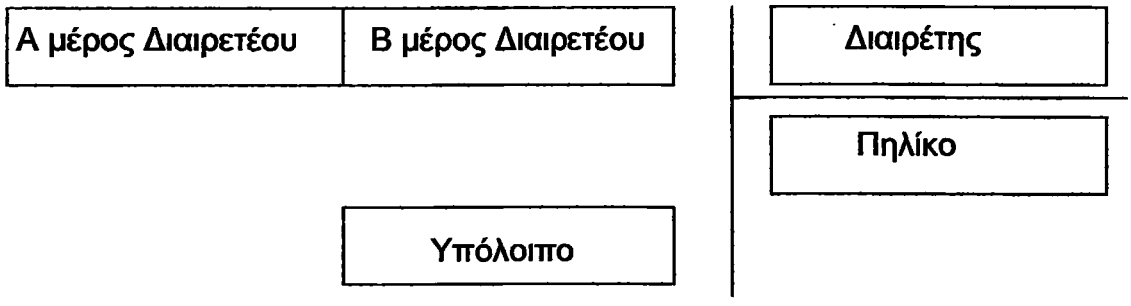
και στα δύο συστήματα. Παρατηρούμε ότι στη περίπτωση της δυαδικής διαίρεσης το μετατοπισμένα πολλαπλάσια είναι το μηδέν ή ο ίδιος ο διαιρέτης. Αυτό έχει σαν συνέπεια, όπως και στον πολλαπλασιασμό άλλωστε, ή διαδικασία της διαίρεσης και του πολλαπλασιασμού να πραγματοποιείται από αλγορίθμους που περιλαμβάνουν πρόσθεση (ή έμμεση πρόσθεση) και μετατόπιση.

Παραδείγματα :

Διαιρετέος	217	11	Διαιρέτης
Μετατοπισμένο πολ/σιο Διαιρέτη	- 11	19	Πηλίκο
Νέος διαιρετέος	107		
Μετατοπισμένο πολ/σιο Διαιρέτη	- 99		
Υπόλοιπο	8		

Διαιρετέος	11011001	1011	Διαιρέτης
Μετατοπισμένο πολ/σιο Διαιρέτη	- 1011	10011	Πηλίκο
Νέος Διαιρετέος	00101		
Μετατοπισμένο πολ/σιο διαιρέτη	- 0000		
Νέος διαιρετέος	1010		
Μετατοπισμένο πολ/σιο διαιρέτη	- 0000		
Νέος διαιρετέος	10100		
Μετατοπισμένο πολ/σιο διαιρέτη	- 1011		
Νέος διαιρετέος	10011		
Μετατοπισμένο πολ/σιο διαιρέτη	- 1011		
Υπόλοιπο	1000		

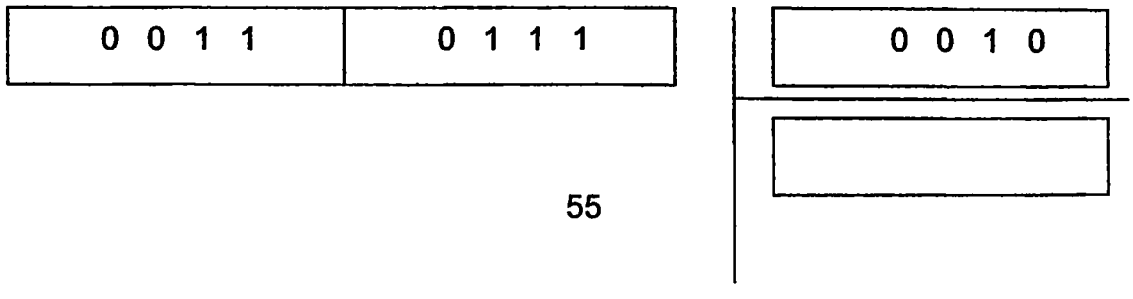
Όταν μια ακέραια διαίρεση εκτελείται από υπολογιστή ο διαιρετέος τοποθετείται αυτόματα σε ένα καταχωρητή με μήκος διπλάσιο από μια θέση μνήμης δεξιά προσανατολισμένος

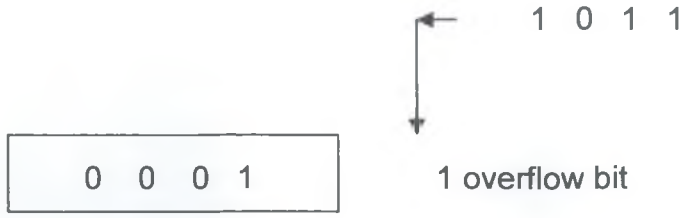


Υπερχείλιση παρουσιάζεται όταν ο διαιρέτης είναι μηδέν ή όταν το πηλίκο δεν χωράει στα n bits. Η τελευταία περίπτωση ανιχνεύεται ως εξής: Απομονώνουμε το πρώτο μέρος του διαιρετέου και αν συμβαίνει ο διαιρέτης να είναι μικρότερος ή ίσος από αυτό τότε έχουμε υπερχείλιση.

Παράδειγμα:

Να ελεγχθεί αν σε υπολογιστή με θέση μήκους 4 bits η διαίρεση 55 : 2 παρουσιάζει υπερχείλιση.





Το πρώτο μέρος του διαιρετέου είναι $0011_2 = 3_{10}$ και είναι μεγαλύτερο από τον διαιρέτη (2_{10}). Αρα έχουμε υπερχείλιση.

Πράγματι το πηλίκο είναι 11011_2 και δεν χωράει σε μια απλή θέση μνήμης .

Διαίρεση προσημασμένων ακεραίων αριθμών

Σύστημα πρόσημο - Μέγεθος

Σε μια μηχανή με μήκος θέσης η bits, ο διαιρετέος μεταφέρεται από μία θέση μνήμης τοποθετείται σε ένα καταχωρητή διπλασίου μήκους , ο διαιρέτης σε μια θέση και το πηλίκο, υπόλοιπο λαμβάνονται σε μια θέση . Η διαίρεση γίνεται ως εξής:

α: Γίνεται η δυαδική διαίρεση των μεγεθών.
β : Πρόσημο του υπολοίπου είναι το πρόσημο του διαιρετέου.
γ : Πρόσημο του πηλίκου είναι το αλγεβρικό πρόσημο του πηλίκου, δηλαδή αν οι παράγοντες της διαίρεσης είναι ομόσημοι τότε είναι (+) διαφορετικά είναι (-).

Στα άλλα δύο συστήματα υπάρχουν ειδικές τεχνικές διαίρεσης οι οποίες όμως εκτελούνται απ' ευθείας από το Hardware. Όπου όμως η διαίρεση γίνεται από το Software εντοπίζουμε τα πρόσημα και τα μεγέθη των παραγόντων, η διαίρεση γίνεται με τα μεγέθη, εντοπίζονται τα πρόσημα του πηλίκου και του υπόλοιπου τα οποία μεταφέρονται πάλι στο αρχικό σύστημα παράστασης.

3.2.6. Άλλοι αριθμητικοί κώδικες

Για να υπάρχει ικανοποιητική επικοινωνία ανθρώπου-υπολογιστή πρέπει οι πληροφορίες που ανταλλάσσονται να είναι σε μορφή άμεσα αντιληπτή από τον άνθρωπο. Για παράδειγμα, οι αριθμητικές πληροφορίες πρέπει δίνονται ή να διαβάζονται, στο δεκαδικό σύστημα αρίθμησης. Είναι όμως γνωστό ότι η εσωτερική παράσταση αυτών των αριθμητικών μεγεθών, μέσα στον υπολογιστή είναι σε διαφορετικό σύστημα. Η μετατροπή έχει ένα σημαντικό χρονικό κόστος που αυξάνει όταν η εσωτερική παράσταση είναι πολύπλοκη, όπως είναι η παράσταση κινητού σημείου. Η χρονική αυτή καθυστέρηση στη μετατροπή από την εξωτερική απλή παράσταση στην εσωτερική πολύπλοκη έχει και τα πλεονεκτήματα της όπως είδαμε στην μελέτη των διαφόρων συστημάτων. Όταν όμως δεν κερδίζουμε τίποτε από τις πολύπλοκες παραστάσεις, εξαιτίας των περιορισμένων απαιτήσεων μας σε υπολογισμούς κυρίως, τότε χρησιμοποιούμε συστήματα παράστασης με λιγότερο πολυπλοκότητα και με μικρό υπολογιστικό κόστος

μετατροπής από/προς τον υπολογιστή. Οι πιο σημαντικοί από τους κώδικες που χρησιμοποιούμε είναι ο BCD και ο Gray.

Κώδικας BCD

Σ' αυτό το σύστημα (Binary-Coded-Decimal) τα ψηφία του δεκαδικού συστήματος αντικαθίστανται το καθένα από την 4 bit μη προσημασμένη δυαδική τους παράσταση (πίνακας 1.11). Παρατηρούμε ότι οι παραστάσεις 1010, 1011, 1100, 1111, δεν έχουν νόημα. Ο BCD κώδικας είναι ένα θεσιακός κώδικας.

Παράδειγμα :

5	6	1	2	6	3	Δεκαδικό σύστημα
0101	1100	0001	0010	0110	0011	Σύστημα BCD

Δύο BCD ψηφία καταλαμβάνουν χώρο 8 bit και μπορούν να παραστήσουν τους μη αρνητικούς αριθμούς από 0 έως 99. Αν σ' αυτό το χώρο τοποθετούσαμε τους γνωστούς 8 bit μη προσημασμένους αριθμούς θα είχαμε παράσταση από 0 έως 256. Βλέπουμε λοιπόν ότι έχουμε μια σημαντική σπατάλη χώρου μνήμης. Παρ' όλα αυτά όμως ο κώδικας αυτός χρησιμοποιείται π.χ τοπικά από τερματικούς σταθμούς που παίζουν το ρόλο ταμιακών μηχανών. Το κέρδος από την χρήση του BCD κώδικα βρίσκεται στο πλεονέκτημα του να μετατρέπονται τα δεκαδικά ψηφία σε δυαδική παράσταση χωρίς χρονοβόρους υπολογισμούς (μέθοδος διαίρεσης με το 2) αλλά με την απλή αναφορά σ' έναν πίνακα με 10 θέσεις ο οποίος περιέχει τα ψηφία του δεκαδικού συστήματος και τις αντίστοιχες δυαδικές τετράδες. Πολλοί υπολογιστές διαθέτουν ένα πλήρες σύνολο από εντολές που εκτελούν πράξεις μεταξύ των BCD αριθμών. Όπως και με τους δυαδικούς αριθμούς υπάρχουν αντίστοιχα συστήματα για την παράσταση των αρνητικών BCD αριθμών.

Ο κώδικας Gray

Στον κώδικα Gray, δυο διαδοχικές παραστάσεις διαφέρουν μεταξύ τους μόνο κατά ένα bit. Για τον λόγο αυτό ονομάζεται και κώδικας μοναδιαίας απόστασης (unit-distance code).Ο κώδικας αυτός δεν είναι θεσιακός.

Ο κώδικας Gray, εξαιτίας της ιδιότητας του (unit-distance code) χρησιμοποιείται πολλές φορές για την μεταφορά πληροφοριών μεταξύ των περιφερικών μονάδων και της κεντρικής μονάδας του υπολογιστή.

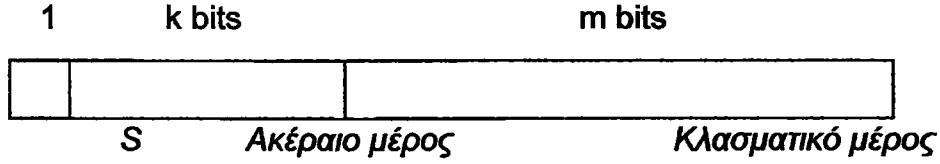
Δεκαδικό	BCD-4	BCD-8	Gray-2	Gray-3 Gray-4
0	0000	0000 0000	00	0 00 0 000
1	0001	0000 0001	01	0 01 0 001
2	0010	0000 0010	11	0 11 0 011
3	0011	0000 0011	10	0 10 0 010

4	0100	0000 0100	-	1 10	0 110
5	0101	0000 0101	-	1 11	0 111
6	0110	0000 0110	-	1 01	0 101
7	0111	0000 0111	-	1 00	0 100
8	1000	0000 1000	-	-	1 100
9	1001	0000 1001	-	-	1 101
10	-	0001 0000	-	-	1 111
11	-	0001 0001	-	-	1 110
12	-	0001 0010	-	-	1 010
13	-	0001 0011	-	-	1 011
14	-	0001 0100	-	-	1 001
15	-	0001 0101	-	-	1 000

3.2.7. Παράσταση πραγματικών αριθμών

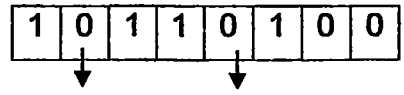
Παράσταση σταθερού σημείου

Στην παράσταση αριθμών σταθερού σημείου (Fixed point numbers) η λέξη των n bits διαιρείται σε τρία τμήματα. Έχουμε ένα τμήμα μήκους 1 για το πρόσημο , ένα τμήμα μήκους κ για το ακέραιο μέρος και ένα τμήμα μήκους m για το κλασματικό μέρος.



Παράσταση σταθερού σημείου

Το δυαδικό σημείο κατέχει σταθερή θέση μεταξύ των ψηφίων. Στο επόμενο παρουσιάζουμε μια παράσταση σταθερού σημείου με 3 ακέραιες και 4 κλασματικές θέσεις.



Παράδειγμα παράστασης σταθερού σημείου

Είναι φανερό ότι οι αριθμοί που μπορούν να παρασταθούν σ' ένα σύστημα

τέτοιο βρίσκονται στο διάστημα από -7.9375 ως 7.9375. Οι ακέραιοι αριθμοί μπορούν να θεωρηθούν σαν αριθμοί σταθερού σημείου με το δυαδικό σημείο πέρα από το LSB.

Παράδειγμα :

Ας υποθέσουμε ότι διαθέτουμε δύο υπολογιστές των 8 bits. Ο υπολογιστής Α χειρίζεται μόνο ακεραίους αριθμούς ενώ ο Β αριθμούς σταθερού σημείου οι οποίοι έχουν την παράσταση που φαίνεται στο προηγούμενο σχήμα Έτσι για τον υπολογιστή Α η παράσταση 01111111 ερμηνεύεται σαν +12710 ενώ για τον Β είναι ο +7.937510. Οι δύο αριθμοί σχετίζονται μεταξύ τους με τη βοήθεια του συντελεστή κλίμακα 2^{-4}. Πράγματι $127*2^{-4}$ = 7.9375. Ο εκθέτης του συντελεστή κλίμακας (Scale Factore) είναι ίσος με το πλήθος των bits του κλασματικού μέρους. Βέβαια ο συντελεστής μπορεί να είναι 2^{-4} ή 2^4 και αυτό εξαρτάται από το σύστημα αναχώρησης.

Η πρόσθεση και η αφαίρεση γίνονται με τους συνηθισμένους κανόνες δυαδικής πρόσθεσης και αφαίρεσης λαμβάνοντας υπόψη τον συντελεστή κλίμακας.

Αν παραστήσουμε με Κ και L δύο δυαδικές παραστάσεις των 8 bit. Η πρόσθεση στο Α συνεπάγεται την κανονική πρόσθεση στο σύστημα πρόσημο-μέγεθος δηλαδή (K+L). Η πρόσθεση στον Β συνεπάγεται την πράξη: (K+L) x 2^{-4} Ανάλογες προσαρμογές γίνονται και για τον πολ/σμό και την διαίρεση.

Ένα από τα κύρια μειονεκτήματα αυτού το συστήματος είναι το περιορισμένο εύρος των αριθμών που μπορούν να κωδικοποιήσουν γεγονός που αναγκάζει τους προγραμματιστές να χρησιμοποιούν συντελεστές κλίμακας.

Παράσταση αριθμών κινητού σημείου

Το σύστημα παράστασης αριθμών κινητού σημείου (Floating point represantation) χρησιμοποιείται συνήθως για υπολογισμούς που απαιτούν μεγάλες ακρίβειες. Απαλλάσσει τον προγραμματιστή από τον πονοκέφαλο της κλίμακας και μπορεί ν' αποθηκεύσει μεγάλους αριθμούς σε σχετικά λίγα bits. Επειδή στο σύστημα αυτό η θέση του δυαδικού σημείου μετακινείται με την ευθύνη του Η/Υ ονομάζεται σύστημα παράστασης κινητού σημείου. Όπως είπαμε και πιο πριν το μεγάλο πλεονέκτημα αυτής της μεθόδου είναι ότι μια λέξη μήκους η bit μπορεί να χωρέσει αριθμούς πολύ μεγαλύτερους από $2^{η}$-1. Το μειονέκτημα της όμως είναι ότι δεν μπορούμε να έχουμε η-1 σημαντικά bits, όπως στο σύστημα παράστασης των ακεραίων Σήμερα σε όλους τους υπολογιστές, οι πραγματικοί (ρητοί) αριθμοί παριστάνονται με το σύστημα κινητού σημείου.

Ενας δυαδικός αριθμός κινητού σημείου, R παριστάνεται από μια αλυσίδα από bits και χαρακτηρίζεται από τις εξής παραμέτρους:

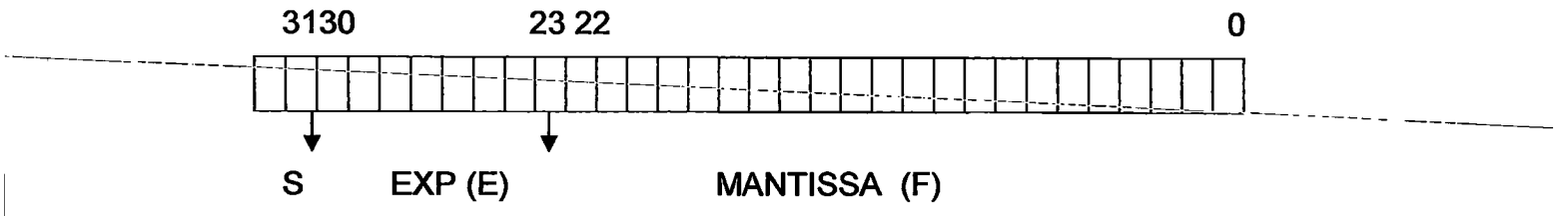
Α)- από το πρόσημο (sign) S του αριθμού.

Β) - από έναν εκθέτη (exponent) Ε μιας βάσης Z, και

Γ)- από ένα συντελεστή (mantissa) F, έτσι ώστε να ισχύει η σχέση:

$$R = (-1)S \times Z^{E} \times F \qquad (1)$$

Η θέση μνήμης του Η/Υ που θ' αποθηκεύσει τον αριθμό χρειάζεται να έχει τρεις χώρους. Ένα για το πρόσημο S, ένα για τον εκθέτη Ε (m bits) και έναν για τον συντελεστή F (κ bits), έτσι ώστε n=1+M+K. Η βάση Z είναι ίδια για όλους τους αριθμούς και δεν χρειάζεται να έχουμε ιδιαίτερο χώρο γι' αυτήν και ονομάζεται βάση του συστήματος κινητής υποδιαστολής.



Πρότυπο κινητού σημείου της IEEE

Όσο μεγαλύτερο είναι το K, τόσο περισσότερο αυξάνεται η ακρίβεια του αριθμού, αντίθετα όσο αυξάνει το M τόσο αυξάνει το μέγεθος του αριθμού. Υπάρχουν πολλά σχήματα παράστασης αριθμών κινητού σημείου. Εμείς θα περιγράψουμε αυτό που προτείνεται σαν πρότυπο από την IEEE (προηγούμενο σχήμα). Η IEEE (Institute of Electrical and Electronics Engineers) είναι ένα διεθνές επιστημονικό ινστιτούτο το οποίο πέραν των άλλων έχει σαν στόχο την κατασκευή διεθνών προτύπων. Σαν παράδειγμα μπορούμε να αναφέρουμε το πρωτόκολλο για δίκτυα 10 Mbps Ethernet που φέρει την κωδικοί ονομασία IEEE 802.3.

Στο σύστημα αυτό ένας απλής ακρίβειας αριθμός κινητού σημείου R, ο οποίος είναι κανονικοποιημένος, έχει τη μορφή:

$$R = (-1)S \times 2^{e-127} \times (1.f) \qquad (2)$$

όπου S είναι το πρόσημο του αριθμού (1 bit)

e είναι ο εκθέτης που καταλαμβάνει χώρο 8 bits

f ένα δυαδικό κλάσμα(fraction ή mantissa) 3 bits

Αν συνδυάσουμε τις σχέσεις (1) και (2) έχουμε :

$$E = e - 127, \quad F = (1.f) \text{ και } \quad Z = 2.$$

Σχετικά με τις τρεις παραπάνω σχέσεις έχουμε να κάνουμε τις παρακάτω παρατηρήσεις:

α) Παρατηρούμε ότι δεν υπάρχει θέση για το πρόσημο του εκθέτη. Η έλλειψη αρνητικού εκθέτη σημαίνει ότι δεν μπορούν να παρασταθούν πολύ μικροί

αριθμοί. Η έλλειψη αυτή αντιμετωπίζεται μ' ένα τέχνασμα. Στον τύπο (1) αντί του εκθέτη E τοποθετείται ο πολωμένος (biased Exponent) εκθέτης e, έτσι ώστε E = e - 127

Η τιμή 127 ονομάζεται πόλωση (bias). Επειδή ισχύει $0 \leq e \leq 255$ θα έχουμε $-127 \leq e - 127 \leq 128$. Το τέχνασμα αυτό άλλαξε τα μεγέθη των αριθμών που μπορούμε να κωδικοποιήσουμε. Χωρίς την πόλωση θα είχαμε αριθμούς με μεγέθη από 0 έως 2^{255} ενώ με την προσθήκη της πόλωσης μπορούμε να κωδικοποιήσουμε αριθμούς με μεγέθη που περιλαμβάνονται μεταξύ των ορίων 2^{-127} και 2^{128}.

β) Επειδή υπάρχουν πολλοί συνδυασμοί των E και F που δίνουν το ίδιο R, θεωρούμε πάντα ότι ο αριθμός R είναι *κανονικοποιημένος* (normalized) ως προς τη βάση Z. Ένας αριθμός ονομάζεται *κανονικοποιημένος* όταν το MSD ως προς βάση Z είναι διάφορο του μηδενός. Για το πρότυπο της IEEE δίνεται ο παρακάτω ορισμός για ένα κανονικοποιημένο αριθμό.

"Ένας αριθμός R θα λέγεται *κανονικοποιημένος* ως προς βάση 2, όταν μπορεί να γραφεί με τη μορφή $R = (-1)S \times 2^{e-127} \times F$, όπου F είναι ένας δυαδικός αριθμός με ένα ακέραιο bit ίσο με την μονάδα, δηλαδή έχουμε τη σχέση F = 1.f"

Από τα παραπάνω προκύπτει ότι $1 \leq 1.f < 2$. Πράγματι, η μικρότερη τιμή του f είναι μηδέν και του 1.f είναι 1. Η μεγαλύτερη τιμή του f είναι $.111...111_2$(23 bits) και του (1. f) = $1.99999999..._{10}$

Σημειώνουμε επίσης ότι η μονάδα αριστερά του δυαδικού σημείου στον παράγοντα (1.f) δεν καταλαμβάνει θέση μέσα στα 32 bit που κωδικοποιείται ο αριθμός R.

Ο πλήρης ορισμός του πρότύπου της IEEE για τα 32 bits είναι :

αν	e = 255	και F <> 0	τότε "κανένας αριθμός"
αν	e = 255	και F = 0	τότε "άπειρο"
αν	0 < e < 255		τότε $R = (-1)S \times 2^{e-127} \times (1.f)$
αν	e = 0	και F <> 0	τότε $R = (-1)S \times 2^{-126} \times (0.f)$
αν	e = 0	και F = 0	τότε R = 0

Θα υπολογίσουμε τώρα τον μεγαλύτερο (Rmax) και τον μικρότερο (Rmin) που μπορούμε να έχουμε στο σύστημα IEEE. Θα γίνει υπολογισμός των απολύτων τιμών.

Υπολογισμός Rmax:

$$Rmax = 2^{Emax} \times Fmax$$

$$E = -126$$

$$Fmin = (1.f)min = 1.0$$

$$Rmin = 2^{-126} \times 1.0 \cong 2^{-126} \cong 1.17 \times 10^{-38}$$

Ο υπολογισμός της πόλωσης εξαρτάται από το μήκος σε bits του χώρου του εκθέτη. Αν ο εκθέτης e έχει μήκος K bits τότε $e_{min} = 0$ και $e_{max} = 2K-1$. Ο

αριθμός που πολώνει τον εκθέτη (bias) θα είναι ο $2^{K-1}-1$ και ο εκθέτης e $-(2^{K-1}-1)$ θα έχει μέγιστο 2^{K-1} και ελάχιστο $-(2^{K-1}-1)$.

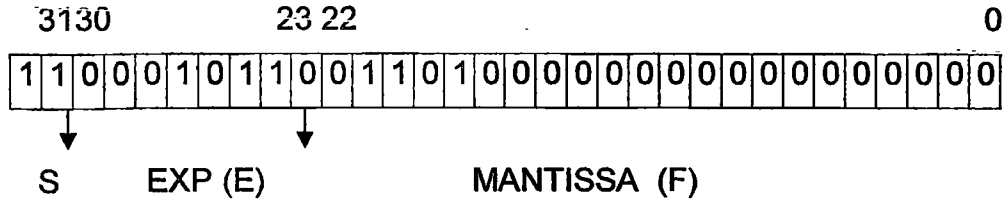
Η αποκωδικοποίηση του περιεχομένου μιας θέσης μνήμης που φιλοξενεί έναν αριθμό κινητής υποδιαστολής R γίνεται ως εξής:

Αλγόριθμος Αποκωδικοποίησης

1 : Μετατρέπουμε τις ποσότητες e και f στο δεκαδικό σύστημα
2 : Υπολογίζουμε τον πολωμένο εκθέτη E10=e10-127.
3 : Υπολογίζουμε τον αριθμό R με βάση το πρότυπο IEEE.

Παραδείγματα :

α) Αν υποθέσουμε ότι μια θέση μνήμης έχει την παρακάτω εικόνα:



θα εφαρμόσουμε τον αλγόριθμο αποκωδικοποίησης για να βρούμε το δεκαδικό της αντίστοιχο.

S = 1

e = 10001011_2 και f=0.001101_2

1 : e_{10} = 139_{10}

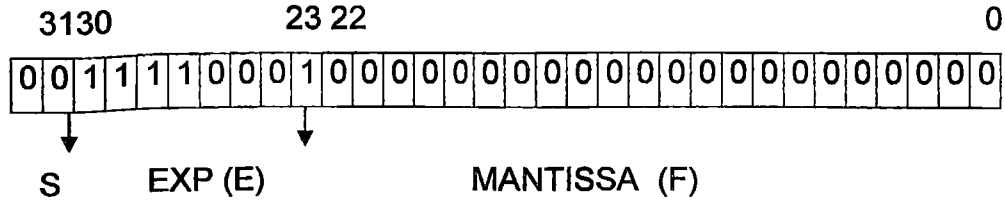
f_{10} = 0.203125^{10}

2 : E_{10} = 139 - 127 = 12_{10}

3 : R = $(-1)1 \times 2^{12} \times 1.203125$ =

= $(-1) \times 4096 \times 1.203125 = -4928$

β) Να γίνει η αποκωδικοποίηση της παρακάτω θέση μνήμης.



S = 0

e = 01111000_2 και f = 0.1_2

1 : e_{10} = 120_{10}

$f_{10} = 0.5_{10}$

2 : $E_{10} = 120 - 127 = -7_{10}$

3 : $R = (-1)^0 \times 2^{-7} \times 1.5 = +.011718750$

Η κωδικοποίηση ακολουθεί τον αντίθετο δρόμο με το πρόσθετο βήμα της κανονικοποίησης του αριθμού. Ένας αριθμός κινητού σημείου R κανονικοποιείται ως εξής: Μετακινούμε τα bit του παράγοντα F έτσι ώστε να τον φέρουμε στη μορφή (1.f). Αν μετακινήσουμε τα ψηφία του F κατά K θέσεις δεξιά μειώνουμε αντίστοιχα κατά K τον εκθέτη E. Η προς τ' αριστερά μετακίνηση σημαίνει αντίστοιχα αύξηση του εκθέτη.

Υποθέτουμε ότι έχουμε τον αριθμό R_{10} και θέλουμε να τον κωδικοποιήσουμε. Η κωδικοποίηση λοιπόν ακολουθεί τα παρακάτω βήματα.

Αλγόριθμος κωδικοποίησης

1 : Γράφουμε τον αριθμό $R_{10} = 2^0 \times F_{10}$.
2 : Μεταφέρουμε το F_{10} στο δυαδικό σύστημα (F_2).
3 : Κανονικοποιούμε την ποσότητα $2^0 \times F_2$ και την φέρνουμε στην μορφή $2^E{}_{10} \times (1.f)_2$.
4 : Επηρεάζουμε τον εκθέτη E_{10} προσθέτοντας το 127_{10}. Δηλαδή, $e_{10} = E_{10} + 127_{10}$
5 : Μεταφέρουμε τον e_{10} στο δυαδικό (e_2).
6 : Αν R είναι Θετικός τότε S=0, αν αρνητικός S=1.
7 : Τοποθετούμε το S στο bit 31.
Τοποθετούμε το e_2 στα bits 30 ως 23.
Τοποθετούμε το f_2 στα bits 22 ως 0.

Στο σημείο αυτό θα θέλαμε να σημειώσουμε ότι οι δύο αλγόριθμοι που παρουσιάσαμε υλοποιούν την περίπτωση $0 < e < 255$ της IEEE και δεν περιέχουν τις ακραίες τιμές 0 και 255. Όπως φαίνεται από τον ορισμό της IEEE αυτές οι τιμές χρησιμοποιούνται για την παράσταση κάποιων ειδικών καταστάσεων. (π.χ άπειρο, τίποτα, μηδέν)

Παραδείγματα :

α) Να κωδικοποιηθεί ο αριθμός -53851_{10}.

B1 : $(1.f) = 53851 = 2^0 \times 53851_{10}$

B2 : $(1.f) = 53851_{10} = 1101001001011011_2$

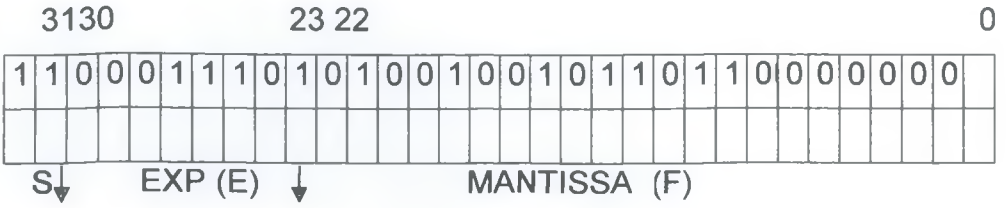
B3 : $(1.f)_2 = 2^0 \times 1101001001011011. = 2^{15} \times 1.101001001011011$

B4 : $e_{10} = 15 + 127 = 142_{10}$

B5 : $e_2 = 10001110_2$

B6 : S=1

B7 :



β) Να κωδικοποιηθεί ο αριθμός -0.3_{10}

1 : $(1.f) = 0.3_{10} = 2^0 \times 0.3_{10}$

2 : $(1.f) = 0.3_{10} = 0.01001100110011001100110011011001\ldots._2$

3 : $(1.f)_2 = 2^0 \times 0.01001100110011001100110011011001\ldots._2 =$

$2^{-2} \times 1.00110011001100110011001\ 1011001_2$

4 : $e_{10} = -2 + 127 = 125_{10}$ out

5: $e_2 = 01111101_2$

6: S=1

7 :

Αν τώρα προσπαθήσουμε ν' αποκωδικοποιήσουμε το περιεχόμενο της παραπάνω θέσης μνήμης θα βρούμε:

S = 1

$e = 01111101_2$ $f = .00110011001100110011001_2$

1 : $e_{10} = 125_{10}$

$f_{10} = 0.1999999910$

2 : $E_{10} = 125 - 127 = -2_{10}$

3 : $R = (-1)1 \times 2^{-2} \times 1.1999999 =$

$= -0.29999999821186066_{10}$

Παρατηρούμε λοιπόν ότι ενώ δώσαμε στον Η/Υ τον αριθμό -0.3 στην πραγματικότητα αποθηκεύθηκε ο αριθμός -0.2999999. Το σφάλμα που προκύπτει μεγαλώνει ή μικραίνει ανάλογα με το μήκος της mantissa. Αυτό οφείλεται στο γεγονός ότι αναγκαστήκαμε να κόψουμε τα bit του $(0.3)_{10}$, πέρα από το 25ο (μετά φυσικά από την κανονικοποίηση). Στο σημείο αυτό μας δίνεται η ευκαιρία να τονίσουμε ότι η κανονικοποίηση αυξάνει την ακρίβεια του συστήματος παράστασης.

Μήκος Mantissa i	2^{-i}	Άθροισμα όλων των 2-i	Ακρίβεια σε ψηφία
1	.5	.5 0	
2	.25	.750	
3	.125	.875 0	
4	.0625	.9375 1	
5	.03125	.96875 1	
6	.015625	.984375 1	
7	.0078125	.9921875 2	
8	.00390625	.99609375 2	
9	.001953125	.998046875 2	
10	.0009765625	.9990234375 3	
11	.00048828125	.99951171875 3	
12	.000244140625	.999755859375 3	
13	.0001220703125	.9998779296875 3	
14	.00006103515625	.99993896484375 4	
15	.000030517578125	.999969482421875 4	
16	.0000152587890625	.9999847412109375 4	
17	7.62939453125D-06	.9999923706054688 5	
18	3.814697265625D-06	.99999618530273445	
19	1.9073486328125D-06	.9999980926513672 5	
20	9.5367431640625D-07	.9999990463256836 6	
21	4.76837158203125D-07	.9999995231628418 6	
22	2.384185791015625D-07	.9999997615814209 6	
23	1.192092895507813D-07	.9999998807907105 6	
	→ Διπλή ακρίβεια ←		
24	5.960464477539063D-08	.9999999403953552 7	
25	2.980232238769531D-08	.9999999701976776 7	
26	1.490116119384766D-08	.9999999850988388 7	
27	7.450580596923828D-09	.9999999925494194 8	
28	3.725290298461914D-09	.9999999962747097 8	
29	1.862645149230957D-09	.9999999981373549 8	
30	9.313225746154785D-10	.9999999990686774 9	
31	4.656612873077393D-10	.9999999995343387 9	
32	2.328306436538696D-10	.9999999997671694 9	
33	1.164153218269348D-10	.9999999998835847 9	
34	5.820766091346741D-11	.9999999999417924 10	
35	2.91038304567337D-11	.9999999999708962 10	
36	1.455191522836685D-11	.9999999999854481 10	
37	7.275957614183426D-12	.9999999999927241 11	
38	3.637978807091713D-12	.9999999999963612 11	
39	1.818989403545856D-12	.9999999999981811 11	
40	9.094947017729283D-13	.9999999999990905 12	
41	4.547473508864641D-13	.9999999999995453 12	
42	2.273736754432321D-13	.9999999999997726 12	

43	1.13686837721616D-13	.9999999999998863	12
44	5.684341886080802D-14	.9999999999999432	13
45	2.842170943040401D-14	.9999999999999716	13
46	1.4210854715202D-14	.9999999999999858	13
47	7.105427357601002D-15	.9999999999999929	14
48	3.552713678800501D-15	.9999999999999965	14
49	1.77635683940025D-15	. 9999999999999982	14
50	8.881784197001252D-16	.9999999999999991	15
51	4.440892098500626D-16	.9999999999999996	15
52	2.220446049250313D-16	.9999999999999998	15

Αντιστοιχία μήκους Mantissa και ακρίβειας (IEEE)

Ο Παραπάνω πίνακας σχηματίστηκε ως εξής : Για να βρούμε την μεγίστη τιμή της mantissa για μήκος i=7 υπολογίζομε το άθροισμα

Mantissa(i=7) = $2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7}$ =

= 0.5 + 0.25 + 0.125 + 0.0625 + 0.03125 + 0.015625 + 0.0078125 = 0.9921875

Αυτό σημαίνει ότι mantissa με μήκος 7 μπορεί να δώσει 2 με 3 ψηφία ακρίβεια.

Όπως είπαμε και πιο πριν το μήκος σε bits της mantissa επηρεάζει το πλήθος των ψηφίων του κλασματικού μέρους του αριθμού που μπορούμε να έχουμε εμπιστοσύνη.

παρουσιάζουμε αυτά τα ψηφία (ακρίβεια) για μήκη από 1 έως 52.

Άσκηση :

Ας υποθέσουμε ότι έχουμε έναν υπολογιστή 16 bit. Η παράσταση ενός αριθμού κινητού σημείου ακολουθεί το IEEE πρότυπο με μια θέση για το πρόσημο, 5 bits για τον επηρεασμένο εκθέτη και 10 bits για την mantissa. Υπολογίστε τον μεγαλύτερο και τον μικρότερο αριθμό που μπορούμε να κωδικοποιήσουμε.

Η πόλωση θα έχει τιμή $2^{5-1}-1$ Άρα ή σχέση μεταξύ E και e γίνεται E = e -(2^{5-1}-1) = e - 15. Από τη σχέση E = e-15 θα υπολογίσουμε το Emax και το Emin. Πράγματι,

Emax = emax -15 = 25-1 - 15 = 16_{10}

Emin = emin -15 = 0 -15 =-15_{10}

Θα υπολογίσουμε τώρα το Fmax και το Fmin .

Fmax = (1.f)max = 1.1111111111_2 = 1.9990234375_{10}

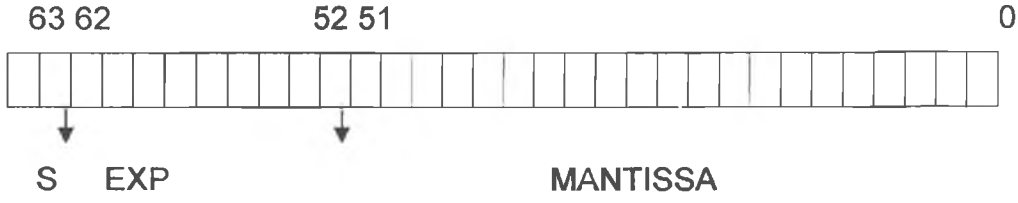
Fmin = (1.f)min = 1.010

'Αρα, και αν λάβουμε υπόψη μας ότι e=0 και e = 16 δεν χρησιμοποιούνται, τότε

Rmax = 2^{Emax} x Fmax = 2^{15} x 1.9990234375 = 65504_{10}

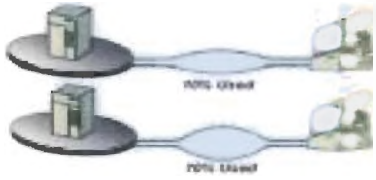
$$Rmin = 2^{Emin} \times Fmin = 2^{-14} \times 1.0 = 1.5125 \times 10^{-5}$$

Το πρότυπο της IEEE επεκτείνεται και σε διπλή θέση μνήμης που μπορεί να φιλοξενήσει αριθμούς με μεγαλύτερο εύρος και με περισσότερα σημαντικά ψηφία. Οι αριθμοί αυτοί είναι γνωστοί σαν αριθμοί διπλής ακρίβειας (double precision). Το αντίστοιχο πρότυπο της IEEE δίνεται στο επόμενο σχήμα



Στη διπλή ακρίβεια έχουμε μήκος για τον εκθέτη 11 bits , και για τον συντελεστή 52 bits. Η πόλωση του εκθέτη είναι 1023 (γιατί;) και η πλήρης περιγραφή δίνεται από τον παρακάτω πίνακα

αν	e = 2047 και F <> 0	τότε	"κανένας αριθμός"
αν	e = 2047 και F = 0	τότε	"άπειρο"
αν	0 < e < 2047	τότε	$R = (-1)S \times 2^{e-1023} \times (1.f)$
αν	e = 0 και F <> 0	τότε	$R = (-1)S \times 2^{-1023} \times (0.f)$
αν	e = 0 και F = 0	τότε	R = 0



3.2.8. Κώδικες χαρακτήρων

Όπως ήδη έχουμε αναφέρει και πιο πριν, οι υπολογιστές χρησιμοποιούνται και για τη επεξεργασία πληροφοριών που δεν έχουν αριθμητική μορφή. Σαν παράδειγμα αναφέρουμε την ταξινόμηση και την επεξεργασία κειμένων, την ανάλυση των εντολών από τους μεταφραστές, την ανάκτηση πληροφοριών με βάση τη σύγκριση με μια λέξη κλειδί, κ.λ.π. Όλα τα προηγούμενα αναφέρονται σε επεξεργασία γραμμάτων, αριθμών, σημείων στίξης και ειδικών συμβόλων.

Όλα αυτά τα σύμβολα είναι κατανοητά από τον άνθρωπο, όχι όμως και από τους υπολογιστές. Για να γίνουν κατανοητές όλες αυτές οι παραστάσεις πρέπει να αντικατασταθούν με τα αριθμητικά τους ισοδύναμα. Για τον σκοπό αυτό υπάρχουν οι λεγόμενοι κώδικες χαρακτήρων, οι οποίοι δεν είναι τίποτε άλλο παρά πίνακες όπου σε κάθε σύμβολο ν' αντιστοιχεί και ένας δυαδικός αριθμός.

Ονομάζουμε σύνολο χαρακτήρων (character set) το πλήθος των διαφορετικών χαρακτήρων που αντιλαμβάνεται ένας υπολογιστής. Το πρώτο σύνολο χαρακτήρων που χρησιμοποιήθηκε περιελάμβανε 64 διαφορετικούς

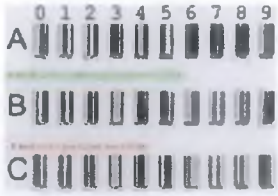
χαρακτήρες. Από αυτούς, 26 ήταν τα γράμματα του λατινικού αλφάβητου, 10 τα ψηφία του δεκαδικού συστήματος , και 28 ειδικά σύμβολα, όπως (* / + - @ % #), σημεία στίξης (. , : ; ! ?) . Για τον κώδικα αυτό χρειαζόμαστε 6 bits για να κωδικοποιήσουμε τους 64 χαρακτήρες. Τα πρώτα χρόνια, τα 6 αυτά bits ήταν αρκετά για να κωδικοποιήσουν τους αναγκαίους για την εποχή χαρακτήρες. Μη ξεχνάτε ότι η χρήση των υπολογιστών γινόταν μόνο για υπολογισμούς και οι εκτυπώσεις ήταν με κεφαλαία και κακής ποιότητας. Ο κώδικας αυτός ονομάστηκε BCD εξ' αιτίας του γεγονότος ότι αποτελούσε επέκταση του 4 bit BCD κώδικα.

Παρ' όλη την ευρεία χρήση του, ο BCD κώδικας ήταν ανεπαρκής για πολλές εφαρμογές. Για παράδειγμα, δεν υπήρχε δυνατότητα για ταυτόχρονη χρήση μικρών και κεφαλαίων, απουσίαζαν τα ελληνικά σύμβολα, για να αναφέρουμε μερικούς από τους περιορισμούς. Για τους παραπάνω λόγους κατασκευάστηκαν δύο διαφορετικοί κώδικες των 8 bits. Ο πρώτος, κατασκευάστηκε για να χρησιμοποιηθεί σαν πρότυπο για την μεταφορά δεδομένων και ονομάζεται ASCII (American Standards Committiee on Information Interchange). Υιοθετήθηκε απ' όλους σχεδόν τους κατασκευαστές υπολογιστών. Η IBM όμως ακολούθησε δικό της κώδικα και κατασκεύασε τον EBCDIC (Extended Binary Coded Decimal Interchange Code). Ο επόμενος πίνακας παρουσιάζει τμήματα των 8 bits κωδίκων και περιλαμβάνει τους κοινούς χαρακτήρες των τριών κωδίκων. Ο κώδικας ASCII παρουσιάστηκε αρχικά σαν ένας κώδικας με 7 bits και μπορούσε να κωδικοποίηση 27 = 127 διαφορετικούς χαρακτήρες. Αργότερα επεκτάθηκε στα 8 bits και περιλαμβάνει 256 χαρακτήρες

Χαρακτήρες	BCD	EBCDIC	ASCII
Blank	110 000	0100 0000	0010 0000
.	011 011	0100 1011	0010 1110
(	111 100	0100 1101	0010 1000
+	010 000	0100 1110	0010 1011
$	101 011	0101 1011	0010 0100
*	101 100	0101 1100	0010 1010
)	011 100	0101 1101	0010 1001
-	100 000	0110 0000	0010 1101
/	110 001	0110 0001	0010 1111
'	111 011	0110 1011	0010 1100
,	001 100	0111 1101	0010 0111
=	001 011	0111 1110	0011 1101
A	010 001	1100 0001	0100 0001
B	010 010	1100 0010	0100 0010
C	010 011	1100 0011	0100 0011
D	010 100	1100 0100	0100 0100
E	010 101	1100 0101	0100 0101
F	010 110	1100 0110	0100 0110
G	010 111	1100 0111	0100 0111
H	011 000	1100 1000	0100 1000

I	011 001	1100 1001	0100 1001
J	100 001	1101 0001	0100 1010
K	100 010	1101 0010	0100 1011
L	100 011	1101 0011	0100 1100
M	100 100	1101 0100	0100 1101
N	100 101	1101 0101	0100 1110
O	100 110	1101 0110	0100 1111
P	100 111	1101 0111	0101 0000
Q	101 000	1101 1000	0101 0001
R	101 001	1101 1001	0101 0010
S	110 010	1110 0010	0101 0011
T	110 011	1110 0011	0101 0100
U	110 100	1110 0100	0101 0101
V	110 101	1110 0101	0101 0110
W	110 110	1110 0110	0101 0111
X	110 111	1110 0111	0101 1000
Y	111 000	1110 1000	0101 1001
Z	111 001	1110 1001	0101 1010
0	000 000	1111 0000	0011 0000
1	000 001	1111 0001	0011 0001
	2000 010	1111 0010	0011 0010
	3000 011	1111 0011	0011 0011
	4000 100	1111 0100	0011 0100
	5000 101	1111 0101	0011 0101
	6000 110	1111 0110	0011 0110
	7000 111	1111 0111	0011 0111
	8001 000	1111 1000	0011 1000
	9001 001	1111 1001	0011 1001

Κώδικες



Όσο αφορά τους ελληνικούς χαρακτήρες αυτοί έχουν κωδικούς μεγαλύτερους από 128.

Στον χώρο αυτό δηλαδή από 128 μέχρι 255 η διεθνής τυποποίηση τοποθετεί τα εθνικά αλφάβητα

Για τα Ελληνικά υπάρχουν οι τυποποιήσεις MS DOS CODEPAGE 737 και

ISO/IEC 8859-7 for windows αυτό είναι γνωστά και σαν ΕΛΟΤ –928.

Ο ΕΛΟΤ είναι ο Ελληνικός Οργανισμός Τυποποίησης. Για περισσότερες πληροφορίες στο Παράρτημα ΙΙ

Η διάταξη των χαρακτήρων μέσα στους κώδικες γίνεται έτσι ώστε να διευκολύνεται η σύγκριση αλυσίδων χαρακτήρων. Για παράδειγμα, το αριθμητικό ισοδύναμο του χαρακτήρα "A" είναι πάντοτε μικρότερο από αυτό του χαρακτήρα "B".

Δεκαδικός κωδικός	Πλήκτρο ελέγχου	όνομα	περιγραφή
0	^@	NUL	μηδενικός χαρακτήρας
1	^A	SOH	αρχή επικεφαλίδας
2	^B	STX	αρχή κειμένου
3	^C	ETX	τέλος κειμένου
4	^D	EOT	τέλος μετάδοσης
5	^E	ENQ	αίτηση
6	^F	ACK	επικύρωση
7	^G	BEL	κουδούνι
8	^H	BS	οπισθοχώρηση
9	^I	HT	οριζόντιο TAB
10	^J	LF	αλλαγή γραμμής
11	^K	VT	κατακόρυφο TAB
12	^L	FF	αλλαγή σελίδας
13	^M	CR	επαναφορά κεφαλής
14	^N	SO	ολίσθηση εκτός
15	^O	SI	ολίσθηση εντός
16	^P	DEL	διαγραφή
17	^Q	DC1	έλεγχος συσκευής 1
18	^R	DC2	έλεγχος συσκευής 2
19	^S	DC3	έλεγχος συσκευής 3
20	^T	DC4	έλεγχος συσκευής 4
21	^U	NAK	αρνητική επικύρωση
22	^V	SYN	συγχρονισμός
23	^W	ETB	τέλος τμήματος κειμένου
24	^X	CAN	ακύρωση
25	^Y	EM	τέλος μέσου
26	^Z	SUB	αντικατάσταση
27	^[	ESC	διαφυγή
28	^/	FS	διαχωριστής αρχείων
29	^]	GS	διαχωριστής ομάδων
30	^^	RS	διαχωριστής εγγράφων
31	^_	US	διαχωριστής μονάδων

Χαρακτήρες ελέγχου ASCII

Οι κωδικοί πού βρίσκονται από τη θέση 0 μέχρι 31 ονομάζονται χαρακτήρες ελέγχου,δεν μπορούν να εκτυπωθούν και είναι κάτι το πολύ διαφορετικό από αυτό που φαίνεται στον παραπάνω πίνακα. Δεν αντιστοιχούν σε χαρακτήρες αλλά σε συγκεκριμένες ενέργειες. Για παράδειγμα όταν στέλνούμε πληροφορίες στον εκτυπωτή μας πρέπει να του στείλουμε πληροφορίες που αφορούν τον τρόπο εκτύπωσης (αλλαγή σελίδας, αλλαγή γραμμής, επαναφορά κεφαλής, κ.λπ.). Οι κανονικοί χαρακτήρες αφορούν "αυτό που θα εκτυπωθεί", ενώ οι χαρακτήρες ελέγχου αφορούν "το πώς θα εκτυπωθεί". Σε μερικούς χαρακτήρες ελέγχου αντιστοιχούν ειδικά πλήκτρα στο πληκτρολόγιο,

όπως το ESC, DEL, το TAB για το HT, το ENTER για το CR το <— για το BS (Back Space) κ.λπ. Στον πίνακα το σύμβολο (^) σημαίνει Ctrl.

3.2.9. Κώδικες ανίχνευσης και διόρθωσης λάθους

Οι μονάδες που συνθέτουν έναν υπολογιστή επικοινωνούν μεταξύ τους μέσω διαδρόμων. Οι υπολογιστές επικοινωνούν μεταξύ τους μέσω δικτύων. Ο φυσικός φορέας επικοινωνίας μπορεί να είναι χαλκός, οπτική ίνα, ή αέρας (ασύρματη επικοινωνία). Πολλοί είναι οι παράγοντές που επηρεάζουν την ασφαλή μεταφορά των δεδομένων(παράσιτα, ηλεκτρομαγνητικά πεδία, ατμοσφαιρικές συνθήκες, βλάβες στις συσκευές κ.λπ). Για να κάνουμε πιο κατανοητές τις επόμενες έννοιες θα αρκεστούμε στην επικοινωνία του υπολογιστή μας με μερικές περιφερειακές συσκευές όπως ο εκτυπωτής, το modem, scanner κ.λπ. Η επικοινωνία γίνεται με τρεις τρόπους. Σειριακή, παράλληλη και USB (Universal Serial Bus). Για τον σκοπό αυτό οι υπολογιστές διαθέτουν αντίστοιχες πόρτες σύνδεσης. Στους προσωπικούς υπολογιστές έχουμε συνήθως δύο σειριακές, μια παράλληλη και μια η δύο USB.

Στην σειριακή επικοινωνία έχουμε μεταφορά ενός bit την φορά. Διαδικασία επιτυγχάνεται με δύο καλώδια. Ένα για την αποστολή και ένα για την λήψη δεδομένων. Υπάρχουν και άλλα επτά βοηθητικά καλώδια τα οποία ρυθμίζουν την αποστολή και λήψη των δεδομένων και μεταφέρουν εντολές ελέγχου. Οι ακροδέκτες ακολουθούν το πρότυπο RS-232 και υπάρχουν σε διατάξεις των 9 και 25 pins (ακίδες) αντίστοιχα. Το πρότυπο RS-232 αφορά τις φυσικές συνδέσεις των ακροδεκτών.

Στην παράλληλη επικοινωνία έχουμε την μεταφορά ενός ολοκλήρου byte και για την σύνδεση των ακροδεκτών ακολουθείται το πρότυπο centronics.

Οι αριθμητικοί ή κώδικες χαρακτήρων πού παρουσιάσαμε στις προηγούμενες παραγράφους δεν εξασφαλίζουν την αλάνθαστή μεταφορά των δεδομένων μεταξύ των μονάδων του υπολογιστή. Το ίδιο ισχύει προφανώς για κάθε μεταφορά πληροφορίας μεταξύ ενός δέκτη και ενός πομπού. Πάντοτε υπάρχει η πιθανότητα να παρουσιαστεί ένα λάθος σε μια θέση (bit posision) κατά τη μεταφορά μιας ομάδος από δυαδικές πληροφορίες. Το λάθος σε μια μόνο θέση (single bit error) παρουσιάζεται πιο συχνά από τα πολλαπλά λάθη και για τον σκοπό αυτό θα ασχοληθούμε μόνο με τους κώδικες ενός λάθους. Οι κώδικες ανίχνευσης πολλαπλών λαθών είναι πολύπλοκοι και εξαιρετικά χρονοβόροι και για αυτό χρησιμοποιούνται σε κρίσιμες περιπτώσεις. Η ανάγκη όμως για αξιοπιστία στη μεταφορά και την επεξεργασία δεδομένων οδήγησε στην κατασκευή κωδίκων που έχουν τη δυνατότητα να ανιχνεύουν ένα λάθος και κωδίκων που να το διορθώνουν. Αυτοί οι κώδικες περιλαμβάνουν δύο ειδών πληροφορίες. Τα δεδομένα που μεταφέρονται (κωδικοποιημένα σε κάποιο γνωστό κώδικα) και τα ψηφία ελέγχου. Ο μηχανισμός λειτουργεί ως εξής: Ο δεκτής στέλνει την κυρίως πληροφορία πλαισιωμένη από τα ψηφία ελέγχου. Η τιμή των ψηφίων ελέγχου δίνεται από τον δέκτη με βάση ένα αλγόριθμο που χαρακτηρίζει τον κώδικα. Ο δέκτης λαμβάνει και τις δύο πληροφορίες (κυρίως πληροφορία και ψηφία ελέγχου) εφαρμόζει εκ νέου τον αλγόριθμο και αν υπάρχει ένα λάθος το ανιχνεύει. Αμέσως μετά μπαίνει σε λειτουργία, αν υπάρχει ,ο μηχανισμός διόρθωσης του

5 μονάδες 6 μονάδες

Ανίχνευση ενός λάθους με κώδικα περιττής ισοτιμίας

Όπως είπαμε και στην αρχή ο παραπάνω κώδικας ανιχνεύει μόνο ένα λάθος. Αν κατά μεταφορά αλλοιωθούν δυο bits τότε ο κώδικας αυτός δεν μπορεί να τα ανιχνεύσει. Αντίστοιχος με τον περιττό κώδικα ισοτιμίας είναι και ο άρτιος κώδικας ισοτιμίας (even parity code).

Κώδικες διόρθωσης λάθους

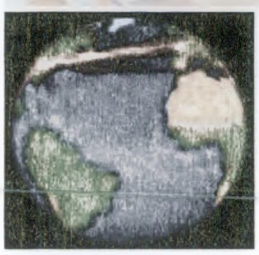
Όπως και στη προηγούμενη παράγραφο έτσι και σ' αυτήν θα ασχοληθούμε με τους κώδικες διόρθωσης ενός λάθους. Οι πιο δημοφιλείς από τους κώδικες αυτούς είναι ο κώδικας ομαδοποιημένης ισοτιμίας και ο κώδικας Hamming.

Κώδικας ομαδοποιημένης ισοτιμίας: Ο κώδικας ισοτιμίας που εξετάστηκε προηγουμένως απαιτεί ένα επί πλέον bit για κάθε λέξη ή byte που μεταφέρεται. Αν τώρα θέλουμε να στείλουμε ένα πακέτο από Μ λέξεις των Ν bits αυτές σχηματίζουν ένα πίνακα με Μ γραμμές και Ν στήλες. Κάθε μια από τής γραμμές διαθέτει ένα ψηφίο ισοτιμίας. Με τον νέο κώδικα προσθέτουμε ένα ακόμα ψηφίο ισοτιμίας για κάθε στήλη.

Αν δημιουργηθεί ένα μόνο λάθος κατά τη μεταφορά ανιχνεύεται είτε από τα οριζόντια , είτε από τα κάθετα ψηφία ισοτιμίας. Ο συνδυασμός όμως των δύο ομάδων ψηφίων ισοτιμίας εντοπίζει και τη θέση του λάθους, το οποίο θα βρίσκεται στη τομή της γραμμής και της στήλης που εντόπισαν το λάθος. Μετά τον εντοπισμό της θέσης του λάθους γίνεται η διόρθωση. Ο κώδικας που περιγράψαμε χρησιμοποιείται στις περιπτώσεις που έχουμε μαζική μεταφορά πληροφοριών , όπως για παράδειγμα στις μαγνητικές ταινίες.

Παράδειγμα:

Υποθέτουμε ότι έχουμε την μεταφορά μιας πληροφορίας που αποτελείται από του χαρακτήρες "ERROR". Οι χαρακτήρες είναι κωδικοποιημένες στο σύστημα ASCII με 7 Bits. Ο εντοπισμός υποθέσουμε ότι έχουμε λανθασμένη μεταφορά στο bit 4 (από αριστερά) του χαρακτήρα "0" (μηδέν αντί για ένα). Με τη βοήθεια των αθροίσεων που θα κάνει ο ακολουθεί την άρτια ισοτιμία.



byte

Ανδέκτης είναι σε θέση να εντοπίσει και να διορθώσει το λάθος.

3.3. Διαγράμματα, Χάρτες, Συνθήκες

Κώδικας Hamming: Ο Κώδικας αυτός σχηματίζεται από την πληροφορία και από ένα αριθμό από ψηφία ελέγχου. Για κάθε δυαδικό μήνυμα μήκους N bits (A_N, A_{N-1}, ... A_1) που πρόκειται να μεταφερθεί προστίθενται και K ψηφία ελέγχου (C_K, C_K-1, ..., C_1). Ο αριθμός των ψηφίων ελέγχου K έχει σχέση με το μήκος του μηνύματος N. Τα ψηφία ισοτιμίας τοποθετούνται σε ειδικές θέσεις του σύνθετου μη μηνύματος. Οι θέσεις είναι η 1,2,4,8,... 2^{K-1}.

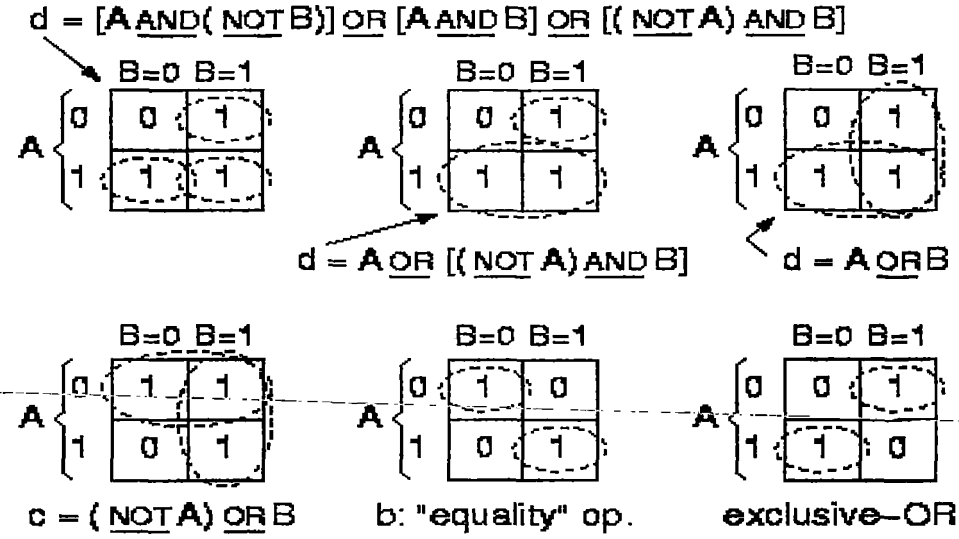
Η τιμή του κάθε ψηφίου ισοτιμίας C_i ορίζεται από τον έλεγχο ισοτιμίας ορισμένων από την αρχή ομάδων bits του αρχικού μηνύματος. Οι ομάδες αυτές ορίζονται από ειδικούς πίνακες. Η διάταξη των ομάδων είναι τέτοια ώστε επιτρέπει την ανίχνευση και την διόρθωση ενός απλού λάθους.

Διαγράμματα Venn Δύο Μεταβλητών, Χάρτες
karnaugh
Οι λογικές συναρτήσεις 2 εισόδων καθορίζονται πλήρως από τις 4 τιμές που αυτές παίρνουν στον καθένα από τους 4 (=2^2) συνδυασμούς των 2 εισόδων τους (γι' αυτό και υπάρχουν ακριβώς 16 (=2^4) διαφορετικές τέτοιες λογικές συναρτήσεις 2 εισόδων). Άρα, ο πίνακας αληθείας τους περιέχει 4 δυαδικές τιμές. Γιά εύκολη αναγνώριση "με το μάτι" των λογικών αυτών συναρτήσεων μας βολεύει να διατάξουμε τις 4 αυτές τιμές σ' ένα δισδιάστατο, όπου οι γραμμές αντιστοιχούν στην τιμή της μίας μεταβλητής εισόδου και οι στήλες αντιστοιχούν στην άλλη μεταβλητή εισόδου, όπως φαίνεται στο σχήμα. Η παράσταση αυτή, ανάλογα πώς την βλέπει και πώς την χρησιμοποιεί κανείς, λέγεται είτε **Χάρτης Karnaugh** (Καρνώ) είτε **Διάγραμμα Venn**.

3.3.1. Διάγραμμα Venn

Το διάγραμμα Venn έχει το εξής νόημα: θεωρούμε ότι η μεταβλητή εισόδου A προσδιορίζει κατά πόσον είμαστε *μέσα* στην "περιοχή A" (κάτω γραμμή) ή έξω από αυτήν (1 = μέσα, 0 = έξω). Ομοίως, η είσοδος B μας λέει αν είμαστε μέσα (1) στην περιοχή B (δεξιά στήλη) ή έξω (0) από αυτήν. Όταν μας δίνουν μία λογική συνάρτηση (δηλαδή άσσους και μηδενικά στα 4 τετραγωνάκια), εμείς κοιτάζουμε σε ποία περιοχή η συνάρτηση αυτή γίνεται αληθής (τιμή = 1), και περιγράφουμε αυτή την περιοχή σαν συνάρτηση των τιμών των εισόδων A και B. Αν η συνάρτηση γίνεται αληθής μόνο στο κάτω δεξιό τετράγωνο, τότε γίνεται αληθής όταν είμαστε "μέσα" στο A (A=1) **και** "μέσα" στο B (B=1), δηλαδή όταν είναι αληθές το A **και** αληθές το B, δηλαδή όταν "A **ΚΑΙ** B" ("A **AND** B"). Κατ' αναλογία, όταν μία λογική συνάρτηση 2 μεταβλητών γίνεται αληθής μόνο στο κάτω αριστερό τετράγωνο, τότε αυτή γίνεται αληθής όταν είμαστε μέσα στο A (A=1) **και** "έξω" από το B (B=0), δηλαδή όταν είναι αληθές το A **και** ψευδές (**όχι** αληθές) το B , δηλαδή όταν "A **ΚΑΙ** (**ΟΧΙ** B)" ("A **AND** (**NOT** B)"). Ομοίως, η λογική συνάρτηση που γίνεται αληθής μόνο στο πάνω δεξιό τετράγωνο είναι η "(**NOT** A) **AND** B", ενώ η συνάρτηση που γίνεται αληθής μόνο στο πάνω αριστερό τετράγωνο είναι η "(**NOT** A) **AND** (**NOT** B.Όταν μία λογική συνάρτηση γίνεται αληθής (τιμή = 1) σε δύο διπλανά τετράγωνα, τότε αυτή μπορεί να περιγραφεί σαν "μέσα" ή "έξω" από την περιοχή της μίας από τις δύο μεταβλητές εισόδου. Έτσι, στο παραπάνω παράδειγμα, η έξοδος f, που γίνεται 1 στα δύο τετράγωνα της

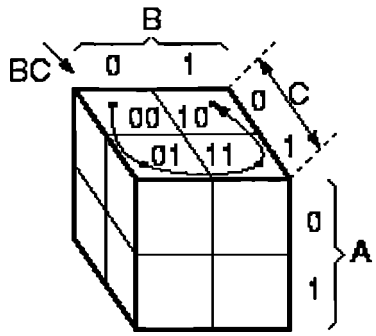
επάνω γραμμής, μπορεί να περιγραφεί σαν "έξω" από το A, δηλαδή **OXI** A, όπως φαίνεται στο παραπάνω σχήμα. Αντίστοιχα, μία συνάρτηση που θα γίνονταν 1 στην κάτω γραμμή θα ήταν ίση με A, αν γίνονταν 1 στη δεξιά στήλη θα ήταν ίση με B, και αν γίνονταν 1 στην αριστερή στήλη θα ήταν ίση με **NOT** B.

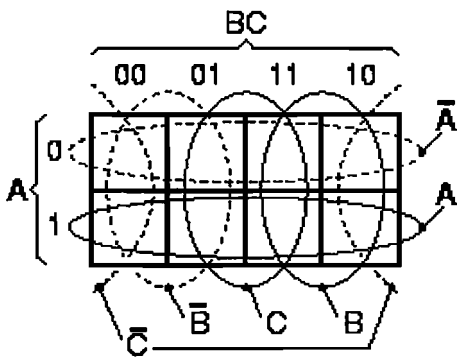


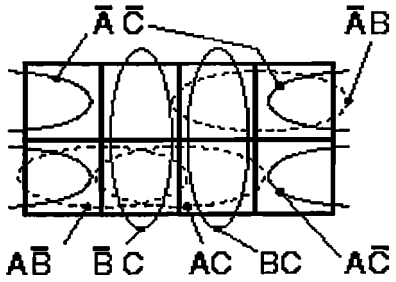
Μιά λογική συνάρτηση που γίνεται αληθής σε τρία τετράγωνα μπορεί να περιγραφεί όπως στο επόμενο σχήμα: επάνω φαίνονται τρείς εναλλακτικές περιγραφές γιά τη συνάρτηση που οδηγεί το τμήμα dτης ένδειξης 7 τμημάτων

Στην πρώτη περιγραφή, η περιοχή αληθείας της d ορίζεται σαν η ένωση τριών περιοχών μεγέθους ενός τετραγώνου η καθεμία. Καθώς παραπάνω η τομή περιοχών αντιστοιχούσε στο λογικό **και**, η ένωση περιοχών, εδώ, αντιστοιχεί στο λογικό **ή**, αφού η συνάρτηση είναι αληθής όποτε είναι αληθής ή ο ένας όρος, ή ο δεύτερος, ή ο τρίτος (ή περισσότεροι ταυτόχρονα). Έτσι, οι τρεις περιοχές του πρώτου σχήματος μας δίνουν την περιγραφή της συνάρτησης που φαίνεται από πάνω, η οποία είναι το λογικό **ή** τριών όρων που ο καθένας τους είναι ένα λογικό **και**.

Η δεύτερη περιγραφή της λογικής συνάρτησης d οδηγεί σε απλούστερη έκφραση, διότι χρησιμοποιεί λιγότερες και μεγαλύτερες βασικές περιοχές. Όσο μεγαλύτερη είναι μία βασική περιοχή **γειτονικών τετραγώνων**, τόσο λιγότερους όρους "**και**" έχει η αντίστοιχη λογική έκφραση· και όσο λιγότερες περιοχές χρειάζεται να ενώσουμε για να καλύψουμε την περιοχή αληθείας της συνάρτησής μας, τόσο λιγότερα κομμάτια θα ενώνουν οι πράξεις **ή**. Τελικά, η οικονομικότερη περιγραφή της λογικής συνάρτησης d είναι η τρίτη, δεξιά, διότι χρησιμοποιεί τις μεγαλύτερες δυνατές βασικές περιοχές, δηλαδή τους απλούστερους όρους· η εν μέρει επικάλυψη των περιοχών δεν έχει καμία σημασία, αφού η λογική πράξη **ή** είναι αληθής όταν είναι αληθείς είτε μία είτε περισσότερες από τις εισόδους της. Κατ' ανάλογο τρόπο, η έξοδος c που ζητάμε για το κύκλωμα του παραδείγματός μας είναι η λογική συνάρτηση "(**NOT** A) **OR** B", όπως φαίνεται στο παραπάνω σχήμα, κάτω αριστερά. Η μέθοδος αυτή της απλοποίησης λογικών συναρτήσεων μέσω συνένωσης γειτονικών τετραγώνων λέγεται "μέθοδος του **Χάρτη Καρνώ**" (Karnaugh Map).





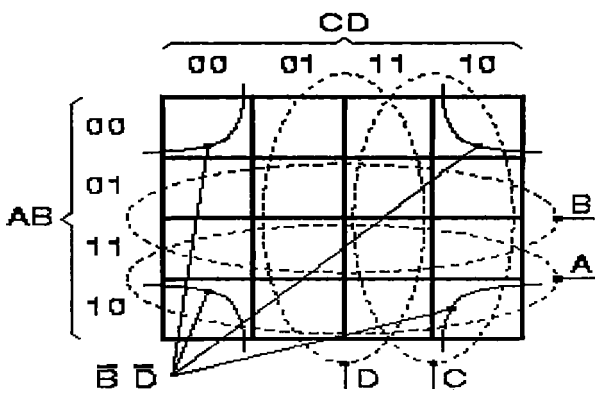


Η έξοδος b του κυκλώματος της πρέπει να ανάβει στα δύο τετράγωνα που φαίνονται στο παραπάνω σχήμα (κάτω μέση), τα οποία όμως, δυστυχώς, **δεν** είναι γειτονικά. Για το λόγο αυτό, τα δύο αυτά τετράγωνα δεν μπορούν να συνενωθούν όπως παραπάνω, και δεν μπορεί να γίνει καμιά ιδιαίτερη απλοποίηση της σχετικής συνάρτησης --αυτή παραμένει αναγκαστικά η ένωση δύο ανεξάρτητων τετραγώνων: b = [(**NOT** A) **AND** (**NOT** B)] **OR** [A **AND** B]. Πρόκειται γιά τη συνάρτηση ισότητας.Η άλλη λογική συνάρτηση 2 μεταβλητών που δεν απλοποιείται είναι η συνάρτηση αποκλειστικού Ή (exclusive OR, ή "XOR"), που επίσης είδαμε στην ίδια παράγραφο, και που φαίνεται στο παραπάνω σχήμα, κάτω δεξιά.

Διαγράμματα Venn και Χάρτες Karnaugh Τριών και Τεσσάρων Μεταβλητών Η επιτυχία των διαγραμμάτων Venn δύο μεταβλητών στο να μας βοηθούν να απλοποιούμε λογικές συναρτήσεις οφείλεται στη διάταξη των τιμών της συνάρτησης στις δύο διαστάσεις με τρόπο τέτοιο που η κάθε μεταβλητή να αντιστοιχεί στη μία διάσταση. Για να πετύχουμε το ίδιο αποτέλεσμα με τρεις μεταβλητές εισόδου, φανταζόμαστε ότι διατάσουμε τα περιεχόμενα του πίνακα αληθείας στις 3 διαστάσεις, σ' ένα σχήμα κύβου, όπως στο εδώ σχήμα. Στη συνέχεια, επειδή είναι άβολο να σχεδιάζουμε τρισδιάστατα σχήματα στο χαρτί, φανταζόμαστε ότι ξεδιπλώνουμε τον κύβο, κόβοντάς τον πίσω στη μέση, και φέρνοντας τις δύο πίσω στήλες στην αριστερή και στη δεξιά άκρη αντίστοιχα,

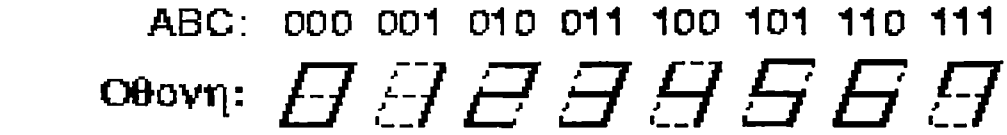
όπως φαίνεται στο σχήμα, δεξιά. Η συνέπεια είναι ότι οι στήλες του πίνακα αντιστοιχούν στις δύο μεταβλητές εισόδου B και C με τη σειρά που φαίνεται στο σχήμα: 00, 01, 11, 10 –η σειρά αυτή (που λέγεται και "κώδικας Gray") διαφέρει από τη συνηθισμένη σειρά αρίθμησης (μέτρησης) στο δυαδικό (00, 01, 10, 11). Με τη σειρά αυτή πετυχαίνουμε η περιοχή αληθείας της μεταβλητής B να αποτελείται από 4 γειτονικά τετράγωνα --τα 4 δεξιά τετράγωνα-- και ταυτόχρονα η περιοχή αληθείας της μεταβλητής C να αποτελείται επίσης από 4 γειτονικά τετράγωνα --τα 4 μεσαία τετράγωνα. Η περιοχή αληθείας του C' (δηλ. του "NOT C") είναι επίσης 4 "γειτονικά" τετράγωνα --δύο στη άκρη αριστερά και δύο στην άκρη δεξιά-- αλλά για να καταλάβουμε ότι αυτά είναι γειτονικά. Γενικά, στο χάρτη 3 μεταβλητών, οιαδήποτε 4 "γειτονικά" τετράγωνα δηλαδή 4 τετράγωνα σε σχήμα 2x2 ή 4x1.αντιστοιχούν σε μία μεταβλητή εισόδου ή στην άρνησή της. Οι δύο οριζόντιες τετράδες αντιστοιχούν στο A και στο A' (NOT A).

Περιοχές δύο γειτονικών τετραγώνων αντιστοιχούν στο λογικό ΚΑΙ δύο εκ των τριών μεταβλητών εισόδου (ή των αρνήσεών τους). Μερικά τέτοια ζευγάρια φαίνονται στο κάτω μέρος του σχήματος. Παρατηρήστε ότι τα ζευγάρια που περιλαμβάνουν το C' μοιάζουν "κομμένα" --ένα τετράγωνο στην άκρη αριστερά και ένα στην άκρη δεξιά-- όμως αποτελούνται και αυτά από τετράγωνα που ήταν γειτονικά πριν ξετυλίξουμε το χάρτη από τη μορφή βαρελιού που λέγαμε πιο πάνω. Τέλος, φυσικά, όπως και στους χάρτες 2 μεταβλητών, μεμονωμένα τετράγωνα αντιστοιχούν στο λογικό ΚΑΙ όλων των μεταβλητών εισόδου ή των αρνήσεών τους, και περιοχές που προκύπτουν από την ένωση ομάδων τετραγώνων αντιστοιχούν στο λογικό Ή των σχετικών όρων. Όταν καλύπτουμε περιοχές με τέτοιες ενώσεις, επιδιώκουμε η κάθε ομάδα γειτονικών τετραγώνων να είναι όσο μεγαλύτερη γίνεται (πάντα βέβαια μεγέθους δύναμης του 2, διατεταγμένη σε σχήμα ορθογωνίου) επικαλύψεις περιοχών δεν μας ενοχλούν --αντίθετα βοηθούν στη μεγιστοποίηση της έκτασης της κάθε μεμονωμένης περιοχής.



Ο χάρτης Karnaugh τεσσάρων μεταβλητών σχεδιάζεται όπως φαίνεται στο επόμενο σχήμα. Πρέπει να φανταστούμε ότι αυτός ο πίνακας 4x4 προέρχεται από **διπλό ξετύλιγμα** μιάς παράξενης σφαιροειδούς επιφάνειας και οριζόντια και κατακόρυφα. Η αριστερή και η δεξιά στήλη ήταν γειτονικές πριν το ξετύλιγμα, σαν να προέρχονται από ένα όρθιο βαρέλι, και αντιστοιχούν στη μεταβλητή D' (NOT D). Ταυτόχρονα, η επάνω και η κάτω γραμμή ήταν κι αυτές γειτονικές, Τετράδες γειτονικών τετραγώνων αντιστοιχούν στο λογικό ΚΑΙ δύο εκ των τεσσάρων μεταβλητών εισόδου, ή των αρνήσεών τους. Μία τέτοια τετράδα --η πιο πολύ κομμένη απ' όλες-- φαίνεται στο σχήμα στις 4 γωνίες: πρόκειται για την περιοχή B'D', και τα τετράγωνά της ήταν όλα γειτονικά πριν το διπλό ξετύλιγμα. Άλλες κομμένες τετράδες έχουν 2 τετράγωνα αριστερά και 2 δεξιά, ή 2 τετράγωνα επάνω και 2

κάτω.Ζευγάρια γειτονικών τετραγώνων αντιστοιχούν στο λογικό ΚΑΙ τριών εκ των τεσσάρων μεταβλητών εισόδου, ή των αρνήσεών τους. Μεμονωμένα τετράγωνα αντιστοιχούν στο λογικό ΚΑΙ όλων των μεταβλητών εισόδου (ή των αρνήσεών τους).



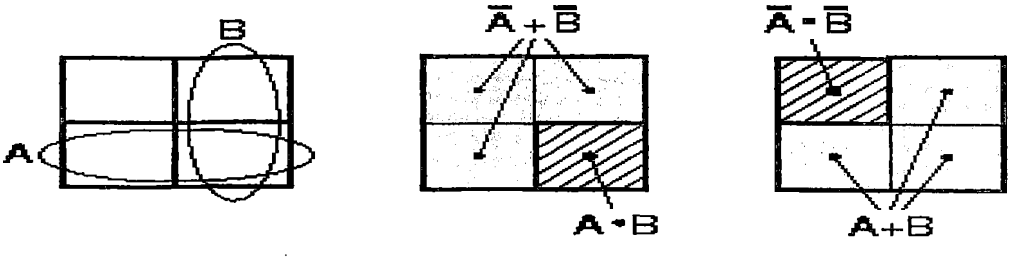
Μπορούν να οριστούν και χάρτες Karnaugh πέντε ή περισσοτέρων μεταβλητών, αλλά δεν είναι πρακτικοί. Εξ' άλλου, ας μην ξεχνάμε ότι η απλοποίηση λογικών συναρτήσεων "με το μάτι" και "με το χέρι" ανήκει στο παρελθόν: σήμερα υπάρχουν αποδοτικοί αλγόριθμοι και αντίστοιχα προγράμματα που κάνουν αυτές τις απλοποιήσεις και πολλές άλλες αυτόματα".

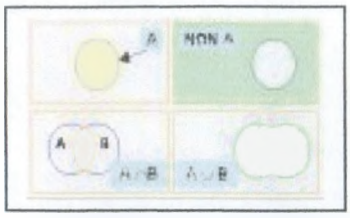
Άσκηση: Οι Αριθμοί 0-7 στην Οθόνη 7 Τμημάτων. Η άσκηση αυτή διότι τα κυκλώματα που προκύπτουν είναι πολύ μεγάλα για να υλοποιηθούν στο χρόνο ενός εργαστηρίου μας --απλώς θα παραδώσετε τη λύση σας μέσα στην αναφορά του εργαστηρίου σας. Ζητείται να βρεθούν οι 7 λογικές συναρτήσεις οδήγησης των 7 τμημάτων του γνωστού μας ενδείκτη, προκειμένου αυτός να εμφανίζει τις 8 ενδείξεις που φαίνονται στο σχήμα, ανάλογα με τον εκάστοτε συνδυασμό τιμών των τριών bits εισόδου A, B, και C. Φτιάξτε τους 7 πίνακες αληθείας σε μορφή 7 χαρτών Karnaugh τριών μεταβλητών, βρείτε τις ομάδες γειτονικών τετραγώνων που χρειάζονται γιά να καλύψουν τις περιοχές ανάματος της κάθε LED, γράψτε τις αντίστοιχες λογικές συναρτήσεις για τις 7 εξόδους, και σχεδιάστε το αντίστοιχο κύκλωμα με πύλες. *Συνθήκες Αδιαφορίας (Don't Care Conditions) Σε όλες τις παραπάνω περιπτώσεις, οι πίνακες αληθείας των επιθυμητών εξόδων ήταν πλήρως προδιαγεγραμμένοι, δηλαδή μας ενδιέφερε η κάθε έξοδος να έχει μιά συγκεκριμένη, προκαθορισμένη τιμή για τον κάθε δυνατό συνδυασμό τιμών των εισόδων.* Υπάρχουν όμως και περιπτώσεις "μερικώς προδιαγεγραμμένων" συστημάτων. Για παράδειγμα, στην παρακάτω άσκηση, μας ζητείται να οδηγήσουμε τη γνωστή μας οθόνη 7 τμημάτων σε τρόπον ώστε να εμφανίζονται σε αυτήν τα δέκα ψηφία από το 0 ως το 9. Για να πετύχουμε δέκα διαφορετικές εξόδους δεν αρκούν προφανώς 3 bits εισόδου -- χρειάζονται τέσσερα. Όμως, τα 4 bits έχουν 16 δυνατούς συνδυασμούς· διαλέγουμε δέκα από αυτούς για να γεννάνε στην οθόνη τα δέκα επιθυμητά ψηφία, ενώ δεν μας ενδιαφέρει τι θα κάνει η οθόνη όταν στις εισόδους εμφανίζεται ένας από τους υπόλοιπους έξι συνδυασμούς --π.χ. το κύκλωμα που μας τροφοδοτεί με τα 4 bits εισόδου ποτέ δεν θα μας δίνει έναν από τους υπόλοιπους 6 συνδυασμούς, ή αν μας δώσει, δεν μας ενδιαφέρει τι θα δείξει η οθόνη σε αυτή την περίπτωση. Όταν ένας πίνακας αληθείας δεν προκαθορίζει την τιμή εξόδου για ορισμένο συνδυασμό τιμών εισόδου, λέμε ότι εκεί έχουμε μία συνθήκη αδιαφορίας (don't care condition), και συχνά βάζουμε στον στη θέση εκείνη του πίνακα και του χάρτη Karnaugh σα σύμβολο ένα "x". Στο χάρτη Karnaugh, όταν αναζητούμε την ελάχιστη δυνατή ένωση των μέγιστων δυνατών περιοχών γειτονικών τετραγώνων προκειμένου να καλύψουμε τους

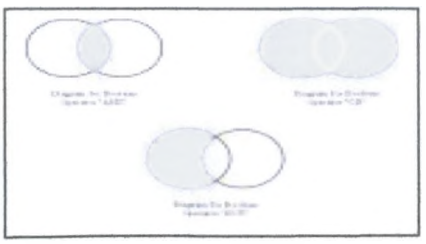
άσσους του χάρτη, θεωρούμε ότι το κάθε "x" είναι ό,τι μας βολεύει. Αν βολεύει να το θεωρήσουμε σαν άσσο, προκειμένου να πετύχουμε μεγαλύτερη περιοχή γειτονικών τετραγώνων, το θεωρούμε σαν άσσο. Αν βολεύει να το θεωρήσουμε σαν μηδενικό, προκειμένου να μην χρειαστούμε μια επιπλέον περιοχή για να το καλύψουμε, το θεωρούμε σαν μηδενικό.

3.3.2. Άλγεβρα Boole

Οι λογικές πράξεις ΚΑΙ, Ή, ΟΧΙ πάνω σε δυαδικές ψηφιακές μεταβλητές ορίζουν μιαν Άλγεβρα, η οποία ονομάστηκε "Άλγεβρα Boole" (Boolean Algebra), προς τιμήν του Μαθηματικού George Boole ο οποίος δημοσίευσε τις πρώτες σχετικές ιδέες το 1849, καθώς εργάζονταν πάνω στη μαθηματική διατύπωση των κανόνων της λογικής νόησης. Περίπου 90 χρόνια αργότερα, το 1938, ο Claude Shannon έδειξε ότι η Άλγεβρα Boole, όπως είχε εν τω μεταξύ εξελιχθεί, περιγράφει και τη λειτουργία των κυκλωμάτων διακοπτών, όπως κάναμε κι εμείς στην αρχή του μαθήματός μας. Η Άλγεβρα Boole μπορεί να δομηθεί ξεκινώντας από τον ("αξιωματικό") ορισμό των τριών πράξεων, ΚΑΙ, Ή, ΟΧΙ, βάσει του πίνακα αληθείας τους που από τον ορισμό τους μέσω της εξαντλητικής απαρίθμησης του αποτελέσματός τους για τον κάθε δυνατό συνδυασμό εισόδων. Για σκοπούς συντομογραφίας, από δω και πέρα, θα συμβολίζουμε τις λογικές αυτές πράξεις με AB [ή και με τελεία στη μέση] (Α και Β), A+B (Α ή Β), και A' [ή και με παύλα από πάνω] (όχι Α)τις μεταβλητές της Άλγεβρας Boole, δηλαδή τις δυαδικές ψηφιακές μεταβλητές, τις λέμε και "*Μεταβλητές Boole*" (Boolean variables). Όπως έχουμε πει, οι δύο τιμές μιας μεταβλητής Boole μπορεί να συμβολίζουν πολλά και διαφορετικά πράγματα, π.χ. αναμένο-σβηστό, ζεστό-κρύο, πάνω-κάτω, μπρός-πίσω, αριστερά-δεξιά, πατημένος-ελεύθερος (διακόπτης), ψηλή-χαμηλή (ηλεκτρική τάση), περνάει - δεν περνάει (ρεύμα), αληθές-ψευδές, ναι-όχι, 1-0, κλπ. Για σκοπούς συντομογραφίας, και πάλι, συνήθως θα χρησιμοποιούμε τα σύμβολα 1 (αληθές, αναμμένο, κλπ), και 0 (ψευδές, σβηστό, κλπ). Ξεκινώντας από τον ορισμό των τριών πράξεων Boole, μπορούμε να αποδείξουμε πολλά θεωρήματα της Άλγεβρας Boole, τα οποία συχνά αντιστοιχούν και σε συνηθισμένες διατυπώσεις της καθημερινής μας λογικής σκέψης. Η απόδειξη μπορεί να γίνει με εξαντλητική επαλήθευση όλων των περιπτώσεων (πίνακας αληθείας), ή με διαγράμματα Venn, ή με τη χρήση άλλων θεωρημάτων. Εκθέτουμε εδώ κάμποσα τέτοια θεωρήματα, κατά σειρά σημαντικότητας. Δύο Αρνήσεις κάνουν Μία Κατάφαση:
Όπως ξέρουμε πολύ καλά και από την καθημερινή μας ζωή, και όπως αποδεικνύεται άμεσα και από τον πίνακα αληθείας, το ΟΧΙ (ΟΧΙ Α) είναι το ίδιο με το Α, δηλαδή: (A')' = A.







3.3.3. Θεώρημα DeMorgan

Όπως παρατηρήσαμε ήδη κάμποσες φορές, και μετά, η άρνηση του **ΚΑΙ** ισοδυναμεί με το **Ή** των αρνήσεων, ενώ η άρνηση του Ή ισοδυναμεί με το **ΚΑΙ** των αρνήσεων. Η ιδιότητα αυτή, γνωστή σαν Αρχή του Δυϊσμού (Duality Principle) ή "Θεώρημα DeMorgan", διατυπώνεται επίσημα ως εξής:

1.(AB)' = A'+B'
2.(A+B)' = A'B'

Για να το αποδείξουμε, αρκεί να κατασκευάσουμε τους πίνακες αληθείας των παραπάνω συναρτήσεων και να τους συγκρίνουμε. Αν τους κατασκευάσουμε σε μορφή διαγράμματος Venn / χάρτη Karnaugh, και σημειώσουμε τα τετράγωνα όπου η κάθε συνάρτηση γίνεται 1, θα προκύψει το σχήμα που φαίνεται. Όπως βλέπουμε, η συνάρτηση AB γίνεται 1 στο κάτω δεξιά τετράγωνο, άρα το συμπλήρωμά της (η άρνησή της), δηλ. η συνάρτηση (AB)', θα γίνεται 1 στις υπόλοιπες περιπτώσεις(αριστερά και πάνω "Γ"). Αυτή η τελευταία περιοχή είναι ίδια με την ένωση (λογικό Ή) της περιοχής A' (A bar --δύο επάνω τετράγωνα) με την περιοχή B' (B bar --δύο αριστερά τετράγωνα), αποδεικνύοντας έτσι την πρώτη από τις παραπάνω σχέσεις. Αντίστοιχα μπορεί να αποδειχτεί και η δεύτερη σχέση, όπως φαίνεται στο δεξί μέρος του σχήματος.

Εναλλακτικά, η δεύτερη σχέση μπορεί να προκύψει από την πρώτη και από την ιδιότητα ότι δύο αρνήσεις κάνουν μία κατάφαση. Ξεκινάμε από το δεξί μέλος της ισότητας που θέλουμε να αποδείξουμε, και το μετασχηματίζουμε με δύο αρνήσεις: **A'B' = [(A'B')']'**. Μέσα στις αγκύλες υπάρχει η άρνηση ενός λογικού ΚΑΙ, άρα μπορούμε να εφαρμόσουμε σε αυτήν το πρώτο θεώρημα **DeMorgan,** και να την μετατρέψουμε στο λογικό Ή των αρνήσεων, οι οποίες στη συνέχεια μπορούν να απλοποιηθούν: **(A'B')' = (A')' + (B')' = A+B.** Βάσει αυτού, η προηγούμενη ισότητα μας δίνει: **A'B' = [(A'B')']' = [A+B]' = (A+B)'**, πράγμα που είναι ακριβώς το δεύτερο θεώρημα **DeMorgan.** Ο τρόπος αυτός απόδειξης μας οδηγεί σε μια δεύτερη διατύπωση της αρχής του δυϊσμού: εάν σε μιαν ισότητα της άλγεβρας Boole αλλάξουμε όλα τα **ΚΑΙ** με **Ή,** και όλα τα **Ή με ΚΑΙ,** τότε προκύπτει μια άλλη, επίσης αληθής ισότητα, η "δυϊκή" της πρώτης (όπως θα δούμε πιο κάτω, αν η ισότητα περιέχει και άσσους ή μηδενικά, τότε πρέπει και αυτά να τα αλλάξουμε από 0 σε 1 και από 1 σε 0).

Επιμεριστική-Ιδιότητα(Distributive-Property):

Κατ' ανάλογο τρόπο, μέσω του πίνακα αληθείας / διαγράμματος Venn, μπορεί κανείς να διαπιστώσει ότι:

A(B+C) = AB + AC
A+(BC) =
(A+B)(A+C)

αν ισχύει το A και επίσης ισχύει το B ή το C, τότε θα πρέπει να ισχύει το A και το B ή να ισχύει το A και το C. Αντίστοιχα, η δεύτερη σχέση λέει ότι αν ισχύει το A ή ισχύει το B και το C, τότε θα ισχύει το A ή το B, καθώς επίσης θα ισχύει το A ή το C. Παρατηρήστε ότι όταν χρησιμοποιούμε τα παραπάνω σύμβολα του ΚΑΙ που μοιάζει με το σύμβολο του πολλαπλασιασμού και του Ή που μοιάζει με το σύμβολο της πρόσθεσης, τότε η πρώτη από τις παραπάνω σχέσεις μοιάζει οικεία, αλλά η δεύτερη καθόλου (αφού, φυσικά, δεν μιλάμε για πρόσθεση και πολλαπλασιασμό).

Όπως και με τις δύο μορφές του θεωρήματος DeMorgan, οι δύο παραπάνω σχέσεις είναι *δυϊκές* μεταξύ τους: αν αντικαταστήσουμε τα ΚΑΙ με Ή και τα Ή με ΚΑΙ, τότε προκύπτει η μία από την άλλη. Ο λόγος είναι ότι η δεύτερη μπορεί να προκύψει από την πρώτη, εφαρμόζοντας τα θεωρήματα DeMorgan (δηλαδή την αρχή του δυϊσμού), και το ότι δύο αρνήσεις κάνουν μία κατάφαση. Ξεκινώντας με το αριστερό μέλος της δεύτερης σχέσης, το μετασχηματίζουμε ως εξής μέχρι να προκύψει το δεξί: A+(BC) = { [A+(BC)]'}' (δύο αρνήσεις) = { A'(BC)' }' (από DeMorgan) = { A'(B'+C') }' (από DeMorgan) = { A'B' + A'C' }' (από την πρώτη επιμεριστική ιδιότητα) = { (A+B)' + (A+C)' }' (από DeMorgan) = { [(A+B)(A+C)]' }' (από DeMorgan) = (A+B)(A+C) (δύο αρνήσεις).

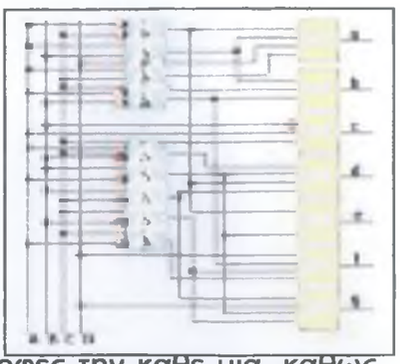
Αντιμεταθετική και Προσεταιριστική Ιδιότητα (Commutative and Associative Property):

Όπως ξέρουμε, η σειρά των μεταβλητών δεν παίζει ρόλο στις πράξεις ΚΑΙ και Ή (αντιμεταθετική ιδιότητα), όπως επίσης στα πολλαπλά ΚΑΙ η σειρά των πράξεων δεν παίζει ρόλο, και το ίδιο και στα πολλαπλά Ή (προσεταιριστική ιδιότητα --γι' αυτό και συνήθως τα γράφουμε χωρίς παρενθέσεις). Παρατηρήστε και πάλι τα ζευγάρια δυϊκών σχέσεων:

AB = BA A+B = B+A (αντιμεταθετική)
A(BC) = (AB)C [συνήθως γράφεται: ABC] (προσεταιριστική)
A+(B+C) = (A+B)+C [συνήθως γράφεται: A+B+C]

ΆλλαΘεωρήματατηςΆλγεβραςBoole:
Μπορούν εύκολα να αποδειχτούν τα παρακάτω επίσης θεωρήματα. Τα δύο θεωρήματα σε κάθε γραμμή --αριστερό και δεξί-- είναι *δυϊκά* μεταξύ τους: το ένα προκύπτει από το άλλο ανταλλάζοντας τα ΚΑΙ με τα Ή, και τα 1 με τα 0.

A·0 = 0 A+1 = 1
A·1 = A A+0 = A
A·A = A A+A = A
A·A'= 0 A+A'= 1
A(A+B) = A A+AB = A
A(A'+B)= AB A+A'B= A+B



(β) Αποδείξτε με διαγράμματα Venn την προσεταιριστική;) ιδιότητα, στις δύο δυϊκές τους μορφές την καθε μια, καθως και τα θεωρήματα A(A+B)=A, A+AB=A, A(A'+B)=AB, και A+A'B=A+B. Χρησιμοποιήστε 2 ή 3 τεμνόμενες ελλείψεις, που παριστάνουν τα σύνολα A, B, και C. Σημειώστε με κατάλληλο χρώμα ή διαγράμμιση τις περιοχές του επιπέδου που αντιστοιχούν στα B+C, A(B+C), AB, AC, AB+AC, BC, A+(BC), A+B, A+C, (A+B)(A+C)· ποιες περιοχές είναι ίδιες με ποιές; Αν έχετε χρόνο και διάθεση, κάντε το ίδιο για τις περιοχές AB, (AB)C, BC, A(BC), A+B, (A+B)+C, B+C, A+(B+C), και τέλος για τις AB, A+AB, A+B, A(A+B), A'B, A+A'B, A'+B, A(A'+B). Αποκλειστικό-Ή και Αποκλειστικό-ΟΥΤΕ (Ισότητα) ότι το *αποκλειστικό-Ή* ("exclusive-OR" ή "XOR") δύο μεταβλητών είναι αληθές τότε και μόνο τότε όταν μία και μόνο μία από τις δυό τους είναι αληθής .

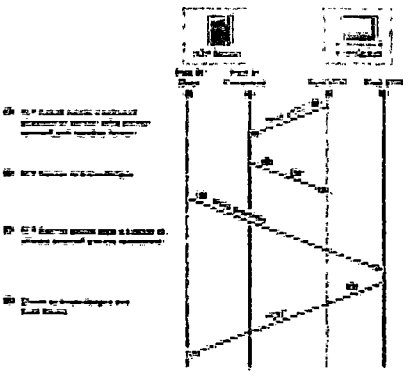
3.4. Λογικά Κυκλώματα

- Η πίνακες αληθείας των δύο αυτών συναρτήσεων, και αποδείξτε μέσω αυτών ότι η συνάρτηση ισότητας είναι η άρνηση (το "συμπλήρωμα") της συνάρτησης αποκλειστικού-Ή. Γιά το λόγο αυτό, η συνάρτηση ισότητας ονομάζεται και "αποκλειστικό-ΟΥΤΕ" ("exclusive-NOR" ή "XNOR").

- Από τον χάρτη Karnaugh του αποκλειστικού-Ή έχουμε δει ότι αυτό είναι: A XOR B = AB'+A'B. Αποδείξτε μέσω αλγεβρικών μετασχηματισμών ότι ισχύει επίσης: A XOR B = (A+B)(A'+B'). Ξεκινήστε από αυτή τη δεύτερη έκφραση, και εφαρμόστε πάνω της δύο φορές την επιμεριστική ιδιότητα του ΚΑΙ πάνω στο Ή, A(B+C)=AB+AC· στη συνέχεια, απλοποιήστε τους τέσσερεις όρους που προκύπτουν, μέχρι να φτάσετε στην πρώτη έκφραση.
(γ) Αφού η συνάρτηση ισότητας είναι η άρνηση του αποκλειστικού-Ή, θα ισχύει: A XNOR B = [A XOR B]' = [AB'+A'B]' = [(A+B)(A'+B')]'Αποκωδικοποιητής και Πολυπλέκτης με Πύλες

Πρόκειται για τα βασικά δικοπτόμενα κυκλώματα ή «πύλες» (gates) που χρησιμοποιούνται στους ψηφιακούς υπολογιστές και σε άλλες ψηφιακές ηλεκτρονικές συσκευές. Το σήμα εξόδου, που χρησιμοποιεί ένα δυαδικό σύστημα αρίθμησης (binary notation), ελέγχεται από το λογικό κύκλωμα

σύμφωνα προς το σύστημα εισόδου (input). Τα τρία βασικά λογικά κυκλώματα είναι το "and", το "or" και το "not".



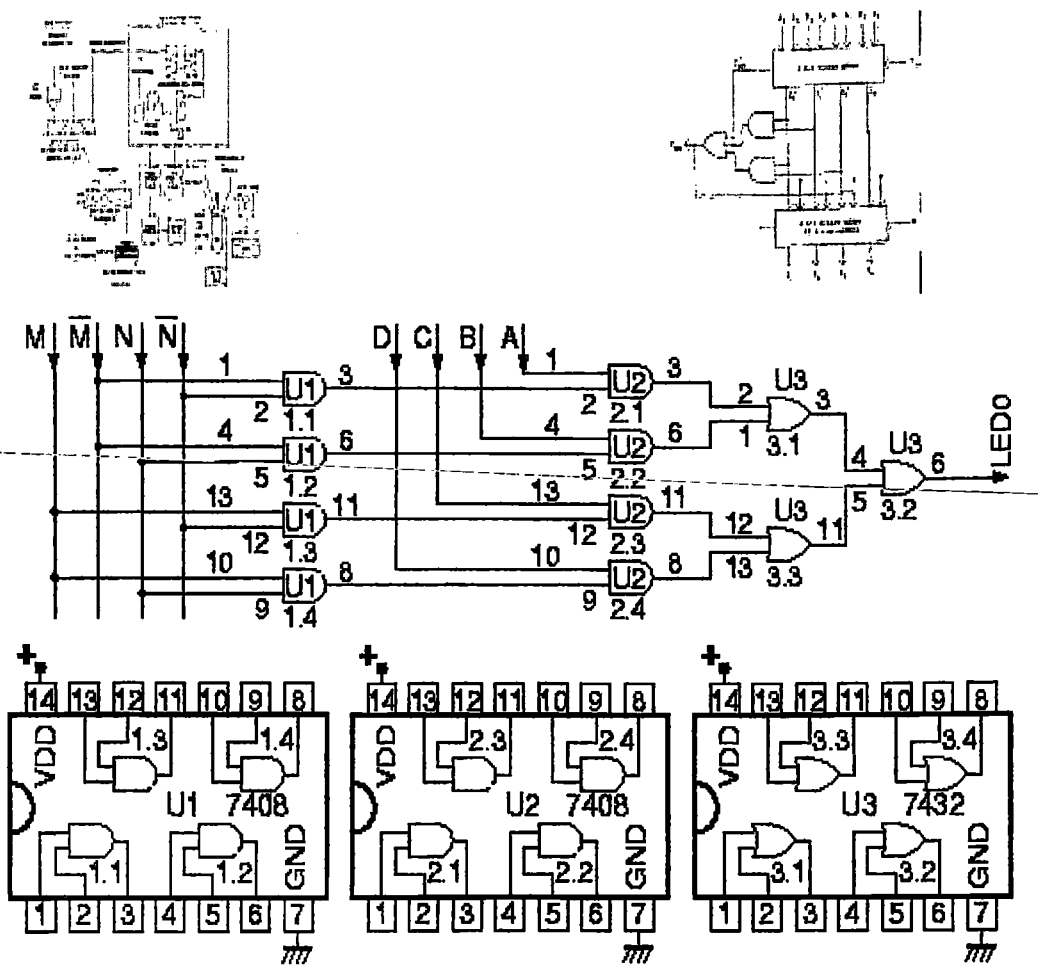
Το κύκλωμα "and" δίνει μια έξοδο (output) δυαδικού 1, αν κάποιο δυαδικό 1 υπάρχει σε κάθε κύκλωμα εισόδου. Διαφορετικά η έξοδος είναι ένα δυαδικό 0. Το κύκλωμα "or" παρέχει ένα δυαδικό 1 (στην είσοδο), αν ένα δυαδικό 1 υπάρχει τουλάχιστον σε ένα κύκλωμα εισόδου. Αλλιώς η έξοδος είναι δυαδικό 0. Τέλος το κύκλωμα "not" μετατρέπει το σήμα εισόδου, δίνοντας μια έξοδο δυαδικού 1 για μια είσοδο δυαδικού 0 ή μια έξοδο 0 για μια είσοδο 1.

Συχνά αυτά τα βασικά λογικά κυκλώματα χρησιμοποιούνται σε συνδυασμό π.χ. ένα κύκλωμα "nand" αποτελείται από τα κυκλώματα "not" και "and". Στα πλαίσια του ηλεκτρονικού εξοπλισμού τα λογικά κυκλώματα έχουν σχεδόν εντελώς ενσωματωθεί στα ολοκληρωμένα κυκλώματα.

3.4.1. Αποκωδικοποιητής και Πολυπλέκτης με Πύλες

Για τον αποκωδικοποιητή είναι ένα ψηφιακό κύκλωμα που έχει μία έξοδο για κάθε ένα, χωριστό συνδυασμό τιμών των εισόδων του. Η κάθε έξοδος ανάβει (γίνεται 1) μόνον όταν οι τιμές των εισόδων βρίσκονται στον αντίστοιχο συνδυασμό. Επομένως, κάθε φορά, μία και μόνο μία έξοδος είναι αναμμένη -- αυτή η οποία αντιστοιχεί στον παρόντα συνδυασμό τιμών των εισόδων. Από τα παραπάνω προκύπτει ότι ο πίνακας αληθείας της κάθε εξόδου, σε μορφή χάρτη Karnaugh, θα είναι παντού 0 εκτός ενός και μόνου τετραγώνου όπου θα είναι 1. Άρα, η συνάρτηση της κάθε εξόδου αντιστοιχεί στο λογικό ΚΑΙ όλων των μεταβλητών εισόδου --είτε αυτών καθεαυτών, είτε των συμπληρωμάτων τους· κάθε έξοδος θα έχει διαφορετικό συνδυασμό αληθών/συμπληρωμάτων γιά τις μεταβλητές εισόδου. Έτσι, η υλοποίηση του αποκωδικοποιητή με λογικές πύλες είναι αυτή που φαίνεται στο σχήμα --εδώ για την περίπτωση αποκωδικοποιητή 3-σε-8. Πάνω αριστερά στο σχήμα υπάρχει ένα συνηθισμένο, απλό σύμβολο για τον αποκωδικοποιητή· το σύρμα εισόδου με την πλάγια γραμμούλα και το "3" σημαίνει "τρία (αδελφά) σήματα (bits)", Στο κάτω μέρος του σχήματος φαίνεται και η παλαιά υλοποίηση του εργαστηρίου 1, με διακόπτες. Σε αυτήν βλέπουμε, π.χ., ότι ρεύμα θα φτάσει στην επάνω έξοδο όταν και μόνον όταν οι διακόπτες A, B, και C είναι όλοι "όχι πατημένοι", δηλαδή όταν (A')ΚΑΙ(B')ΚΑΙ(C'). Ομοίως, στην δεύτερη έξοδο θα φτάσει ρεύμα όταν και μόνον όταν A και B είναι "όχι πατημένοι" και C είναι

πατημένος, δηλαδή όταν (Α')ΚΑΙ(Β')ΚΑΙ(C), κ.ο.κ. Παρατηρούμε ότι οι λογικές αυτές συναρτήσεις είναι ίδιες με τις αντίστοιχες του κυκλώματος με πύλες.

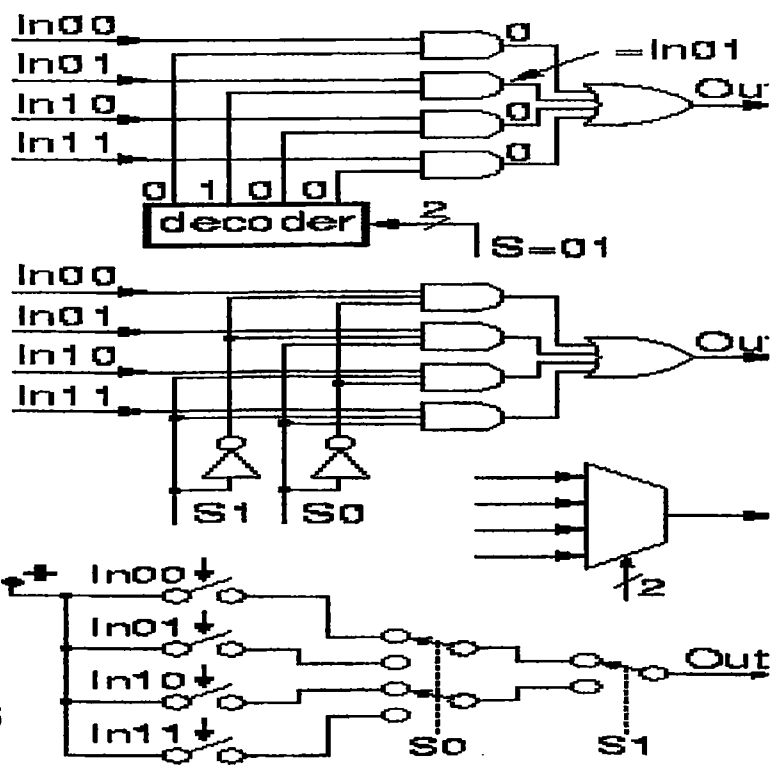


3.4.2. Πολυπλέκτης 4-σε-1 με Πύλες

Για πολυπλέκτες μιλήσαμε και είπαμε ότι έχουν μιάν είσοδο επιλογής (με όσα bits χρειάζεται) η οποία με την τιμή της έπιλέγει μιά από τις εισόδους δεδομένων, και στη συνέχεια η τιμή αυτής της επιλεγμένης εισόδου δεδομένων οδηγείται και δίδεται στην έξοδο· αλλάζοντας την τιμή των bits επιλογής αλλάζει και το ποιά είσοδος δεδομένων οδηγείται στην έξοδο. Επίσης, στο πείραμα παρατηρήσαμε ότι ο πολυπλέκτης έχει μιά συγγένεια με τον αποκωδικοποιητή, επειδή τα bits επιλογής πρέπει ναποκωδικοποιηθούν, ούτως ώστε γιά καθε διαφορετικό συνδυασμό τους να προκληθεί διαφορετική ροή πληροφοριών. Αυτό φαίνεαι στο επάνω μέρος του επομένου σχήματος: σε κάθε συνδυασμό τιμών των bits επιλογής, δηλαδή σε κάθε έξοδο του αποκωδικοποιητή, αντιστοιχεί κι από μία είσοδος δεδομένων, In00 έως In11, καθώς και από μιά πύλη ΚΑΙ. Στο παράδειγμα που δείχνει το σχήμα (πολυπλέκτης 4-σε-1), τα (δύο) bits επιλογής έχουν τιμή 01, και γι' αυτό η δεύτερη έξοδος του αποκωδικοποιητή είναι αναμένη (1), και φυσικά όλες οι άλλες σβηστές (0).

Επειδή A·0 = 0 γιά οιοδήποτε A, όλες οι πύλες ΚΑΙ εκτός από την μία "επιλεγμένη" πύλη δίνουν έξοδο 0. Επειδή A·1 = A γιά οιοδήποτε A, η επιλεγμένη πύλη δίνει στην έξοδό της τιμή ίση με αυτήν της εισόδου δεδομένων της --δηλαδή της In01 στο εδώ παράδειγμα με S=01. Στη συνέχεια, οι έξοδοι όλων των πυλών ΚΑΙ οδηγούνται στις εισόδους μιάς μεγάλης πύλης Ή. Επειδή A+0 = A γιά οιοδήποτε A, οι είσοδοι 0 της πύλης Ή --που αντιστοιχούν σε όλες τις πύλες ΚΑΙ πλην της επιλεγμένης-- δεν επηρεάζουν την τιμή εξόδου της Ή· έτσι, η τελική αυτή έξοδος παίρνει την τιμή της μίας εισόδου της πύλης Ή που δεν είναι αναγκαστικά 0, δηλαδή της εξόδου της επιλεγμένης πύλης ΚΑΙ, η οποία όπως είδαμε ισούται με την τιμή της επιλεγμένης εισόδου δεδομένων (της In01 εδώ, γιά S=01).

Επειδή ο αποκωδικοποιητής αποτελείται και αυτός από πύλες ΚΑΙ, και επειδή (AB)C = A(BC) = ABC, οι πύλες ΚΑΙ του αποκωδικοποιητή μπορούν να συνενωθούν με τις πύλες ΚΑΙ που αυτές οδηγούν, δίνοντας το κύκλωμα που φαίνεται στο μέσον του σχήματος. Στο κάτω μέρος του σχήματος υπάρχει το παλαιό κύκλωμα του πολυπλέκτη με διακόπτες, παρατηρούμε ότι ουσιαστικά πρόκειται γιά την ίδια λογική συνάρτηση: ρεύμα μπορεί να περάσει από την θετική τροφοδοσία προς την έξοδο Out όταν βρεί διέξοδο μέσα από έναν από τέσσερεις εναλλακτικούς δρόμους (4 παράλληλοι δρόμοι αντιστοιχούν στο λογικό Ή 4 εισόδων). Ο πρώτος δρόμος άγει όταν In00 πατημένος και S0 και S1 όχι πατημένοι, δηλαδή όταν (In00) (S0') (S1'), που αντιστοιχεί στην πρώτη πύλη ΚΑΙ του νέου κυκλώματος, κ.ο.κ. γιά τους άλλους τρείς εναλλακτικούς δρόμους.



Πείραμα:

Πολυπλέκτης 4-σε-1 με Πύλες Κατασκευάστε και ελέγξτε έναν πολυπλέκτη 4-σε-1 με πύλες, όπως το πρώτο κύκλωμα του προηγουμένου σχήματος. Δεν χρησιμοποιούμε τη δεύτερη παραλλαγή του κυκλώματος επειδή δεν έχουμε πύλες AND 3 εισόδων· ομοίως, επειδή δεν έχουμε πύλες OR 4 εισόδων, χρησιμοποιούμε ένα δέντρο τέτοιων πυλών των 2 εισόδων. Η πρώτη βαθμίδα του κυκλώματος είναι ένας αποκωδικοποιητής 2-σε-4 ίδιος με εκείνον του πειράματος εκμεταλλευόμαστε και πάλι το γεγονός ότι η πλακέτα εισόδων/ εξόδων μας δίνει τόσο τη θετική όσο και την αρνητική πολικότητα των διακοπτών M και N. Στο σχήμα που δίδεται εδώ, οι πύλες είναι τοποθετημένες σε σημεία που βοηθούν στην ανθρώπινη κατανόηση της λειτουργίας του κυκλώματος, και όχι στις θέσεις που αυτές έχουν μέσα στα chips της υλοποίησης. Γιά να μπορούμε όμως να βρίσκουμε αμέσως σε ποιό σύρμα του κυκλώματος αντιστοιχεί το κάθε σημείο του σχήματος, η κάθε πύλη έχει το όνομα του chip που την υλοποιεί (U1, U2, U3,...), και ο κάθε ακροδέκτης πύλης έχει τον αριθμό του ακροδέκτη του αντίστοιχου chip όπου αυτός βρίσκεται στο κύκλωμα. Συνδέστε την έξοδο σε μία ενδεικτική λυχνία και ελέγξτε το κύκλωμά σας: η LED δείχνει πάντα την τιμή της σωστής "επιλεγμένης" εισόδου δεδομένων A, B, C, ή D;

3.4.3. Μεθοδολογία Αποσφαλμάτωσης (Debugging) Κυκλωμάτων

Μεθοδολογία Αποσφαλμάτωσης (Debugging) Κυκλωμάτων Καθώς αυξάνει η πολυπλοκότητα των κυκλωμάτων που έχετε να υλοποιήσετε, αυξάνουν και οι κίνδυνοι σφαλμάτων κατά την κατασκευή που κάνετε στο εργαστήριο. Πρώτο σας μέλημα πρέπει πάντα να είναι να φτιάχνετε κυκλώματα σωστά εκ κατασκευής: σ' ένα πολύπλοκο σύστημα είναι ευκολότερο να προλάβετε τα λάθη από το να προσπαθείτε εκ των υστέρων να βρείτε σε τι οφείλονται λανθασμένες συμπεριφορές του συστήματος σε σπάνιες και παράξενες περιπτώσεις εισόδων του (και το ίδιο ισχύει και γιά τα προγράμματα που γράφετε σε άλλα μαθήματα και κυρίως στην επαγγελματική σας σταδιοδρομία). Επειδή όμως τα λάθη είναι ανθρώπινα, μερικές φορές θα σας τύχει να κάνετε μερικά, οπότε πρέπει να έχετε μια μεθοδολογία ελέγχου (test) του αποτελέσματος (είναι το κύκλωμα σωστό;), και αν ο έλεγχος αυτός διαπιστώσει λάθη χρειάζεστε μιά μεθοδολογία εύρεσής τους (αποσφαλμάτωση - debugging). Επιπλέον, υπάρχουν και άλλοι αστάθμητοι παράγοντες, όπως π.χ. ότι ενδέχεται μερικά chips ή μερικοί ακροδέκτες από chips να είναι καμένα/καμένοι, ή μερικοί ακροδέκτες ή σύρματα να μην κάνουν καλή επαφή στην πλακέτα. Τις δυσκολίες αυτές δεν έχετε να τις

αντιμετωπίσετε μόνο στο εργαστηριο του μαθήματός μας, αλλά και με τα πραγματικά κυκλώματα, μελλοντικά στην επαγγελματική σας σταδιοδρομία· οπλιστείτε λοιπόν με υπομονή και μέθοδο, και μάθετε από τώρα να τις αντιμετωπίζετε.

Ξεκινάτε πάντα την κατασκευή σας με ένα καθαρό, *κατανοητό* σχεδιάγραμμα του κυκλώματός σας όπως το σχήμα του παραπάνω πειράματος Κάνετε τις συνδέσεις σας με την τροφοδοσία κλειστή. Μόλις ανάψετε την τροφοδοσία, ακουμπήστε κάθε chip με το δάκτυλο σας να δείτε αν *ζεσταίνεται* υπερβολικά. Αν υποπτευθείτε ότι κάποιο chip ζεσταίνεται απότομα, σβήστε αμέσως την τροφοδοσία: ίσως και να το προλάβετε πρίν καεί! Ελέγξτε αν οι τάσεις τροφοδοσίας είναι συνδεδεμένες στους σωστούς ακροδέκτες. Γιά το ύποπτο chip, ελέγξτε τις εξόδους του: ίσως κάποια από αυτές είναι βραχυκυκλωμένη με τάση τροφοδοσίας ή με άλλη έξοδο του ίδιου ή άλλου chip.

Όταν ελέγχετε την τιμή (0 ή 1) ενός ακροδέκτη, προτιμάτε να την ελέγχετε *πάνω στον ίδιο* τον ακροδέκτη του chip, ει δυνατόν, αντί πάνω σε κάποιο σύρμα που (υποτίθεται ότι) είναι συνδεδεμένο στον ακροδέκτη: εάν η σύνδεση είναι κακή, ο ακροδέκτης θα σας δείξει τι βλέπει ή τι βγάζει το ίδιο το chip, ενώ το σύρμα μπορεί και να μην κάνει καλή επαφή. Όταν έχετε συνδέσει, στο κύκλωμά σας, τους ακροδέκτες δύο chips μεταξύ τους, ελέγξτε πρώτα την τιμή πάνω στον έναν ακροδέκτη, και στη συνέχει πάνω στον άλλον ακροδέκτη· αν τις βρείτε διαφορετικές, σημαίνει ότι η σύνδεση δεν είναι καλή. Γιά να ελέγξτε την τιμή (τάση) ενός ακροδέκτη, χρησιμοποιήστε ένα σύρμα συνδεδεμένο σε μιά ενδεικτική λυχνία (LED).

Εάν το κύκλωμά σας δεν συμπεριφέρεται όπως πρέπει, *ιχνηλατήστε* (trace) το σφάλμα κινούμενοι είτε προς τα πίσω, από τη λανθασμένη έξοδο προς τις εισόδους που την επηρρεάζουν, είτε προς τα εμπρός, από τις εισόδους προς τις εξόδους των πυλών. Ας πούμε ότι προχωρούμε από τις εισόδους προς τις εξόδους. Οι είσοδοι που εσείς δίνετε από τους διακόπτες, φτάνουν σωστές στα ποδαράκια του chip όπου φτάνουν; Αν όχι, φταίει κάποιο σύρμα ή σύνδεση. Αν όλες οι είσοδοι μιάς πύλης ενός chip έχουν τις σωστές τιμές (πάνω στα ποδαράκια του chip), η έξοδος αυτής της πύλης (πάνω στα ποδαράκια του chip) έχει τη σωστή τιμή; Αν όχι (και οι τροφοδοσίες είναι σωστές), υποπτευόμαστε είτε ότι η έξοδος είναι βραχυκυκλωμένη με τάση τροφοδοσίας ή με άλλη έξοδο του ίδιου ή άλλου chip, είτε ότι το chip μπορεί να είναι καμένο. Μετά τον έλεγχο των πρώτων πυλών που τροφοδοτούνται από τις εξωτερικές εισόδους, προχωρούμε στον έλεγχο των επομένων πυλών, που τροφοδοτούνται από τις εξόδους των πρώτων, κ.ο.κ.

Με ανάλογη μεθοδολογία προχωρούμε και από τις εξόδους πίσω προς τις εισόδους. Ποιά έξοδος δεν έχει τη σωστή τιμή; Ποιά πύλη τροφοδοτεί αυτή την έξοδο; Η πύλη αυτή, πάνω στα ποδαράκια του chip, έχει τη σωστή ή λάθος τιμή; Αν η έξοδος της πύλης (πάνω στα ποδαράκια του chip) είναι λάθος, τότε οι είσοδοί της (πάνω στα ποδαράκια του chip) τι τιμή έχουν; Η τιμή των εισόδων δικαιολογεί την τιμή της εξόδου; Αν το λάθος της εξόδου δεν δικαιολογείται από τις τιμές των εισόδων, μήπως η έξοδος είναι βραχυκυκλωμένη με τάση τροφοδοσίας ή με άλλη έξοδο του ίδιου ή άλλου chip; Αν το λάθος της εξόδου οφείλεται σε λανθασμένες τιμές των εισόδων,

ποιός φταίει γι' αυτές; Έτσι προχωρούμε προς τα πίσω στο κύκλωμα, μέχρι να φρούμε τον αρχικό φταίχτη....

Κεφάλαιο 4: Αρχιτεκτονική υπολογιστικών συστημάτων

4.1. Κεντρική μνήμη

Η Κεντρική μνήμη (main memory), που αναφέρεται επίσης και ως πρωτεύουσα μνήμη (primary memory), δίνει σε ένα υπολογιστικό σύστημα τη δυνατότητα καταχώρησης και ανάκλησης:

Α. *των υπό εκτέλεση προγραμμάτων, που καταχωρούνται ως ακολουθίες πράξεων.*

Β. *των πληροφοριακών δεδομένων επί των οποίων τα εκτελούμενα προγράμματα εφαρμόζονται.*

Πιο απλά, η κεντρική μνήμη των υπολογιστικών συστημάτων θεωρούμαι ότι χρησιμοποιείται για τους ακόλουθους τέσσερις βασικούς λόγους:

1. Για να καταχωρούνται τα πληροφοριακά δεδομένα, τα οποία εισάγονται στο υπολογιστικό σύστημα για άμεση επεξεργασία.
2. Για να καταχωρούνται τα αποτελέσματα πράξεων, που εκτελούνται επί των πληροφοριακών δεδομένων, που ήδη είναι καταχωρημένα στη μνήμη.
3. Για να καταχωρούνται αντίγραφα των πληροφοριακών δεδομένων, που ήδη υπάρχουν στην μνήμη, αλλά απαιτείται η αναπαραγωγή τους.
4. Για να καταχωρούνται οι ακολουθίες των πράξεων, που πρόκειται να εκτελεσθούν. Οι πράξεις καταχωρούνται σε μια κωδικοποιημένη μορφή, που μετά την αποκωδικοποίηση της, είναι άμεσα εκτελέσιμοι από την αριθμητική και λογική μονάδα του υπολογιστικού συστήματος. Η μορφή με την οποία καταχωρούνται οι πράξεις αποτελεί μια τεχνική γλώσσα. Η γλώσσα αυτή, την οποία θεωρητικά "αντιλαμβάνεται" το υπολογιστικό σύστημα (δηλαδή η μηχανή), ονομάζεται, για το λόγο αυτό, "γλώσσα μηχανής" (machine language). Τα προγράμματα, όπως έχει αναφερθεί, αποτελούνται από οδηγίες (ή εντολές) προς τη μηχανή, για τις υπό εκτέλεση πράξεις. Αποτελούνται λοιπόν από ακολουθίες εντολών που καθορίζουν τις πράξεις, τις οποίες η μηχανή πρόκειται να εκτελέσει .

Η μνήμη ενός ηλεκτρονικού υπολογιστικού συστήματος μοιάζει με μια κυψέλη, που αποτελείται από ένα οργανωμένο πλήθος κυψελίδων, ή κελιών, ή στοιχείων. Κάθε στοιχείο της μνήμης μπορεί να βρίσκεται σε μια από δυο δυνατές καταστάσεις, ή μεταφορικά, να περιέχει μια από δυο δυνατές τιμές.

Παλαιοτέρα, η υλοποίηση των στοιχείων της μνήμης πραγματοποιείται με μικροσκοπικούς μαγνητικούς δακτυλίους. Οι οποίοι εμαγνητιζοντο είτε δεξιόστροφα, για να αποδώσουν τη μια κατάσταση, είτε αριστερόστροφα για αποδώσουν την άλλη.
Η μνήμη που αποτελείται από μαγνητικούς δακτυλίους, ονομάζεται μαγνητική μνήμη (core). Η ανάκληση της τιμής που περιέχει ένας μαγνητικός δακτύλιος

πραγματοποιείται μέσω ενός δεύτερου αγωγού, αλλά αυτό έχει σαν αποτέλεσμα την απώλεια της τιμής του και για το λόγο αυτό απαιτείται ένας επί πλέον αγωγός, για να επαναφέρει στο δακτύλιο την προηγούμενη τιμή του.

Ο μαγνητισμός των μαγνητικών δακτυλίων ήταν μόνιμος και μπορούσε να μεταβληθεί με αλλαγή της φοράς του ρεύματος του αγωγού που τον διαπερνούσε. Δεν ήταν συνεπώς αναγκαία η διατήρηση της μαγνητικής μνήμης με συνεχή τροφοδοσία ηλεκτρικού ρεύματος. Απλώς η μαγνητική μνήμη ήταν δυνατόν να παραμένει αμετάβλητη και να κρατά συνεπώς πληροφοριακά δεδομένα για όσο διάστημα ήταν επιθυμητό.
Η μαγνητική μνήμη και γενικά κάθε είδους μνήμης που δεν απαιτεί τροφοδοσία για να διατηρήσει τα στοιχεία που περιέχει, ονομάζεται «σταθερή μνήμη».
Σήμερα τα στοιχεία μνήμης υλοποιούνται μέσω ημιαγωγικών ολοκληρωμένων κυκλωμάτων πυριτίου που έχουν σαν ελάχιστο στοιχείο το «flip flop». Το flip flop θεωρείται όπως και ο μαγνητικός δακτύλιος, ως το ελάχιστο στοιχείο μνήμης.

Βασική διαφορά των **flip flops** από τους μαγνητικούς δακτυλίους, είναι:

Α . *Η μεγάλη τους ταχύτητα*
Β . *Ο μικρός τους όγκος*
Γ . *Δεν είναι σταθερή* (δηλαδή η ημιαγωγική μνήμη είναι ασταθής)

Τα ολοκληρωμένα κυκλώματα, που χρησιμοποιούνται στην κεντρική μνήμη των υπολογιστικών συστημάτων διακρίνονται στα:

1. *δυναμικά*
2. *στατικά*

και η αντίστοιχη μνήμη διακρίνεται σε δυναμική και στατική .

Η δυναμική μνήμη είναι ασταθής εξαιτίας της απώλειας του φορτίου του πυκνωτή και απαιτεί περιοδική αναπαραγωγή του φορτίου για να διατηρείται.

Η στατική μνήμη είναι και αυτή ασταθής αλλά δεν απαιτεί αναπαραγωγή των τιμών που περιέχουν τα στοιχεία της. Απαιτεί όμως τροφοδοσία με ρεύμα . Τα στοιχεία της στατικής μνήμης είναι πιο πολύπλοκα . Για το λόγο αυτό η στατική μνήμη χρησιμοποιείται σε ειδικές περιπτώσεις , ενώ η δυναμική μνήμη χρησιμοποιείται κυρίως στην κεντρική μνήμη των υπολογιστικών συστημάτων .

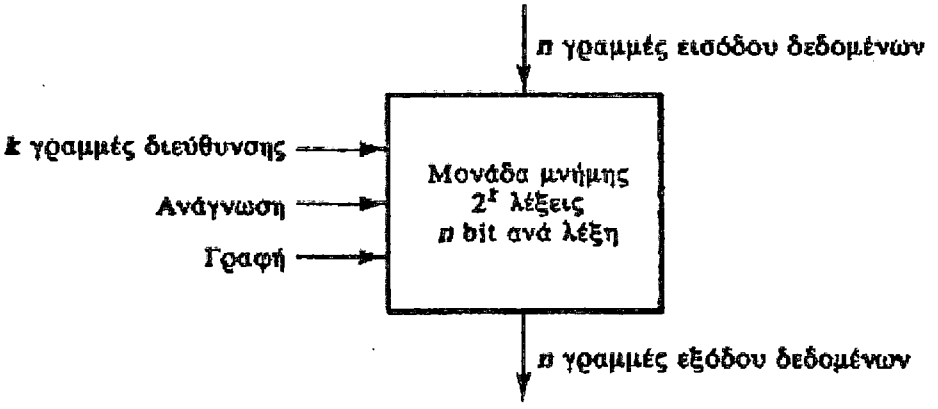
4.1.1. Τρόποι Διαχωρισμού Μνημών

1. Μόνο ανάγνωσης -Ανάγνωσης & Εγγραφής
2. Στατικές -Δυναμικές
3. Τυχαίας -Σειριακής Προσπέλασης
4. Πρόσκαιρες (volatile) ή Μη (non-volatile)

4.1.2. Μνήμη Τυχαίας Προσπέλασης

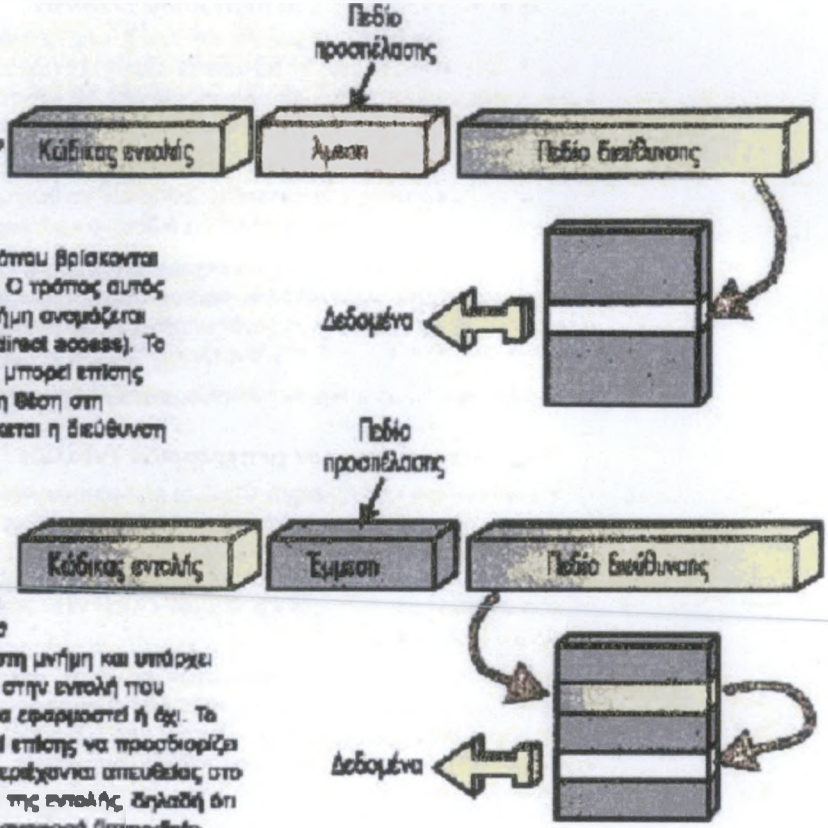
Μονάδα Μνήμης: Ένα σύνολο από κύτταρα αποθήκευσης μαζί με τα απαραίτητα κυκλώματα για τη μεταφορά πληροφοριών. Αν η μεταφορά αυτή μπορεί να γίνει από και προς οποιαδήποτε τυχαία θέση ονομάζεται Μνήμη Τυχαίας Προσπέλασης.

Οι δυαδικές πληροφορίες αποθηκεύονται ομαδοποιημένες σε λέξεις (words).



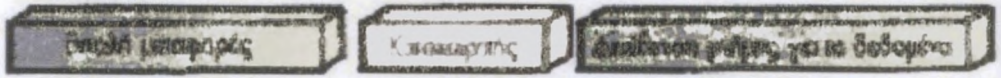
➤ *Πεδία που προσδιορίζουν τον τρόπο προσπέλασης στη μνήμη.*

Συνήθως το πεδίο διεύθυνσης της εντολής προσδιορίζει τη θέση της μνήμης όπου βρίσκονται κάποια δεδομένα. Ο τρόπος αυτός αναφοράς στη μνήμη ονομάζεται *άμεση αναφορά* (direct access). Το πεδίο διεύθυνσης μπορεί επίσης να προσδιορίζει τη θέση στη μνήμη όπου βρίσκεται η διεύθυνση η οποία τελικά περιέχει τα δεδομένα και πρέπει να χρησιμοποιήσει η εντολή. Αυτή είναι η *έμμεση αναφορά* (indirect access) στη μνήμη και υπάρχει ειδικό πεδίο μέσα στην εντολή που προσδιορίζει αν θα εφαρμοστεί ή όχι. Το πεδίο αυτό μπορεί επίσης να προσδιορίζει ότι τα δεδομένα περιέχονται απευθείας στο τμήμα διεύθυνσης της εντολής, δηλαδή ότι γίνεται απευθείας αναφορά (immediate access).



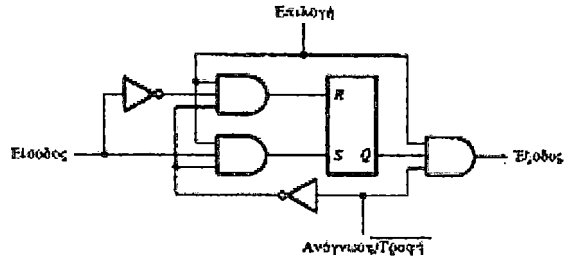
➢ *Πεδία που προσδιορίζουν τη συνθήκη για τις εντολές άλματος.* Στον Άβακα, η εντολή άλματος υπό συνθήκη ελέγχει αν ο συσσωρευτής Α έχει αρνητική τιμή. Ένα άλμα υπό συνθήκη όμως μπορεί να γίνει ανάλογα με τις τιμές διαφόρων καταχωρητών, με το αποτέλεσμα μίας αριθμητικής ή λογικής πράξης που έγινε προηγουμένως, κλπ. Οι συνθήκες αυτές κωδικοποι-ούνται μέσα σε ένα πεδίο της εντολής.

➢ *Πεδία που αναφέρονται σε καταχωρητές της ΚΜΕ.* Όταν η ΚΜΕ περιέχει πολλούς καταχωρητές, οι εντολές μπορούν να αναφέρονται σε οποιονδήποτε από αυτούς.

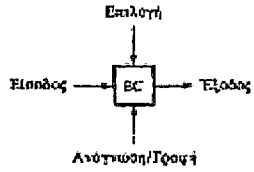


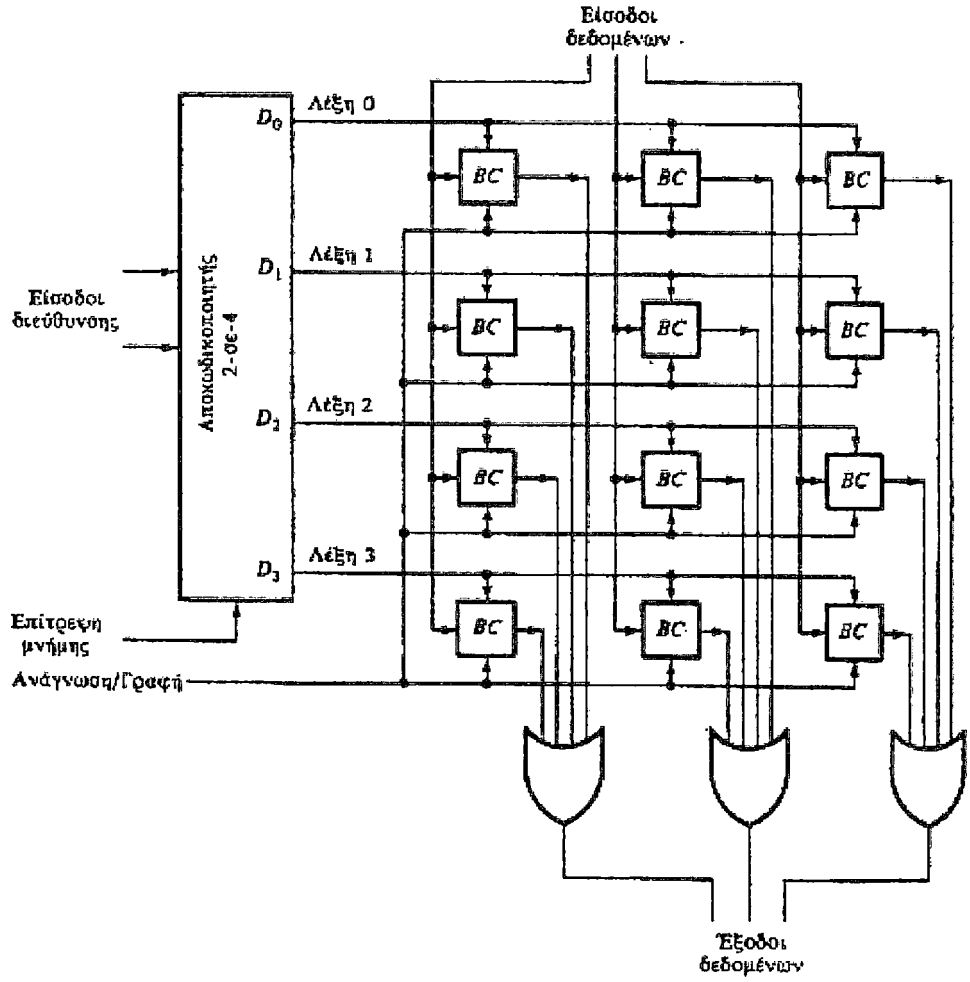
Μία εντολή της μορφής LDA, μπορεί να μεταφέρει δεδομένα από τη μνήμη σε οποιοδήποτε καταχωρητή και όχι μόνο τον Α. Έτσι υπάρχει ένας κώδικας εντολής, που σημαίνει «μεταφορά από τη μνήμη σε καταχωρητή» και ένα πεδίο εντολής που περιέχει τον «κωδικό» του καταχωρητή στον οποίο θα γραφούν τα δεδομένα.

4.1.3. Ένα κύτταρο μνήμης



(α) Λογικό διάγραμμα





4.1.4. Οργάνωση της Μνήμης

Οι δύο διαφορετικές καταστάσεις που είναι δυνατό να βρίσκεται το ελάχιστο στοιχείο μνήμης συμβολίζεται με τα δυαδικά ψηφία 0 και 1. Θεωρούμε ότι τα δυαδικά ψηφία που αποτελούν το σύνολο { 0 , 1 } είναι ένα αλφάβητο δύο πρακτικών συμβόλων.

Διάφορα στοιχεία μνήμης είναι:

Α . *Το bit*
Β . *Η λέξη*
Γ . *Το byte*

Bit

Το ελάχιστο στοιχείο μνήμης του pc μπορεί να αποδώσει και να αποθηκεύσει δύο απλές καταστάσεις όπου αντιστοιχούν σε ένα δυαδικό ψηφίο. Για το λόγο αυτό το ελάχιστο στοιχείο μνήμης ονομάζεται bit. Τα όνομα bit προέρχεται από το πρώτο και τα δύο τελευταία γράμματα του όρου binary digIT που σημαίνει δυαδικό ψηφίο.

Λέξη

Μία λέξη είναι μια ακολουθία από bit. Το πλήθος Ν των bit που περιέχονται σε μια λέξη είναι χαρακτηριστικό για κάθε είδος μηχανής.
Μία λέξη μπορεί να αποδώσει μέχρι 2 ^ Ν το πολύ διαφορετικές καταστάσεις οι οποίες αντιστοιχούν στους δυνατούς συνδυασμούς των καταστάσεων των Ν bit. Ο χαρακτηριστικός αριθμός Ν που αποδίδει το πλήθος των bit ανά λέξη ποικίλλει στους διάφορους τύπους μηχανών.

Byte

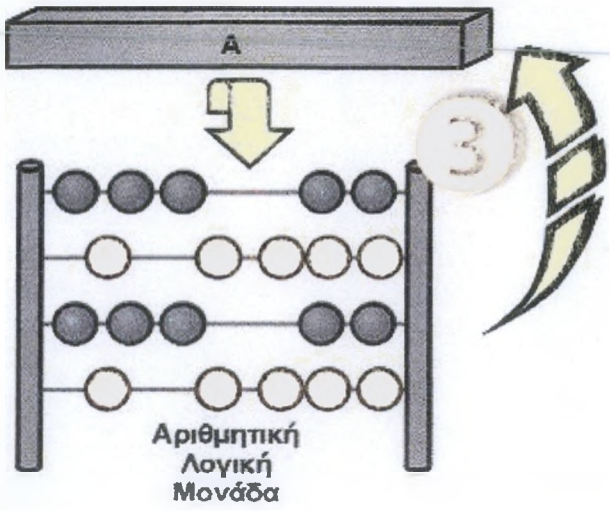
Είναι μια τυπική υποδιαίρεση των λέξεων που αποτελεί την πιο συνηθισμένη ομαδοποίηση των bit, είναι μια ακολουθία από bit που ονομάζεται byte. Σύμφωνα με αυτό μια λέξη των 8 bit αποτελείται από 1 byte.

4.2. Αριθμητική Λογική Μονάδα

Ένα ηλεκτρονικό υπολογιστικό σύστημα έχει τη δυνατότητα να εκτελεί αριθμητικές και λογικές πράξεις, να καταχωρεί στην κεντρική του μνήμη δεδομένα και ακολουθίες εντολών να μεταφέρει πληροφοριακά δεδομένα και να εισάγει πληροφοριακά δεδομένα από και προς το περιβάλλον του.

Οι συνηθισμένες αριθμητικές πράξεις όπως η πρόσθεση η αφαίρεση ο πολλαπλασιασμός και η διαίρεση θεωρούνται ως απλές . Υπάρχουν όμως και περισσότερο πολύπλοκες όπως οι υπολογισμοί των τιμών των διαφόρων συναρτήσεων δηλαδή εξαγωγή τετραγωνικής ρίζας, ύψωση σε δύναμη καθώς και άλλες τριγωνομετρικές και μαθηματικές συναρτήσεις. Στις λογικές πράξεις περιλαμβάνονται οι συγκρίσεις, οι συνδέσεις λογικών

παραστάσεων με τους λογικούς τελεστές και η πραγματοποίηση επιλογών μεταξύ ακολουθιών εντολών. Ο απλός χρήστης έχει την εντύπωση ότι όλες οι πράξεις εκτελούνται άμεσα από το υπολογιστικό σύστημα. Στην πραγματικότητα οι διαδικασίες εκτέλεσης των απλών ή και πολύπλοκων αριθμητικών και λογικών πράξεων και γενικά όλων των υπολογισμών διατυπώνονται πάντα πριν από την εκτέλεση τους στη γλώσσα της συγκεκριμένης μηχανής στην οποία πρόκειται να εκτελεστούν. Η γλώσσα αυτή είναι στοιχειώδης, περιλαμβάνει πολύ απλές εντολές οι οποίες εκτελούνται με την απλή αυτή μορφή και οδηγούν στους επιθυμητούς υπολογισμούς. Όλες οι πράξεις όμως, είτε αριθμητικές, είτε λογικές, πραγματοποιούνται στην Αριθμητική και Λογική μονάδα (Arithmetic and Logical Unit) του ηλεκτρονικού υπολογιστικού συστήματος.



Μονάδες εκτέλεσης ακεραίων

Η περισσότερη από τη δουλειά που γίνεται σε ένα υπολογιστή γίνεται με ακέραιες πληροφορίες, κάτι που σημαίνει ότι ακέραια νούμερα και δεδομένα αναπαρίστανται από ακέραιους αριθμούς. Οι ακέραιοι περιλαμβάνουν ακέραιους αριθμούς, χαρακτήρες και άλλα παρόμοια δεδομένα. Οι μη ακέραιοι αριθμοί ονομάζονται αριθμοί κινητής υποδιαστολής και χειρίζονται διαφορετικά από τη μονάδα κινητής υποδιαστολής. Η μονάδα ακεραίων μπορεί σε ορισμένους επεξεργαστές να χειριστεί και λειτουργίες κινητής υποδιαστολής, αλλά πολύ πιο αργά συγκρινόμενη με τη μονάδα κινητής υποδιαστολής.

Η μονάδα εκτέλεσης ακεραίων είναι εκεί που τελικά εκτελούνται οι εντολές. Παλαιότεροι επεξεργαστές είχαν μόνο μια τέτοια μονάδα και οι εντολές εκτελούνταν σειριακά. Οι πιο καινούριοι έχουν διάφορες τέτοιες μονάδες, επιτρέποντας σε περισσότερες από μια εντολές να εκτελούνται ταυτόχρονα, επιτρέποντας έτσι αύξηση της απόδοσης. Οι επεξεργαστές που μπορούν να το κάνουν αυτό ονομάζονται υπερβαθμωτοί (superscalar). Πιο εξελιγμένοι

επεξεργαστές μπορεί να έχουν εξειδικευμένες τέτοιες μονάδες, σχεδιασμένες για την εκτέλεση συγκεκριμένων τύπων εντολών.

Μονάδα κινητής υποδιαστολής (Floating Point Unit – FPU) και συνεπεξεργαστής (coprocessor)

Η μονάδα κινητής υποδιαστολής είναι μια εξειδικευμένη μονάδα εκτέλεσης πράξεων μεταξύ αριθμών κινητής υποδιαστολής. Αριθμός κινητής υποδιαστολής είναι ο οποιοσδήποτε αριθμός εκτός από ακέραιος. Κάθε αριθμός με δεκαδική τελεία αναπαρίσταται σαν αριθμός κινητής υποδιαστολής.

Οι πρώτοι επεξεργαστές μπορούσαν να χειριστούν άμεσα, μόνο ακέραιους αριθμούς. Ο χειρισμός των πραγματικών αριθμών γινόταν μέσω υπορουτίνων που φυσικά επιβάρυναν την ταχύτητα επεξεργασίας. Όταν χρειάζονταν περισσότερη μαθηματική δύναμη αναγκάζονταν να χρησιμοποιήσουν ένα μαθηματικό συνεπεξεργαστή (math coprocessor), μια μονάδα κινητής υποδιαστολής (Float Point Unit), που αναλάμβανε την εκτέλεση πολύπλοκων μαθηματικών υπολογισμών με μεγάλη ταχύτητα. Αργότερα ο συνεπεξεργαστής ήταν ενσωματωμένος στον επεξεργαστή.

Η χρησιμοποίηση του συνεπεξεργαστή αυξάνει την απόδοση του υπολογιστή όταν εκτελούνται προγράμματα που χρησιμοποιούν αριθμούς κινητής υποδιαστολής, όπως είναι τα προγράμματα τρισδιάστατων γραφικών, σαν το AutoCad. Στην περίπτωση όμως που εκτελούνται προγράμματα γραφείου, τα οποία δε χρησιμοποιούν αριθμούς κινητής υποδιαστολής, έτσι ώστε να χρησιμοποιείται ο συνεπεξεργαστής, η παρουσία του δεν αυξάνει την απόδοση του συστήματος.

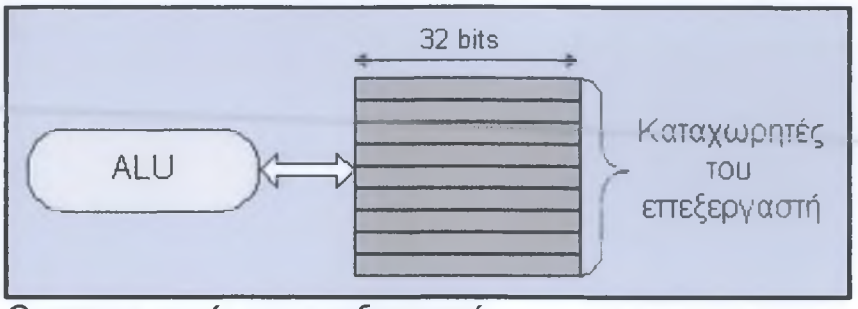
Πρώτου επιπέδου κρυφή μνήμη (L1 cache) και ελεγκτής κρυφής μνήμης

Όλοι οι σύγχρονοι επεξεργαστές έχουν μια μικρή, υψηλής ταχύτητας κρυφή μνήμη κοντά στο chip, για να κρατάει τα πιο πρόσφατα χρησιμοποιημένα δεδομένα και εντολές από τη μνήμη. Μια αρχή της επιστήμης των υπολογιστών που αποκαλείται τοπικότητα ισχυρίζεται ότι αν ο επεξεργαστής αναφέρθηκε σε μια περιοχή της μνήμης, είναι πολύ πιθανό να αναφερθεί ξανά σε αυτή πολύ σύντομα. Χρησιμοποιώντας μια κρυφή μνήμη για να κρατάει τις πιο πρόσφατα χρησιμοποιημένες τιμές της μνήμης, ο επεξεργαστής δε χρειάζεται κάθε φορά να πηγαίνει σε αυτή, για να τις φορτώσει. Αυτό παρέχει μια σημαντική αύξηση της απόδοσης, καθώς η μνήμη του συστήματος είναι πολύ πιο αργή από την κρυφή μνήμη του επεξεργαστή.

Η κρυφή μνήμη του επεξεργαστή ονομάζεται βασική (ή πρώτου επιπέδου) καθώς είναι η πιο κοντινή μνήμη στον επεξεργαστή. Κάθε φορά που ο επεξεργαστής απαιτεί πληροφορίες από τη μνήμη ο ελεγκτής της κρυφής μνήμης χρησιμοποιεί ειδικά κυκλώματα για να ελέγξει αν τα δεδομένα είναι ήδη στην κρυφή μνήμη. Αν είναι, ο επεξεργαστής εξοικονομεί το χρόνο της πρόσβασης στην κύρια μνήμη. Οι περισσότεροι επεξεργαστές χρησιμοποιούν

μετά την τιμή πίσω στη μνήμη. Αυτό φυσικά γίνεται πάρα πολύ γρήγορα και χωρίς να το καταλαβαίνει ο χρήστης.

Το πλάτος (σε bits) των καταχωρητών του επεξεργαστή καθορίζει πόσα δεδομένα μπορεί να επεξεργαστεί σε μια χρονική στιγμή. Αυτό ορισμένες φορές χρησιμοποιείται για να προσδιορίσει το "μέγεθος" του επεξεργαστή. Για παράδειγμα, μπορεί να ακούσετε να μιλούν για "16 bit επεξεργαστή" ή "32 bit επεξεργαστή". Αυτός ο όρος συνήθως αναφέρεται στο μέγεθος του καταχωρητή μέσα στον επεξεργαστή. Ωστόσο, αυτός ο όρος κάποιες φορές χρησιμοποιείται λανθασμένα και κάποιοι αναφέροντας το μέγεθος του επεξεργαστή, εννοούν για παράδειγμα το εύρος του διαύλου του.



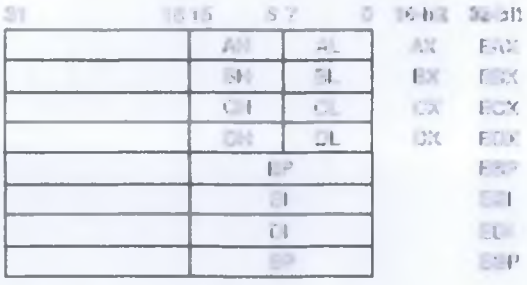
Οι καταχωρητές του επεξεργαστή

Όσους περισσότερους καταχωρητές έχει ο επεξεργαστής, τόσο μεγαλύτερη ευκινησία έχουν οι προγραμματιστές να γράψουν καλύτερο κώδικα. Ωστόσο, κάτι τέτοιο αυξάνει την πολυπλοκότητα του επεξεργαστή.

Στην ALU περιέχονται ένας μικρός αριθμός θέσεων μνήμης που λέγονται καταχωριστές, καθώς και εκείνα τα ηλεκτρονικά κυκλώματα που απαιτούνται για την εκτέλεση των πράξεων. Οι καταχωριστές που βρίσκονται στην ALU χρησιμοποιούνται για την προσωρινή καταχώρηση των τιμών, που πρόκειται να χρησιμοποιηθούν στους υπό εκτέλεση υπολογισμούς. Χρησιμοποιούνται επίσης και για προσωρινή καταχώριση των αποτελεσμάτων των υπολογισμός, πριν από τις καταχωρίσεις τους στην κεντρική μνήμη. Συνήθως, ένας ή περισσότεροι από τους καταχωριστές της ALU, που χρησιμοποιούνται στην εκτέλεση πράξεων, αλλά έχουν μεγαλύτερες δυνατότητες από τους υπόλοιπους καταχωριστές, ονομάζονται συσσωρευτές.

- Το μοντέλο καταχωρητών ειδικής χρήσης
- Καταχωρητές δεδομένων
    - EAX (AX (AH/AL)): Συσσωρευτής (*Accumulator*)
    - EBX (BX (BH/BL)): Βάση μνήμης
    - ECX (DX (CH/CL)): Μετρητής
    - EDX (DX (DH/DL)): Πολλαπλασιασμός, διαίρεση
- Καταχωρητές διευθύνσεων
    - SP: (stack pointer) Δείκτης στοίβας (*Stack pointer*)
    - BP: (base pointer) Βάση μνήμης

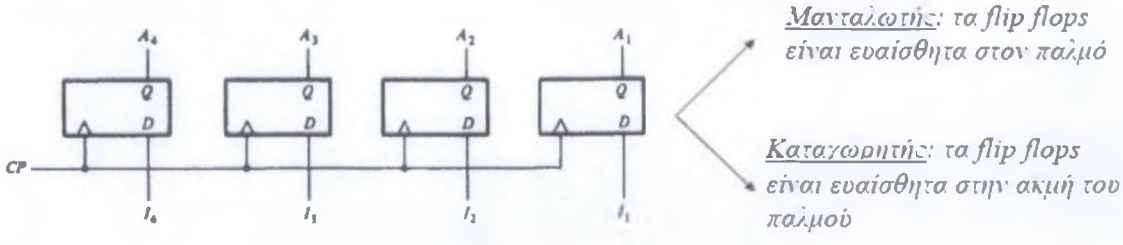
- SI: (source index) Δείκτης προέλευσης
- DI: (desination index) Δείκτης προορισμού



- Καταχωρητές τμημάτων μνήμης
  - CS: (code segment) Τμήμα εντολών
  - DS: (data segment) Τμήμα δεδομένων
  - SS: (stack segment) Τμήμα στοίβας
  - ES: (extra segment) Πρόσθετο τμήμα
  - FS: (extra segment) Πρόσθετο τμήμα
  - GS: (extra segment) Πρόσθετο τμήμα
- Καταχωρητές ελέγχου
  - IP: (instruction pointer) Μετρητής εντολής
  - FLAGS: Ενδείκτες (*Flags*)

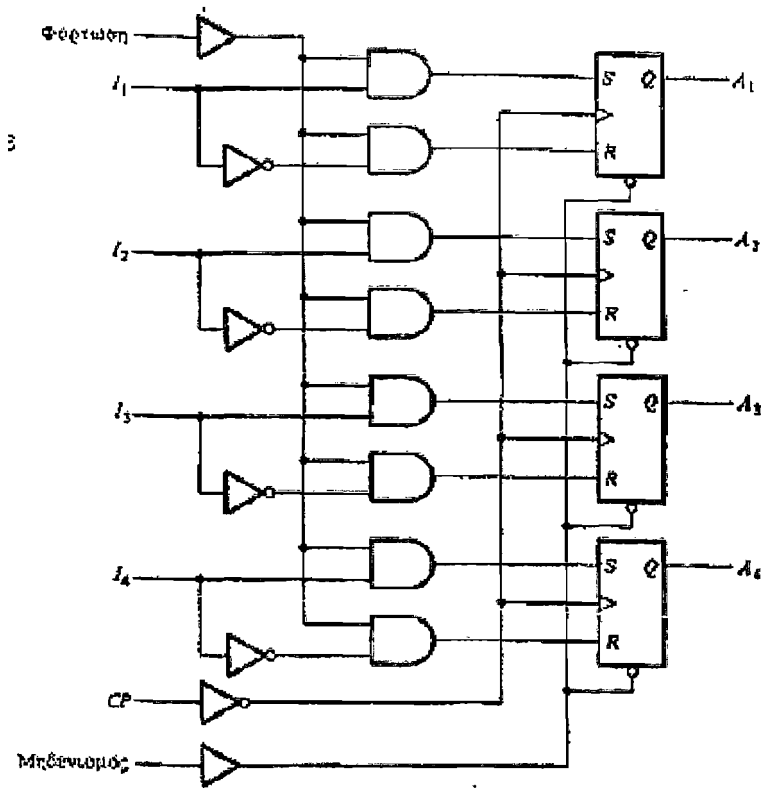
Καταχωρητής

Ο απλούστερος δυνατός τύπος καταχωρητή αποτελείται μονάχα από flip-flops χωρίς εξωτερικές πύλες.



4.3.1. Καταχωρητής με Παράλληλη Φόρτωση

Φόρτωση (loading): η μεταφορά πληροφοριών μέσα σε έναν καταχωρητή. **Παράλληλη** αν γίνεται σε όλα τα bits μαζί. **Σειριακή** αν γίνεται σε ένα ένα cell χωριστά.
Ο λογικός έλεγχος του ρολογιού (επίτρεψη/απόρριψη) δεν πρέπει να γίνεται με λογικές πύλες αλλά να χρησιμοποιούνται ασύγχρονες είσοδοι.
Ο απομονωτής στην είσοδο φόρτωσης τοποθετείται για λόγους μείωσης φορτίου.



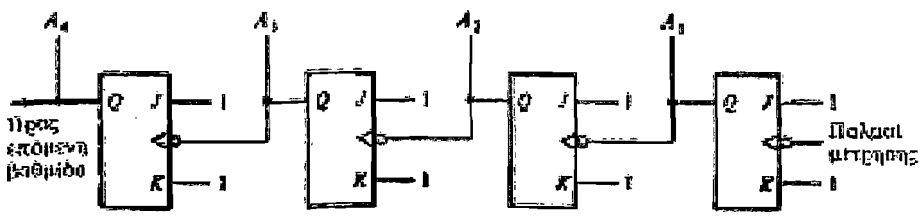
4.4. Μετρητές

Οι μετρητές είναι καταχωρητές που παίρνουν διαδοχικά μια προκαθορισμένη σειρά καταστάσεων.

Μετρητής Ριπής: Οι είσοδοι CP όλων των flip flops εκτός του πρώτου πυροδοτούνται από τις ακμές των κυματομορφών που βγαίνουν από τα άλλα flip flops.

Σύγχρονος Μετρητής: Οι είσοδοι CP όλων των flip flops τροφοδοτούνται από τους ίδιους παλμούς ρολογιού.

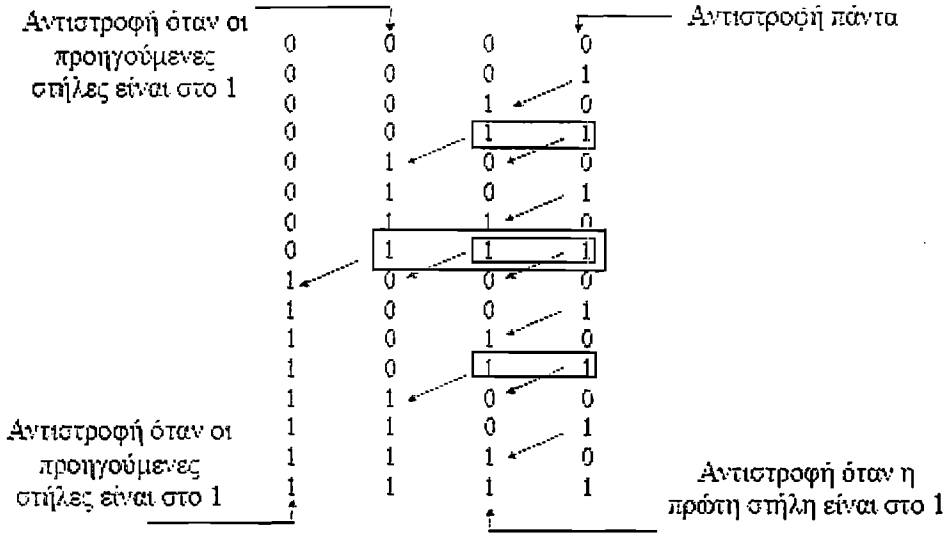
4.4.1. Μετρητές Ριπής



Ακολουθία μετρήσεων				
A_4	A_3	A_2	A_1	Συνθήκες αντιστροφής της κατάστασης των flip-flops
0	0	0	0	Αντιστρ. A_1
0	0	0	1	Αντιστρ. A_1. A_1 πάει από 1 σε 0, άρα αντιστρ. A_2
0	0	1	0	Αντιστρ. A_1
0	0	1	1	Αντιστρ. A_1. A_1 πάει από 1 σε 0, άρα αντιστρ. A_2. A_2 πάει από 1 σε 0, άρα αντιστρ. A_3
0	1	0	0	Αντιστρ. A_1
0	1	0	1	Αντιστρ. A_1. A_1 πάει από 1 σε 0, άρα αντιστρ. A_2
0	1	1	0	Αντιστρ. A_1
0	1	1	1	Αντιστρ. A_1. A_1 πάει από 1 σε 0, άρα αντιστρ. A_2. A_2 πάει από 1 σε 0, άρα αντιστρ. A_3. A_3 πάει από 1 σε 0, άρα αντιστρ. A_4

Αν πάρουμε την έξοδο μέτρησης από τους ακροδέκτες Q΄ τότε έχουμε έναν μετρητή προς τα κάτω. Άλλος τρόπος είναι να πυροδοτούνται τα flip flops από την θετική ακμή του ρολογιού.

4.4.2. Σύγχρονη Μέτρηση



4.4.3. Σύγχρονος μετρητής

Σύγχρονοι Μετρητές: Οι είσοδοι CP όλων των flip flops πυροδοτούνται από τον κοινό παλμό ρολογιού.
Στο δυαδικό μετρητή κάθε flip flop πρέπει να αντιστρέφεται μόνο στην ακμή του ρολογιού και όταν όλα τα λιγότερο σημαντικά ffs είναι στο 1.
Οι σύγχρονοι μετρητές έχουν ομοιόμορφη δομή και μπορούν να κατασκευαστούν εύκολα με πύλες και αντιστρέφοντα flip flops. Ο σύγχρονος μετρητής λειτουργεί το ίδιο και με πυροδότηση στη θετική ακμή.

4.5. Μονάδα ελέγχου

Η *μονάδα ελέγχου* είναι ένα κύκλωμα που ελέγχει τη ροή των πληροφοριών προς τον επεξεργαστή. Κατά κάποιο τρόπο είναι το "μυαλό μέσα στο μυαλό", καθώς ελέγχει ότι γίνεται μέσα στον επεξεργαστή, ο οποίος ελέγχει τον υπόλοιπο υπολογιστή.

Οι λειτουργίες που εκτελούνται από τη μονάδα ελέγχου ποικίλλουν δραστικά από την εσωτερική αρχιτεκτονική του επεξεργαστή, καθώς η μονάδα ελέγχου εκτελεί αυτή την αρχιτεκτονική. Σε ένα κανονικό επεξεργαστή, που εκτελεί τις x86 εντολές η μονάδα ελέγχου εκτελεί τις λειτουργίες της ανάκλησης, της αποκωδικοποίησης, της εκτέλεσης των εντολών και μετά την αποθήκευση των αποτελεσμάτων. Σε ένα επεξεργαστή με πυρήνα RISC, η μονάδα ελέγχου έχει σαφώς περισσότερο έργο να κάνει. Διαχειρίζεται τη μετατροπή των x86 εντολών σε RISC μικροεντολές και διαμοιράζει τις μικροεντολές μεταξύ των διαφόρων μονάδων εκτέλεσης. Σε ένα τέτοιο επεξεργαστή η μονάδα ελέγχου μπορεί να είναι διαχωρισμένη σε μικρότερες μονάδες εξαιτίας της πολυπλοκότητας της εργασίας που πρέπει να εκτελέσει.

Η μονάδα ελέγχου και η αριθμητική και λογική μονάδα αποτελούν μαζί την κεντρική μονάδα επεξεργασίας ενός ηλεκτρονικού υπολογιστικού συστήματος .

Οι βασικές λειτουργίες της μονάδας ελέγχου είναι :

- *Ο συντονισμός όλων των λειτουργιών με σήματα ελέγχου*
- *Ο προσδιορισμός της εκάστοτε επόμενης προς εκτέλεση εντολής*
- *Η μεταφορά της υπό εκτέλεση εντολής στη ΜΕ*
- *Η αποκωδικοποίηση της υπό εκτέλεση εντολής*
- *Η επίβλεψη των λειτουργιών κατά την εκτέλεση της εντολής*
- *Ο έλεγχος των διαδικασιών μνήμης , καταχώρησης και ανάκλησης.*

Η ομάδα ελέγχου είναι η οργανωτική δύναμη στον Η/Υ και αποτελείται από δύο κύριους καταχωρητές.

1. Καταχωρητής εντολών (instruction register): Είναι ένας καταχωρητής ο οποίος δέχεται όλες τις εντολές του προγράμματος από την μνήμη μια προς μια όπου αναγνωρίζονται και αναλύονται σε επί μέρους εργασίες και τέλος εκτελούνται.
2. Μετρητής προγράμματος (program counter): Ο μετρητής διαθέτει τη διεύθυνση της επόμενης εντολής προς εκτέλεση. Η εντολή μεταφέρεται στον καταχωρητή εντολών (IR) και μετά ο μετρητής προγράμματος θα αυξηθεί κατά 1 για να υποδείξει την επόμενη εντολή προς εκτέλεση.

Κατά την εκτέλεση ενός προγράμματος, η ΜΕ λαμβάνει από τη μνήμη ορισμένα κωδικοποιημένα ηλεκτρικά σήματα, τα οποία αντιστοιχούν σε *εντολές*. Τα κυκλώματα της μονάδας ελέγχου πρώτα ***"αποκωδικοποιούν"*** τις εντολές αυτές και στη συνέχεια εκπέμπουν σήματα ελέγχου προς:

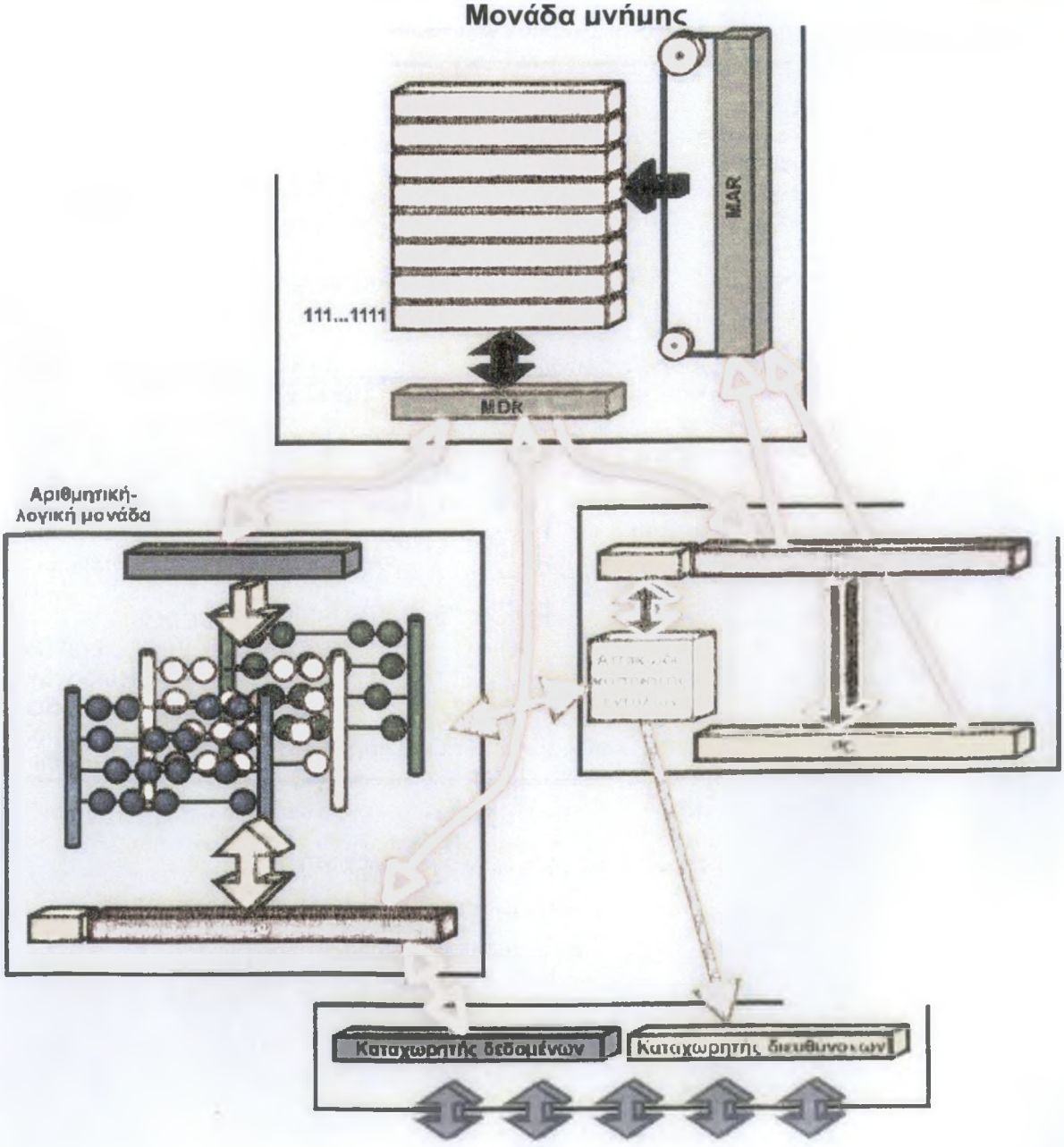
- Την μνήμη για την εξαγωγή πληροφοριών προς τις άλλες μονάδες του υπολογιστή ή την λήψη πληροφοριών απ΄αυτές και την αποθήκευσή τους στη μνήμη

- Την Α/Λ μονάδα για την εκτέλεση των επιθυμητών υπολογιστικών πράξεων

- Τις μονάδες εισόδου/εξόδου, για τη μεταφορά πληροφοριών από τον υπολογιστή προς το εξωτερικό του περιβάλλον, και αντιστρόφως

Κατά την εκτέλεση ενός προγράμματος που σύμφωνα με τα γνωστά , αποτελείται από μια ακολουθία από εντολές , πραγματοποιούνται τα εξής . Κάθε εντολή σύμφωνα με τη σειρά της μέσα στο πρόγραμμα μεταφέρεται στην κεντρική μονάδα επεξεργασίας . Στη συνέχεια αποκωδικοποιείται και αρχίζει να εκτελείται . Κατά την εκτέλεση παράγονται τα κατάλληλα σήματα ελέγχου τα οποία είναι διαφορετικά για κάθε εντολή και οδηγούν τη μηχανή στις αντίστοιχες ενέργειες .
Η διαδικασία αυτή επαναλαμβάνεται μέχρι και της τελευταίας εντολής του προγράμματος .

Η γενική οργάνωση του υπολογιστή φαίνεται και στο σχήμα που ακολουθεί.



4.6. Μνήμη Συστήματος

Η μνήμη του συστήματος (**system memory**), είναι ο χώρος στον οποίο αποθηκεύονται τα προγράμματα που "τρέχουν" σε μια δεδομένη χρονική στιγμή, καθώς και τα δεδομένα που χρησιμοποιούνται και παράγονται από αυτά τα προγράμματα. Ο όρος μνήμη (memory) είναι γενικός και αναφέρεται σε κάθε τμήμα του υπολογιστή, που μπορεί να αποθηκεύσει προσωρινά ή μόνιμα κάποιες πληροφορίες. Ωστόσο, ο όρος "μνήμη" αναφέρεται κυρίως στην **κύρια μνήμη του συστήματος (main system memory),** δηλαδή στο χώρο που αποθηκεύονται εντολές προς εκτέλεση από τον επεξεργαστή, καθώς και τα αντίστοιχα δεδομένα, που χρειάζονται για να εκτελεστούν σωστά οι παραπάνω εντολές. Η μνήμη του συστήματος παίζει σημαντικό ρόλο στη σωστή λειτουργία του υπολογιστή, αφού συνεισφέρει θετικά σε διάφορες παραμέτρους:

- **Απόδοση**: Το είδος και η ποσότητα της μνήμης του συστήματος επηρεάζει τη συνολική απόδοση του υπολογιστή. Για παράδειγμα, ανεπάρκεια στη μνήμη του συστήματος, μπορεί να αναγκάσει τον επεξεργαστή να εργάζεται μέχρι και 50% κάτω από την πραγματική του απόδοση.

- **Υποστήριξη Λογισμικού**: Τα νέα προγράμματα απαιτούν πολύ περισσότερη μνήμη σε σχέση με τα παλαιότερα. Έτσι, μεγαλύτερα ποσά μνήμης μας δίνουν τη δυνατότητα να "τρέξουμε" τα σύγχρονα προγράμματα που κυκλοφορούν

- **Αξιοπιστία και Σταθερότητα**: Ανεπαρκής ποσότητα μνήμης καθώς και λανθασμένος τύπος μπορεί να προκαλέσουν την εμφάνιση διαφόρων μυστήριων προβλημάτων στο σύστημα. Αν χρησιμοποιήσουμε μνήμη υψηλής ποιότητας στην κατάλληλη ποσότητα, το σύστημα θα λειτουργεί ομαλά και τα προβλήματα που σχετίζονται με τη μνήμη σχεδόν θα εκλείψουν. Ωστόσο, στην περίπτωση που το σύστημά μας δεν υποστηρίζει κάποιον τύπο μνήμης υψηλής ποιότητας, είναι σίγουρο ότι θα εμφανιστούν σημαντικά προβλήματα. Επομένως θα πρέπει να χρησιμοποιούμε τον κατάλληλο τύπο μνήμης και στην κατάλληλη ποσότητα, προκειμένου να λειτουργεί η μνήμη σύμφωνα με τις προδιαγραφές της.

- **Αναβάθμιση**: Στην αγορά κυκλοφορούν διαφόρων τύπων μνήμες, από τις οποίες άλλες είναι πάρα πολύ δημοφιλείς και άλλες λιγότερο. Αν υπάρχει πρόθεση για μελλοντική αναβάθμιση του συστήματος, προτείνεται η επιλογή της μνήμης να είναι τέτοια που θα μπορεί να χρησιμοποιηθεί και στο νέο σύστημα.

4.6.1. Μνήμη Ανάγνωσης Μόνο (ROM)

Η μνήμη **ROM (Read Only Memory- Μνήμη Μόνο Ανάγνωσης)** είναι ένας ειδικός τύπος μνήμης, τα περιεχόμενα της οποίας δε μεταβάλλονται. Έτσι, η μνήμη ROM είναι μνήμη μόνο ανάγνωσης, ενώ η κύρια μνήμη είναι μνήμη

ανάγνωσης αλλά και γραφής. Υπάρχουν δυο κυρίως λόγοι εξαιτίας των οποίων χρησιμοποιείται η ROM στους υπολογιστές:

- **Μονιμότητα:** Σε πολλές εφαρμογές, όπως στους υπολογιστές (BIOS), στις ηλεκτρικές συσκεύες, στα ηλεκτρονικά παιχνίδια κάποια δεδομένα πρέπει να παραμείνουν αποθηκευμένα ακόμα και μετά τη διακοπή της τροφοδοσίας. Για παράδειγμα, μέσα στο BIOS (Basic Input Output System) κάθε υπολογιστή είναι αποθηκευμένες δυαδικές πληροφορίες, που είναι απαραίτητες για την εκκίνησή του καθώς και για τον έλεγχο της σωστής λειτουργίας του. Οι πληροφορίες αυτές πρέπει να διατηρούνται πάντοτε, ανεξάρτητα από το αν ο υπολογιστής τροφοδοτείται ή όχι με ρεύμα. Η ROM, εξαιτίας του γεγονότος ότι διατηρεί τα περιεχόμενα της ονομάστηκε **μη πτητική μνήμη (non-volatile storage)**. Οι σκληροί δίσκοι είναι επίσης μη πτητική μνήμη, ενώ η κύρια μνήμη δεν είναι, αφού δε διατηρεί τα περιεχόμενά της μετά τη διακοπή της τροφοδοσίας του υπολογιστή με ρεύμα.

- **Ασφάλεια:** Το γεγονός ότι δεν μπορούμε να μεταβάλλουμε τα περιεχόμενα μιας μνήμης ROM, έχει ως αποτέλεσμα να αποτελεί ένα είδος προστασίας έναντι των εσκεμμένων (ή τυχαίων) απόπειρων μεταβολής των περιεχομένων της. Έτσι, υπάρχει προστασία έναντι των ιών (τεχνικά υπάρχει περίπτωση να μολυνθεί από ιό μόνο ένα είδος μνήμης ROM: η μνήμη EPROM).

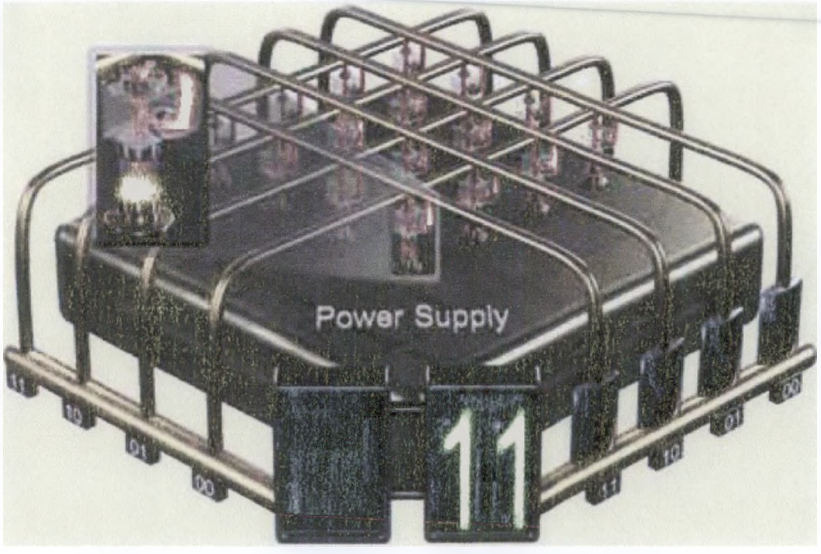
Όπως ειπώθηκε παραπάνω, η μνήμη ROM χρησιμοποιείται για την αποθήκευση προγραμμάτων χαμηλού επιπέδου, που πρέπει να είναι διαθέσιμα κάθε στιγμή. Το πιο χαρακτηριστικό παράδειγμα είναι το BIOS program, που είναι αποθηκευμένο σε ένα ειδικό chip μνήμης ROM, το οποίο είναι γνωστό και ως system BIOS ROM. Αν και το βασικό κριτήριο διαχωρισμού των μνημών ROM από τα υπόλοιπα είδη μνημών, είναι η μονιμότητα των περιεχομένων τους, υπάρχουν κάποιοι τύποι ROMs στους οποίους μπορούμε, ακολουθώντας ειδικές διαδικασίες, να μεταβάλλουμε τα περιεχόμενά τους. Τέλος, θα πρέπει να γίνει κατανοητό, ότι η ταχύτητα μεταφοράς δεδομένων από μια μνήμη ROM είναι σαφώς μικρότερη από αυτή μιας μνήμης RAM, γι΄ αυτό και τα περιεχόμενα της ROM αντιγράφονται στην κύρια μνήμη πριν χρησιμοποιηθούν για οποιοδήποτε σκοπό.

4.6.2. Τυπική ROM

Τα δεδομένα μιας **τυπικής ROM (Standard ROM)** εισάγονται κατά την κατασκευή της, με την έκθεση ενός φωτοευαίσθητου υλικού μέσω μιας μάσκας που περιέχει το επιθυμητό σχέδιο των bits και με την αφαίρεση στην συνέχεια της εκτεθειμένης (ή της μη εκτεθειμένης) επιφάνειας. Με τον τρόπο αυτό "καίγονται" κάποιοι σύνδεσμοι και αποτυπώνονται οι δυαδικές πληροφορίες. Από τη στιγμή που γίνει η εγγραφή της ROM, οι πληροφορίες δεν μπορούν να αλλάξουν, ενώ αν παρουσιαστεί τέτοια ανάγκη πρέπει να αντικατασταθεί το chip της ROM. Οι ROMs είναι πολύ φθηνότερες από τις RAMs, όταν γίνεται παραγγελία μεγάλων ποσοτήτων για την κάλυψη του κόστους κατασκευής της μάσκας. Δεν είναι όμως καθόλου ευέλικτες, αφού δεν μπορούν να τροποποιηθούν μετά την εγγραφή τους.

4.6.3. Προγραμματιζόμενη μνήμη ανάγνωσης

Για να διευκολυνθεί η ανάπτυξη νέων εφαρμογών, αναπτύχθηκαν οι PROMs (Programmable Read Only Memories - Προγραμματιζόμενες Μνήμες Μόνο Ανάγνωσης), που είναι πιο ευέλικτες, γιατί επιτρέπουν τον προγραμματισμό τους όχι μόνο από τον κατασκευαστή, αλλά και από το χρήστη. Ο χρήστης μπορεί να "γράψει" το πρόγραμμα του, χρησιμοποιώντας μια ειδική μηχανή γνωστή ως "προγραμματιστής PROM". Η μνήμη τοποθετείται στη μηχανή αυτή που έχει ήδη διαβάσει το πρόγραμμα και βραχυκυκλώνει (καίει) τους συνδέσμους (fusible links), που στη PROM παριστάνουν "1". Από τη στιγμή αυτή, οι πληροφορίες που αποθηκεύτηκαν παραμένουν αναλλοίωτες. Ο προγραμματισμός της PROM είναι γνωστός και ως burning και κατά κάποιο τρόπο είναι μια διαδικασία παρόμοια με την εγγραφή των CD-R.



Το κάψιμο των συνδέσμων σε μια μνήμη PROM

4.6.4. Διαγράψιμη PROM

Η μνήμη EPROM (Erasble PROMs - Διαγράψιμη PROM) είναι η εξέλιξη της απλής PROM. Το χαρακτηριστικό των EPROMs είναι, ότι μπορούν να προγραμματιστούν πολλές φορές, διατηρώντας πάντα τα χαρακτηριστικά μιας PROM. Έτσι, όταν το κρυσταλλικό πλαίσιο μιας EPROM εκτίθεται σε ισχυρή υπεριώδη ακτινοβολία για 15 λεπτά περίπου, όλα τα bits που περιέχει γίνονται ίσα με "1". Στη συνέχεια, μπορούν να γραφτούν πληροφορίες σε αυτές με τη βοήθεια ενός προγραμματιστή PROM. Οι EPROMs είναι πολύ φθηνότερες από τις PROMs γιατί μπορούν να ξαναχρησιμοποιηθούν. Οι EPROMs κατά κάποιο τρόπο μοιάζουν με τα CD-RW, αφού μπορούμε να σβήσουμε τα περιεχόμενά τους και να γράψουμε νέες πληροφορίες.

Προγραμματιστής PROM

4.6.5. Ελεγκτής μνήμης

Σε κάθε υπολογιστή υπάρχει ένα ολοκληρωμένο κύκλωμα που ονομάζεται **ελεγκτής μνήμης** (**memory controller**) και ελέγχει τις λειτουργίες που σχετίζονται με τη μνήμη. Παράγει τα απαραίτητα σήματα για να γίνονται σωστά οι λειτουργίες της ανάγνωσης / γραφής στη μνήμη και συνδέει τη μνήμη με τα υπόλοιπα μέρη του υπολογιστή. Συνήθως, ο ελεγκτής μνήμης είναι ενσωματωμένος στο chipset συστήματος.

Προσπέλαση μνήμης και χρόνος προσπέλασης

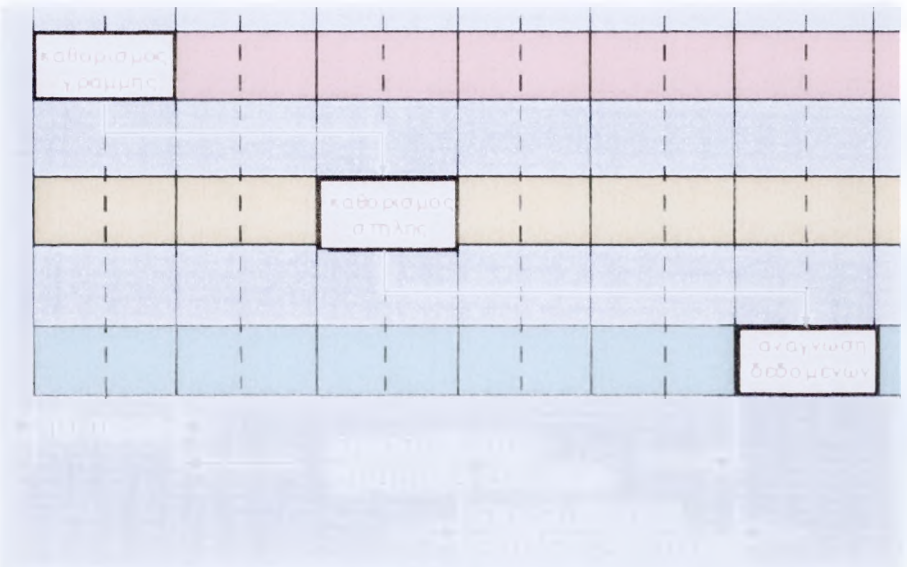
Όταν διαβάζουμε ή γράφουμε στη μνήμη, λέμε ότι κάνουμε μια προσπέλαση (access) σε αυτή. Κάθε προσπέλαση στη μνήμη ελέγχεται από μια ειδική διαδικασία του ελεγκτή της μνήμης, η οποία παράγει τα σήματα που καθορίζουν πιο σημείο της μνήμης θα προσπελαστεί και τι είδους προσπέλαση (ανάγνωση / γραφή) θα γίνει.

Για να κατανοήσουμε πως γίνεται η προσπέλαση στη μνήμη, θα πρέπει να αναφέρουμε τον τρόπο διευθυνσιοδότησης μέσα στο chip της. Έστω λοιπόν, ότι διαθέτουμε συνολικά 16 Mbit μνήμης, στη μορφή 4 Mbit *4. Αυτό λογικά σημαίνει ότι, υπάρχουν 4M=4194304 διαφορετικές διεθύνσεις (γραμμές), ενώ κάθε γραμμή έχει τέσσερα διαφορετικά κύτταρα αποθήκευσης. Για τη διευθυνσιοδότηση 4194304=222 διαφορετικών θέσεων, θα πρέπει οι δυαδικές διευθύνσεις να έχουν μήκος 22 bits.

Ωστόσο, στην πράξη δεν υπάρχουν τόσες πολλές διευθύνσεις (γραμμές) μέσα στο chip της μνήμης. Θα μπορούσαμε να φανταστούμε τη μνήμη, ως ένα διδιάστατο πίνακα (n γραμμές και m στήλες) κελιών, όπου σε κάθε κελί αποθηκεύεται η ελάχιστη πληροφορία, δηλαδή ένα bit "0" ή "1". Κάθε γραμμή του πίνακα έχει μια ξεχωριστή "διεύθυνση", που βοηθάει στην άμεση προσπέλαση της πληροφορίας, η οποία είναι αποθηκευμένη στα στοιχεία (κελιά) της γραμμής. Σε μια διεύθυνση των 22 bits τα 11 τελευταία bits χρησιμοποιούνται για να καθορίσουν τη γραμμή και τα πρώτα 11 bits για να καθορίσουν τη στήλη. Έστω για παράδειγμα ότι γίνεται μια αναφορά στη θέση μνήμης 2871405, που αντιστοιχεί στη δυαδική διεύθυνση "10101111010 00001101101". Πρώτα θα σταλούν τα 11 τελευταία bits "00001101101" που

καθορίζουν τη διεύθυνση της γραμμής. Στην συνέχεια, θα σταλούν τα 11 πρώτα bits "10101111010" για τον καθορισμό της στήλης.

Με μια πρώτη ματιά, φαίνεται ότι ο σχεδιασμός της μνήμης, βάσει του οποίου κατακερματίζουμε τη διεύθυνση, είναι πιο περίπλοκος, από το να χρησιμοποιούμε ολόκληρη τη διεύθυνση των 22bits για να καθορίσουμε μια γραμμή. Ο λόγος για τον οποίο γίνεται ο κατακερματισμός της διεύθυνσης, είναι καθαρά οικονομικός. Αν χρησιμοποιούσαμε και τα 22 bits για τη διεύθυνση κάθε γραμμής, το chip DRAM της μνήμης θα είχε 22 pins. Αντί για 22 pins, έχουμε 22-11=11 pins. Επίσης, θα χρειάζονται πλέον μόνο 11 σήματα ελέγχου, αντί για 22. Βέβαια, το να αποστέλεται η διεύθυνση σε δυο δόσεις, προκαλεί μια καθυστέρηση. Φυσικά οι υπολογιστές δεν διαθέτουν ένα μοναδικό chip μνήμης, αλλά πολλά, το πλήθος των οποίων εξαρτάται από το συνολικό μέγεθος της DRAM. Τα chips οργανώνονται σε modules, ενώ το αμέσως ανώτερο επίπεδο οργάνωσης είναι τα banks.



Διάγραμμα χρονισμού για μια διαδικασία ανάγνωσης από τη κύρια μνήμη

Παρακάτω περιγράφεται με τη μορφή βημάτων, η διαδικασία ανάγνωσης από μια ασύγχρονη μνήμη DRAM. Ωστόσο, η νέας τεχνολογίας μνήμες DRAM, είναι συγχρονισμένες με τα κύριο ρολόι του συστήματος. Για την καλύτερη κατανόηση της διαδιακασίας θα υποθέσουμε ότι έχουμε το μοντέλο μνήμης DRAM που αναφέρθηκε παραπάνω, δηλαδή, στη διεύθυνση των 22 bits τα 11 τελευταία bits χρησιμοποιούνται για να καθορίσουν τη γραμμή και τα πρώτα 11 bits για να καθορίσουν τη στήλη:
Η διεύθυνση της μνήμης που πρόκειται να προσπελαστεί για ανάγνωση τοποθετείται στο δίαυλο των διευθύνσεων (address bus).

Ο ελεγκτής της μνήμης αποκωδικοποιεί τη διεύθυνση και καθορίζει, πιο chip της περιέχει τη διεύθυνση που θα προσπελαστεί.

Τα bits της διεύθυνσης που καθορίζουν τη "γραμμή" (τα 11 τελευταία bits) στέλνονται στο κατάλληλο chip.

Αφού περάσει αρκετός χρόνος μέχρι να σταθεροποιηθεί το σήμα που καθορίζει τη επιλεγόμενη γραμμή, ο ελεγκτής μνήμης παράγει ένα ειδικό σήμα που ονομάζεται row address strobe ή row address select ("/RAS") και έχει τιμή μηδέν.
Όταν το σήμα /RAS σταθεροποιηθεί στην τιμή μηδέν "0", έχει επιλεχθεί πλέον η κατάλληλη γραμμή και διαβάζονται τα περιεχόμενα της (όλες οι στήλες), από τα ειδικά κυκλώματα του chip. Ας σημειωθεί ότι αυτή η διαδικασία ανάγνωσης των περιεχομένων της γραμμής, αναζωογωνούν ταυτοχρόνως και τα περιεχόμενά της.

Τα bits της διεύθυνσης που καθορίζουν τη "στήλη" (τα 11 πρώτα bits) στέλνονται στο κατάλληλο chip.

Αφού περάσει αρκετός χρόνος μέχρι να σταθεροποιηθεί το σήμα που καθορίζει την επιλεγόμενη στήλη, ο ελεγκτής μνήμης παράγει ένα ειδικό σήμα που ονομάζεται column address strobe ή column address select ("/CAS") και έχει τιμή μηδέν.

Όταν το σήμα /CAS σταθεροποιηθεί στην τιμή μηδέν "0", έχει επιλεχθεί πλέον η κατάλληλη στήλη και το περιεχόμενο του επιλεγόμενου κελιού οδηγείται στον buffer εξόδου του chip.

Ο buffer εξόδου του chip, τοποθετεί το περιεχόμενό του, στο δίαυλο δεδομένων της μνήμης από όπου μεταφέρονται στη συσκευή που έκανε την αίτηση ανάγνωσης.

Η διαδιακασία προσπέλασης / ανάγνωσης των περιεχομένων ενός κελιού της μνήμης, παρουσιάζει ομοιότητες με τον τρόπο επιλογής ενός κελιού (για παράδειγμα το κελί J34) στα λογιστικά φύλλα. Πήγαινε στη γραμμή #34 και στη συνέχεια επέλεξε το κελί της J στήλης.

Η διαδικασία γραφής σε μια ασύγχρονη μνήμη DRAM, ακολουθεί τα ίδια περίπου βήματα με τη διαφορά ότι, οι πληροφορίες μεταφέρονται προς τα chips της μνήμης. Χρησιμοποιώντας ένα ειδικό σήμα ελέγχου "R/W" (Read/Write), μπορούμε να καθορίσουμε πότε θα γίνεται ανάγνωση και πότε γραφή στα chips της μνήμης. Η διαδικασία ανάγνωσης / γραφής που περιγράφτηκε παραπάνω, είναι πάρα πολύ απλουστευμένη για λόγους κατανόησης, αφού στην πράξη συνυπάρχουν και άλλοι παράγοντες που ρυθμίζουν τη λειτουργία του chip.

Ο χρόνος που απαιτείται από τη στιγμή που παρουσιάζεται μια αίτηση ανάγνωσης προς τη μνήμη, μέχρι να είναι διαθέσιμες οι αντίστοιχες πληροφορίες, είναι γνωστός ως χρόνος προσπέλασης (access time ή tAC) και μετράται σε nanoseconds (ns).

4.6.6. Σύγχρονη και Ασύγχρονη DRAM

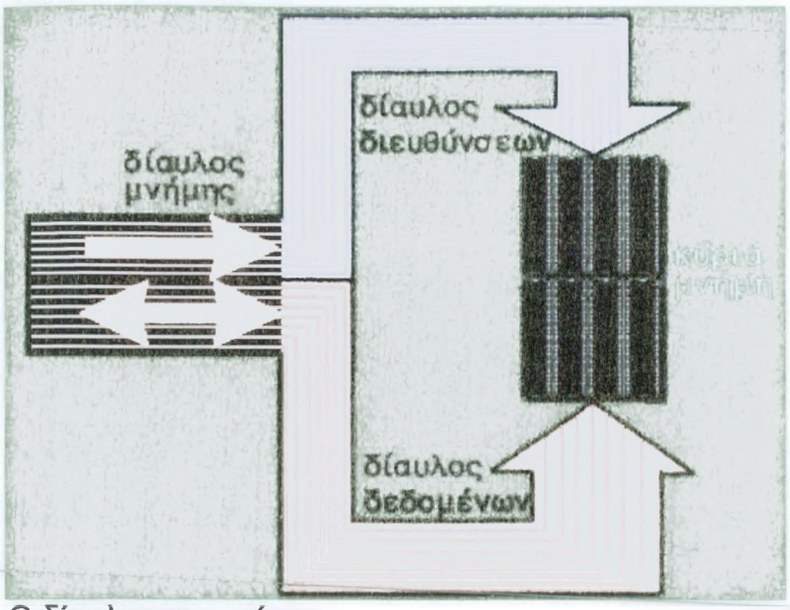
Η τυπική DRAM που χρησιμοποιήθηκε στους αρχικούς IBM PC, χαρακτηριζόταν ως ασύγχρονη (**asynchronous DRAM**). Ο όρος ασύγχρονη έχει να κάνει με το γεγονός ότι, η μνήμη δεν είναι συγχρονισμένη με το κύριο ρολόι του συστήματος (system clock). Αν υποθέσουμε ότι αυτή τη στιγμή αρχίζει μια προσπέλαση στη μνήμη, μετά από κάποιο χρονικό διάστημα θα εμφανιστούν τα περιεχόμενα της μνήμης, στα οποία έγινε η προσπέλαση στο δίαυλο. Τα σήματα που παράγονται από τον ελεγκτή της μνήμης, δεν είναι συγχρονισμένα με το ρολόι του συστήματος. Η ασύγχρονη μνήμη δουλεύει μια χαρά για συστήματα μικρής σχετικά ταχύτητας, αλλά δεν ενδείκνυται για ταχύτερα συστήματα (**>66 MHz**).

Οι νέοι τύποι DRAM, ονομάζονται "**synchronous DRAM**" ή "**SDRAM**", αφού τα σήματα που διαχειρίζονται και παράγουν είναι συγχρονισμένα με το ρολόι του συστήματος. Η σύγχρονη DRAM είναι πολύ ταχύτερη της ασύγχρονης DRAM, πράγμα που συμβάλλει στη βελτίωση της απόδοσης του συστήματος. Η SDRAM είναι κατάλληλη για συστήματα μεγάλης ταχύτητας.

4.6.7. Δίαυλος μνήμης

Ο δίαυλος της μνήμης (memory bus), είναι ένα σύνολο καλωδίων που χρησιμοποιούνται για τη μεταφορά της διεύθυνσης προς τη μνήμη, καθώς και για τη μεταφορά των δεδομένων από και προς τη κύρια μνήμη. Σε πολλούς υπολογιστές, ο δίαυλος της μνήμης είναι κοινός με το δίαυλο του επεξεργαστή.

Ο δίαυλος της μνήμης αποτελείται από δυο τμήματα: το δίαυλο των δεδομένων (data bus) και το δίαυλο των διευθύνσεων (address bus). Συνήθως, όταν αναφερόμαστε στο δίαυλο της μνήμης, εννοούμε το δίαυλο των δεδομένων, ο οποίος χρησιμοποιείται για την πραγματική μεταφορά των δεδομένων από και προς τη μνήμη. Ο δίαυλος των διευθύνσεων, χρησιμοποιείται για τη μεταφορά της διεύθυνσης στη μνήμης, όπου θα πραγματοποιηθεί μια προσπέλαση για ανάγνωση ή γραφή. Όσο μεγαλύτερος είναι ο δίαυλος των δεδομένων (περισσότερα καλώδια), τόσα περισσότερα δεδομένα μπορούν να μεταφερθούν ταυτόχρονα, με αποτέλεσμα να βελτιώνεται η απόδοση όλου του συστήματος. Το εύρος ζώνης (bandwidth) του διαύλου των δεδομένων, μετρά την ποσότητα της πληροφορίας που μπορεί να μεταφερθεί μέσω των καλωδίων του και είναι συνάρτηση του αριθμού των bits που μπορούν να μεταφερθούν ταυτόχρονα και της ταχύτητας μεταφοράς των δεδομενών.



Ο δίαυλος της μνήμης

Θα μπορούσαμε να φανταστούμε τον δίαυλο των δεδομένων σαν ένα δρόμο ταχείας κυκλοφορίας. Το εύρος του είναι το πλήθος των διαφορετικών λωρίδων που έχει, ενώ η ταχύτητά του, καθορίζει το πόσο γρήγορα μπορούν να κινηθούν τα αυτοκίνητα. Το εύρος ζώνης του αυτοκινητόδρομου, είναι ο μέγιστος αριθμός των αυτοκινήτων που θα περάσουν σε ένα ορισμένο χρονικό διάστημα, υπό το καθεστώς των περιορισμών στον αριθμό των διαθέσιμων λωρίδων και στην επιτρεπόμενη ταχύτητα.
Το εύρος του διαύλου των διευθύνσεων καθορίζει το μέγιστο αριθμό διαφορετικών διευθύνσεων που μπορεί να αναγνωρίσει ο επεξεργαστής. Στο παραπάνω παράδειγμα με το δρόμο ταχείας κυκλοφορίας, μπορούμε να πούμε ότι το πλήθος των διαφορετικών εξόδων από τον αυτοκινητόδρομο, είναι κάτι ανάλογο με το εύρος του διαύλου των διευθύνσεων.

4.6.8. Ταχύτητα μνήμης και κρυφή μνήμη του συστήματος

Η χρησιμοποίηση σχετικά μεγάλων ποσοτήτων κρυφής μνήμης, έχει ως αποτέλεσμα την ικανοποίηση του μεγαλύτερου μέρους των αιτήσεων για την κύρια μνήμη, από την κρυφή μνήμη. Αν χρησιμοποιήσουμε ταχύτερη μνήμη, βελτιώνεται η απόδοση του συστήματος, αλλά σε μικρό ποσοστό και αυτό γιατί οι περισσότερες αιτήσεις ικανοποιούνται από την κρυφή μνήμη και έτσι δεν εκμεταλλευόμαστε την ταχύτερη πλέον κύρια μνήμη. Για παράδειγμα, αν η ταχύτητα της κύριας μνήμης αυξηθεί κατά 50%, τότε η απόδοση του συστήματος θα αυξηθεί κατά 2.5-5% μόνο. Εξαιτίας του παραπάνω φαινομένου, δεν βελτιώνεται σημαντικά η απόδοση ενός συστήματος στο οποίο έχουμε τοποθετήσει SDRAM.

4.6.9. Προσπέλαση μνήμης με τη μέθοδο της ριπής

Μπορούμε να φανταστούμε τη μνήμη σαν ένα διδιάστατο πίνακα, όπου σε κάθε στοιχείο (κύτταρο-cell) του μπορούμε να αποθηκεύσουμε ένα bit

πληροφορίας. Για να κάνουμε μια προσπέλαση σε ένα κύτταρο, πρέπει πρώτα να δώσουμε τη διεύθυνση της γραμμής και στη συνέχεια τη διεύθυνση της στήλης. Στην πράξη δεν διαβάζουμε ποτέ ένα μόνο bit, αλλά ομάδες των 32 ή 64 bits.

Υπάρχει ένας αριθμός απαραίτητων βημάτων που πρέπει να εκτελεστούν κατά την πρώτη προσπέλαση στη μνήμη, με αποτέλεσμα να υπάρχει μια επιπλέον καθυστέρηση κατά τη διάρκεια αυτής. Παράγονται αρκετά σήματα που πρέπει να σταθεροποιηθούν και να σταλούν σε διάφορα μέρη του υπολογιστή προκειμένου να αρχίσει η προσπέλαση στη μνήμη. Στη συνέχεια, στέλνεται η διεύθυνση της γραμμής και μετά η διεύθυνση της στήλης ,προκειμένου να καθοριστεί από πιο κελί θέλουμε να διαβάσουμε ή να γράψουμε. Εξαιτίας όλων αυτών των καθυστερήσεων, η πρώτη προσπέλαση στη μνήμη διαρκεί από 4 ως 7 παλμούς. Ο χρόνος που απαιτείται, για να ικανοποιηθεί η πρώτη αίτηση για τη μνήμη, είναι γνωστός και ως λανθάνων χρόνος προσπέλασης (latency) της μνήμης.

Όπως είπαμε, οι καθυστερήσεις που σημειώνονται μέσα στη μνήμη, έχουν να κάνουν κυρίως με διάφορα σήματα και όχι με την πραγματική μεταφορά δεδομένων. Αν μπορούσαμε να μειώσουμε τις καθυστερήσεις λόγω σημάτων, η απόδοση του συστήματος θα βελτιωνόταν σημαντικά. Επίσης, αν θέλουμε να διαβάσουμε 4 διαδοχικά blocks των 64 bits, θα χρειαστεί να δώσουμε τη διεύθυνση μόνο του πρώτου. Με τον τρόπο αυτό, μειώνονται οι καθυστερήσεις που έχουν να κάνουν με την αποστολή της διεύθυνσης και έτσι αυξάνεται η απόδοση του συστήματος. Στα σύγχρονα συστήματα ακολουθείται αυτή η μέθοδος που είναι γνωστή ως προσπέλαση κατά ριπή (burst mode access ή bursting). Έτσι, η καθυστέρηση της πρώτης προσπέλασης δεν επαναλαμβάνεται και στις 3 επόμενες προσβάσεις. Επομένως, για την πρώτη προσπέλαση θα χρειαστούμε 4-7 παλμούς και για κάθε μια από τις 3 επόμενες μόνο 1-3 παλμούς. Η δευτερεύουσα κρυφή μνήμη (system secondary cache) του συστήματος χρησιμοποιεί αυτή την τεχνική και διαβάζει κατά ομάδες των 256 bits, με αποτέλεσμα να είναι ταχύτατη.

Ο χρονισμός μιας μνήμης με πρόσβαση ριπής εκφράζεται με τη συντομογραφία "x-y-y-y". Ο πρώτος αριθμός ("x") αναπαριστά τον αριθμό των παλμών ρολογιού που απαιτούνται για την πρώτη προσπέλαση (για ανάγνωση ή γραφή) στη μνήμη. Οι άλλοι τρεις αριθμοί ("y-y-y"), είναι ο αριθμός των παλμών για τη δεύτερη, τρίτη και τέταρτη προσπέλαση στη μνήμη. Για παράδειγμα, ο χρονισμός "5-2-2-2", δηλώνει ότι θα απαιτηθούν συνολικά 11 παλμοί για να ολοκληρωθεί η προσπέλαση και στα 4 blocks των 64 bits, ενώ αν δεν χρησιμοποιούνταν η τεχνική του burst mode access θα χρειαζόταν τουλάχιστον 20 παλμοί.

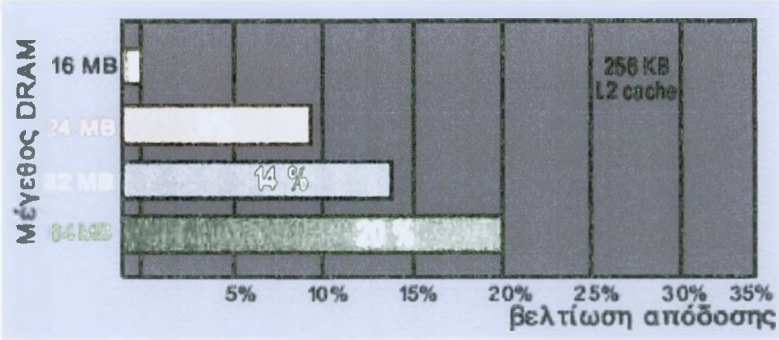
Όλοι οι τύποι μνημών παρουσιάζουν την επιπλέον καθυστέρηση της πρώτης προσπέλασης. Από τον κανόνα αυτό δεν ξεφεύγουν φυσικά οι ταχύτατες SDRAM καθώς επίσης και η δευτερεύουσα κρυφή μνήμη. Ο χρονισμός για αυτές τις μνήμες ακολουθεί το μοντέλο "x-1-1-1", δηλαδή οι προσβάσεις στη μνήμη εκτός της πρώτης, διαρκούν ένα παλμό. Ωστόσο, η ανάγκη για διευθυνσιοδότηση της SDRAM έχει ως αποτέλεσμα ο πρώτος αριθμός στην ακολουθία "x-1-1-1" να είναι μεγαλύτερος από τον αντίστοιχο αριθμό για την

ακολουθία της δευτερέουσας κρυφής μνήμης. Έτσι, ένας καλός συνδυασμός χρονισμού για την SDRAM είναι "5-1-1-1", ενώ για τη δευτερέουσα κρυφή μνήμη "3-1-1-1" ή "2-1-1-1".

4.7. Μέγεθος μνήμης

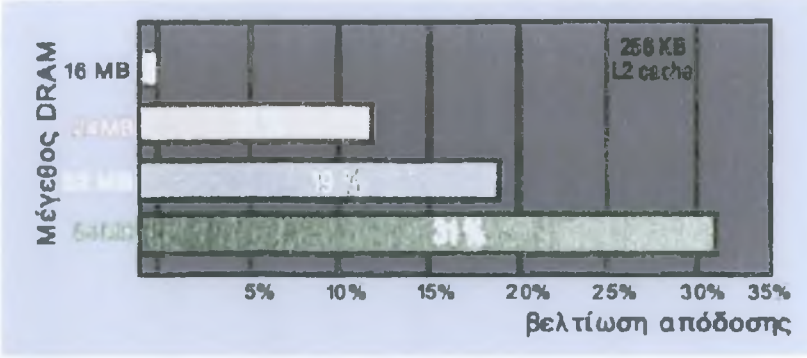
4.7.1. Μέγεθος μνήμης και απόδοση συστήματος

Το μέγεθος της κύριας μνήμης, που διαθέτει ένας υπολογιστής, παίζει σημαντικό ρόλο στη διαμόρφωση της συνολικής απόδοσης του συστήματος. Επομένως, ανάλογα με το σύστημα που διαθέτουμε και το σκοπό για τον οποίο θα το χρησιμοποιούμε (είδος προγραμμάτων που χρησιμοποιούμε), πρέπει να έχουμε εγκαταστήσει τον κατάλληλο τύπο μνήμης στην κατάλληλη ποσότητα.



Η βελτίωση της απόδοσης σε ένα σύστημα Pentium 100MHz

με λειτουργικό σύστημα Windows 95 για διάφορα μεγέθη DRAM



Η βελτίωση της απόδοσης σε ένα σύστημα Pentium Pro 200MHz

με λειτουργικό σύστημα Windows 95 για διάφορα μεγέθη DRAM

Το μέγεθος της μνήμης δεν επηρεάζει την ταχύτητά της, ούτε επίσης την ταχύτητα του επεξεργαστή, των chipsets, της μητρικής πλακέτας καθώς και άλλων εξαρτημάτων του υπολογιστή. Αυτό ισχύει μόνο στην περίπτωση που όλα τα προγράμματα που "τρέχουν" ταυτοχρόνως χωράνε στην κύρια μνήμη του συστήματος. Στην περίπτωση συστημάτων πολυεργασίας, χρησιμοποιούμε εικονική μνήμη. Αν το μέγεθος της κύριας μνήμης είναι πολύ μικρό σε σχέση με τις πραγματικές απαιτήσεις, θα έχουμε πολλές μεταφορές σελίδωνμεταξύ του σκληρού δίσκου και της κύρια μνήμης, οπότε η επιπλέον καθυστέρηση θα ρίχνει σημαντικά την απόδοση του συστήματος. Ο καλύτερος τρόπος για να εκτιμήσουμε τη σχέση μεταξύ του μεγέθους της κύριας μνήμης και της συνολικής απόδοσης του συστήματος, είναι να συγκρίνουμε ίδια συστήματα που εκτελούν τις ίδιες εργασίες και έχουν διαφορετικές ποσότητες μνήμης (του ίδιου τύπου).

4.7.2. Μέγιστη ποσότητα μνήμης και κρυφής μνήμης

Θα πρέπει να γίνει κατανοητό ότι δεν μπορούμε να χαρτογραφήσουμε όλη την κύρια μνήμη στην κρυφή μνήμη. Παράδειγμα αποτελούν τα chipsets της Intel 430FX, 430VX, 430HX και 430TX. Από αυτά τα chipsets, μόνο το 430HX υποστηρίζει 128 MB κύριας μνήμης, από τα οποία μόνο 64 MB μπορούν να γραφούν στην κρυφή μνήμη.

4.7.3. Πραγματική και Εικονική μνήμη

Η **Real memory**, είναι η **πραγματική** μνήμη που έχουμε εγκαταστήσει στο σύστημά μας. Ωστόσο, μερικές φορές είναι καλό να "ξεγελούμε" τον υπολογιστή και να τον κάνουμε να "πιστεύει" ότι έχει στη διάθεσή του περισσότερη μνήμη, από αυτή που πραγματικά διαθέτει. Ιδιαίτερα σε περιπτώσεις όπου τρέχουμε μεγάλα προγράμματα ή σε περιπτώσεις πολυεργασίας (multitasking), η μέθοδος αυτή είναι χρήσιμη και έχει ονομαστεί **εικονική μνήμη (virtual memory).**

Η εικονική μνήμη λειτουργεί κατά τον εξής απλό τρόπο: Ας υποθέσουμε ότι το λειτουργικό σύστημα χρειάζεται 80 MB για να κρατήσει όλα τα προγράμματα που τρέχουν, αλλά υπάρχουν μόνο 32 MB πραγματικής κύριας μνήμης. Το λειτουργικό σύστημα υποθέτει ότι είναι διαθέσιμα 80 MB εικονικής μνήμης, οπότε "καλεί" ένα πρόγραμμα που διαχειρίζεται την εικονική μνήμη και είναι γνώστο ως **διαχειριστής εικονικής μνήμης (virtual memory manager)**. Στη συνέχεια, δημιουργείται ένα **αρχείο ανταλλαγής (swap file)** στο σκληρό δίσκο, με μέγεθος 80-32=48 MB, που ανταλλάσει τα περιεχόμενά του με αυτά της πραγματικής κύριας μνήμης. Το λειτουργικό σύστημα πλέον "βλέπει" 80 MB μνήμης, ενώ ο διαχειριστής εικονικής μνήμης είναι επιφορτισμένος με τη διαχείριση της πραγματικής μνήμης των 32 MB.

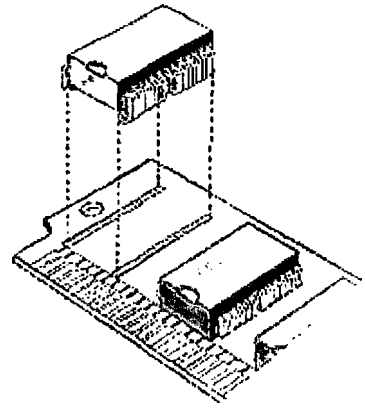
Φυσικά, από τα 80 MB που απαιτούνται, μόνο τα 32 MB είναι αποθηκεύμενα στην κύρια μνήμη, ενώ τα υπόλοιπα 48 MB βρίσκονται στο αρχείο ανταλλαγής του δίσκου. Όταν το λειτουργικό σύστημα χρειάζεται ένα τμήμα της μνήμης που όμως βρίσκεται στο αρχείο του δίσκου, ο διαχειριστής εικονικής μνήμης διαβάζει αυτό το τμήμα από τα αρχείο και το αντιγράφει στην κύρια μνήμη, στη θέση ενός block δεδομένων που δεν έχουν χρησιμοποιηθεί πρόσφατα. Τα

creep. Επομένως, έπρεπε κάθε φορά να ανοίγουμε τον υπολογιστή και να επανασυνδέουμε τα DIPs. Κλασσικό παράδειγμα αποτελούσαν τα παλιά συστήματα XT.

Στα σύγχρονα συστήματα τα DIPs δε συνδέονται απευθείας στη μητρική πλακέτα, αλλά συνδυαζόταν πολλά μαζί μέσα σε ένα μικρό κύκλωμα, που ονομάζεται **memory module**. Υπάρχουν δυο βασικοί τύποι memory module: **DIMM (Dual Iniline Memory Module) και SIMM (Single Iniline Memory Module)**. Αυτά τα κυκλώματα που περιείχαν τα DIPs, εισάγονταν σε ειδικές υποδοχές, που είχαν σχεδιαστεί έτσι ώστε να περιορίζεται κατά το δυνατόν το φαινόμενο chip creep.

4.8.2. SIMMs

Τα SIMMs είναι ο πιο κοινός τύπος memory module που χρησιμοποιείται στο χώρο των προσωπικών υπολογιστών. Η αρχική έκδοση των SIMMs modules ήταν των 8 bits πάνω σε μια μικρή κάρτα που έδινε 1, 2, ή 4 MB RAM. Συνδεόταν στη μητρική πλακέτα με έναν 30-pin συνδετήρα. Επειδή το κάθε module διέθετε 8 bits, οι 16-bit επεξεργαστές (286 και 386SX) χρειάζονταν 2 SIMMs σε ένα ζεύγος. Εξαιτίας αυτού υπήρχε χώρος για 2 modules που αποκαλούνταν banks. Οι επεξεργστές των 32 bit (386DX και 486) απαιτούσαν 4 SIMMs των 8 bits σε κάθε bank, αφού τα δικά τους banks είχαν εύρος 32 bits. Έτσι, σε έναν τυπικό επεξεργαστή πρώτης γενιάς, όπως ο 486, μπορούσαμε να εγκαταστήσουμε 4 X 1 MB, 4 X 2 MB, ή 4 X 4 MB σε κάθε bank. Εάν διαθέταμε ένα μόνο bank (διαθέσιμος χώρος για 4 modules), ήταν δαπανηρό να αυξήσουμε τη RAM, γιατί έπρεπε να απομακρύνουμε τα παλιά modules. Το πρότυπο των 30-pin SIMMs χρησιμοποιήθηκε και για συστήματα με επεξεργαστές τρίτης και τέταρτης γενιάς. Σε συστήματα με επεξεργαστές τέταρτης, πέμπτης και έκτης γενιάς χρησιμοποιείται το πρότυπο των 72-pin SIMMs.



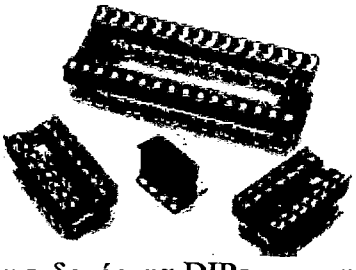
Ο τρόπος σύνδεσης των DIPs πάνω σε ένα memory module τύπου SIMM

συνήθως 3 υποδοχές για DIMM modules.Το μεγάλο πλεονέκτημα της SDRAM είναι η αυξημένη ταχύτητα που επιτρέπει και την αύξηση της ταχύτητας του διαύλου μνήμης. Με EDO RAM σε ταχύτητα 60 ns μπορούμε να επιτύχουμε μέγιστη ταχύτητα στο δίαυλο μνήμης 75 MHz, ενώ με SDRAM η ταχύτητα του διαύλου μνήμης μπορεί να φτάσει και τα 100 MHz. Επιπλέον, η SDRAM εργάζεται σε συγχρονισμό με το δίαυλο μνήμης για καλύτερη απόδοση του συστήματος. Όλα τα νέα chipsets μπορούν να χειριστούν SDRAM. Μερικές μητρικές πλακέτες διαθέτουν εξίσου υποδοχές για DIMM και SIMM modules, έτσι ώστε να υπάρχει η δυνατότητα επιλογής ανάμεσα σε EDO (υποδοχές για SIMM) και σε SDRAM (υποδοχές για DIMM).

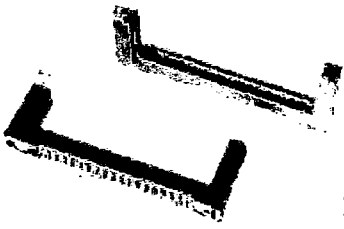
Υπάρχουν δυο διαφορετικοί τύποι DIMMs ανάλογα με τη τάση λειτουργίας τους. Έτσι, έχουμε τα DIMMs των 3.3 Volts και 5.0 Volts. Επίσης, υπάρχουν εκδόσεις buffered και unbuffered DIMMs. Το στάνταρ είναι η έκδοση των 3.3 Volts unbuffered DIMM. Τέλος, υπάρχει και μια ακόμα έκδοση DIMM, η οποία χρησιμοποιείται στους φορητούς υπολογιστές (laptop) και ονομάζεται SODIMM (small outline DIMM).

4.8.5. Συνδετήρες και υποδοχές

Οι συνδετήρες (connectors) των memory modules, καθώς και οι υποδοχές (sockets) της μητρικής πλακέτας είναι διαθέσιμα σε δυο χρώματα: σε χρυσό και ασημένιο. Οι περισσότερες παλιές μητρικές πλακέτες χρησιμοιούσαν για τη σύνδεση των SIMMs, υποδοχές με χρυσό χρώμα, ενώ στις νέες μητρικές πλακέτες χρησιμοποιούνται υποδοχές ασημένιου χρώματος. Είναι σημαντικό να κατανοήσουμε ότι πρέπει να τοποθετούμε τα memory modules σε υποδοχές του ίδιου χρώματος (δηλαδή "χρυσά" memory modules σε "χρυσές" υποδοχές και "ασημένια" memory modules σε "ασημένιες" υποδοχές). Αν τοποθετήσουμε ένα memory module σε διαφορετικού χρώματος υποδοχή, τότε υπάρχει η περίπτωση μετά από κάποιο χρονικό διάστημα να συμβεί χημική αντίδραση μεταξύ των επιφανειών διαφορετικού χρώματος και να παραχθούν οξείδια που επικάθονται στα σημεία επαφής. Αυτά τα οξείδια καθιστούν τη σύνδεση μη αξιόπιστη και μπορούν να προκαλέσουν μια σειρά δυσλειτουργιών (βραχυκύκλωμα) μέσα στο σύστημα. Προκειμένου να αποφύγουμε δυσάρεστες καταστάσεις, πρέπει να προσέχουμε όταν αγοράζουμε μνήμη, να συμπίπτει το χρώμα των memory modules με το χρώμα των υποδοχών της μητρικής πλακέτας. Το στάνταρ που χρησιμοποιείται στις σύγχρονες μνήμες με SIMMs, είναι τα ασημένια memory modules. Η χρησιμοποίηση χρυσού χρώματος αντί για ασημένιο μεώνει το κόστος και αν τηρηθούν μερικές προφυλάξεις έχει την ίδια αξιοπιστία.



υποδοχές για DIPs



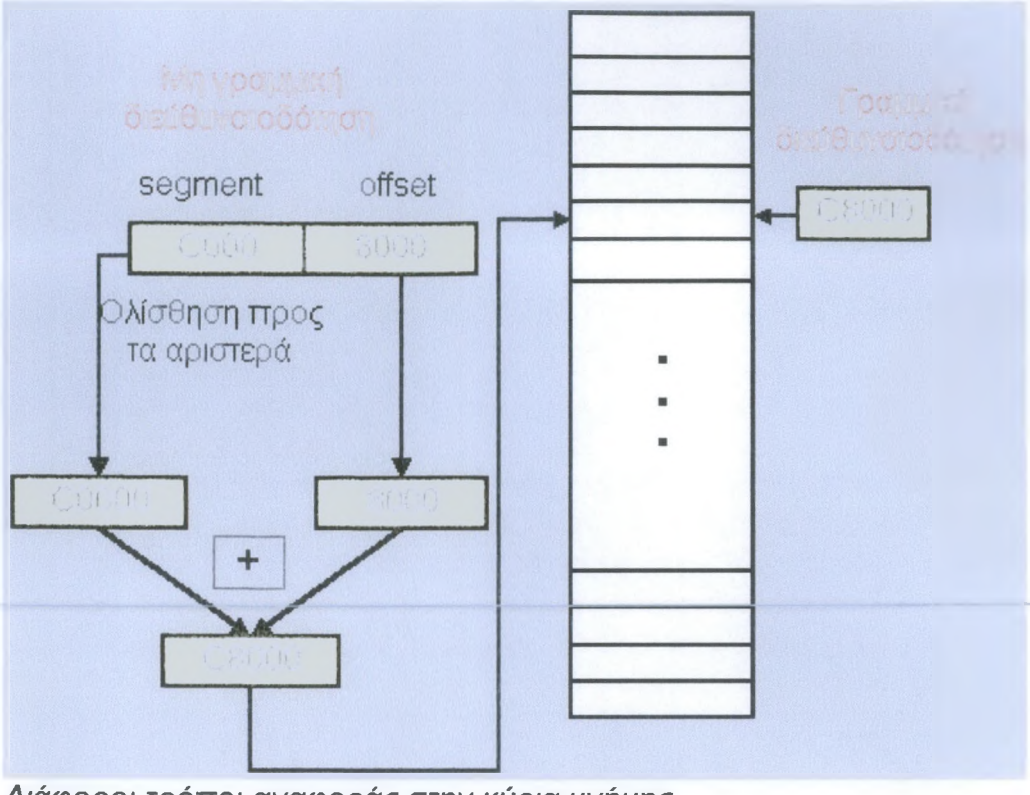
υποδοχές για DIMMs

υποδοχές για SIMMs

4.8.6. Λογική οργάνωση μνήμης

Για να κατανοήσουμε πως είναι λογικά οργανωμένη η μνήμη, θα πρέπει πρώτα να καταλάβουμε πως ο υπολογιστής χειρίζεται τις διευθύνσεις της μνήμης. Ο απλούστερος και ευκολότερος τρόπος για να αναφερθούμε στις διευθύνσεις της μνήμης είναι να χρησιμοποιήσουμε μια γραμμική σειρά διευθύνσεων, δηλαδή η κάθε διεύθυνση να διαφέρει από την προηγούμενη και την επόμενη κατά ένα ψηφίο (δεκαεξαδικό ή δυαδικό). Για παράδειγμα, λέμε ότι το BIOS του σκληρού δίσκου IDE αρχίζει από τη διεύθυνση C8000 της μνήμης (η διεύθυνση C8000 είναι εκφρασμένη βάσει του δεκαεξαδικού συστήματος). Ωστόσο, οι επεξεργαστές χρησιμοποιούν διαφορετικό τρόπο για την αναφορά στις διευθύνσεις της μνήμης.

Στα συστήματα με επεξεργαστές x86 οι διευθύνσεις της μνήμης χωρίζονται σε δυο μέρη: στο **segment address** και στο **offset**. Αυτά τα δυο μέρη δημιουργούν την πραγματική διεύθυνση ως εξής: αρχικά κάνουμε μια ολίσθηση στο segment address κατά ένα ψηφίο προς τα αριστερά και στη συνέχεια προσθέτουμε το offset (μετατόπιση). Στην πράξη, για να αναφερθούμε σε μια διεύθυνση χρησιμοποιούμε το συμβολισμό **segment:offset**. Για παράδειγμα, έστω ότι θέλουμε να αναφερθούμε στη γραμμική διεύθυνση C8000. Χρησιμοποιώντας τον παραπάνω συμβολισμό, η γραμμική διεύθυνση C8000, ισοδυναμεί με C000:8000. Έτσι, παίρνουμε το segment address C000, το οποίο και ολισθένουμε κατά ένα ψηφίο προς τα αριστερά, οπότε παίρνουμε τον αριθμό C0000, στον οποίο προσθέτουμε το offset 8000 και παράγεται η γραμμική διεύθυνση C8000. Η γραμμική διεύθυνση μπορεί επίσης να προσπελαστεί και με το συνδυασμό C800:0000.



Διάφοροι τρόποι αναφοράς στην κύρια μνήμης

4.8.7. Λογικά μέρη της μνήμης

Η κύρια μνήμη του συστήματος αν και αντιμετωπίζεται τις περισσότερες φορές ως μια οντότητα, στην πράξη έχει χωριστεί σε 4 διαφορετικά λογικά μέρη, καθένα από τα οποία έχει και ένα συγκεκριμένο ρόλο:

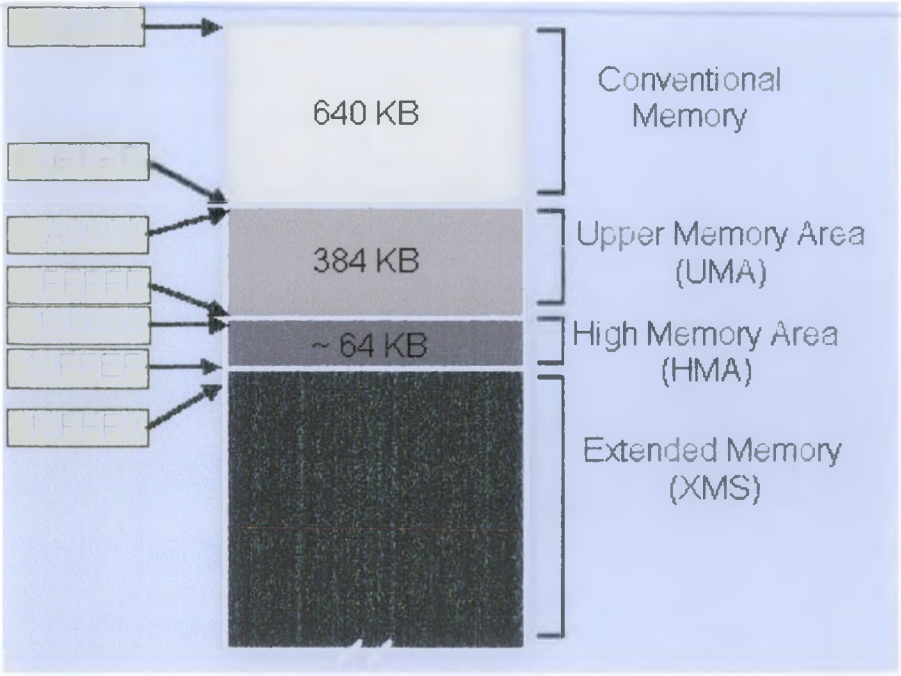
- **Conventional Memory:** Τα πρώτα 640 KB της κύριας μνήμης του συστήματος είναι γνωστά ως **συμβατική μνήμη (conventional memory)**. Είναι ο χώρος της μνήμης που χρησιμοποιείται από τα προγράμματα του DOS, καθώς και από αρκετούς οδηγούς. Η συμβατική μνήμη αρχίζει από τη γραμμική διεύθυνση 00000 και τελειώνει στη διεύθυνση 9FFFF.

- **Upper Memory Area (UMA):** Είναι τα τελευταία 384 KB του πρώτου megabyte της κύρια μνήμης του συστήματος. Τα 384 KB της UMA που ακολουθούν τα 640 KB της Conventional Memory αποτελούν το πρώτο megabyte της κύρια μνήμης. Ο χώρος της UMA χρησιμοποιείται από διάφορες συσκευές του συστήματος, καθώς και για ορισμένες ειδικές εργασίες όπως για το shadowing στις ROMs των οδηγών. Η Upper Memory Area αρχίζει από τη γραμμική διεύθυνση A0000 και τελειώνει στη διεύθυνση FFFFF.

- **High Memory Area (HMA):** Είναι τα πρώτα 64 KB (μείον 16 bytes) του δεύτερου megabyte της κύρια μνήμης του συστήματος. Από τεχνικής απόψεως, πρόκειται για τα πρώτα 64 KB της εκτεταμένης μνήμης

(extended memory), αλλά μπορεί να προσπελαστεί ως ξεχωριστή οντότητα, μόνο όταν ο επεξεργαστής λειτουργεί σε real mode, οπότε και δε συμπεριλαμβάνεται μέσα στην υπόλοιπη εκτεταμένη μνήμη. Χρησιμοποιείται όταν τα προγράμματα του DOS χρειάζονται επιπλέον conventional memory, ενώ ο χώρος αυτός ορίζεται από τις διευθύνσεις 100000 και 10FFEF.

- **Extended Memory (XMS):** Είναι ο χώρος πάνω από τη HMA και μέχρι το τέλος της κύριας μνήμης του συστήματος. Χρησιμοποιείται για προγράμματα και τα δεδομένα όταν το λειτουργικό σύστημα τρέχει σε protected mode, όπως για παράδειγμα όλες οι εκδόσεις των Windows. Η εκτεταμένη μνήμη αρχίζει από τη διεύθυνση 10FFF0 και εκτείνεται μέχρι το τέλος της κύριας μνήμης. Από τεχνικής απόψεως, η HMA είναι μέρος της extended memory όταν ο επεξεργαστής δε λειτουργεί σε real mode.



Λογική οργάνωση της μνήμης

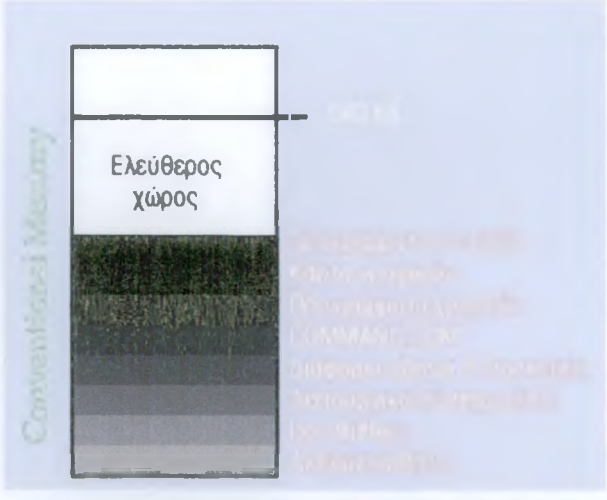
4.8.8. Συμβατική μνήμη

Τα πρώτα 640 KB της κύριας μνήμης του συστήματος ονομάζονται **συμβατική μνήμη** (**conventional memory**). Η ονομασία έχει σχέση με το γεγονός ότι το λειτουργικό σύστημα DOS καθώς και τα προγράμματα που τρέχουν κάτω από το DOS χρησιμοποιούν συνήθως αυτό το χώρο της κύριας μνήμης. Αρχικά, αυτός ήταν και ο μοναδικός χώρος τον οποίο μπορούσαν να χρησιμοποιήσουν τα διάφορα προγράμματα. Σήμερα, αν και οι υπολογιστές διαθέτουν πολύ περισσότερη μνήμη από τους αρχικούς υπολογιστές, εντούτοις σε ορισμένες περιπτώσεις αυτά τα 640 KB της conventional memory εξακολουθούν να είναι ο σημαντικότερος χώρος της κύρια μνήμης. Αυτό ωφείλεται στο γεγονός ότι για να χρησιμοποιηθούν τα υπόλοιπα τμήματα της κύριας μνήμης από τα προγράμματα του DOS, θα πρέπει να

έχουμε ειδικό software. Η conventional memory εμπεριέχεται στο διάστημα 00000-9FFFF.

Για ποιον όμως λόγο υπάρχει αυτό το όριο των 640 KB στο μέγεθος της conventional memory; Ο λόγος για τον οποίο υπάρχει αυτό το όριο στο μέγεθος της conventional memory, έχει σχέση με την απόφαση της IBM να τοποθετήσει το χώρο που θα χρησιμοποιούνταν από το σύστημα κάτω από το χώρο που προορίζονταν για τα προγράμματα των χρηστών. Επίσης, και το γεγονός ότι οι πρώτοι επεξεργαστές μπορούσαν να διευθυνσιοδοτήσουν μέχρι 1 MB κύριας μνήμης συνέβαλε στο όριο των 640 KB της conventional memory. Οι παραπάνω λόγοι οδήγησαν στο διαχωρισμό της conventional memory από την υπόλοιπη κύρια μνήμη (extended memory). Αν ο χώρος που προορίζονταν για τα προγράμματα των χρηστών (conventional memory) είχε τοποθετηθεί κάτω από το χώρο που χρησιμοποιούνταν αποκλειστικά από το σύστημα, θα υπήρχε δυνατότητα να επεκτείνουμε την conventional memory.

Κατά τη διάρκεια της δεκαετίας του '80, καθώς το μέγεθος και η πολυπλοκότητα των προγραμμάτων αυξανόταν με ταχείς ρυθμούς, το μέγεθος της conventional memory φάνταζε πολύ μικρό, αν λάβουμε υπόψη και το γεγονός ότι δεν ήταν διαθέσιμα στα προγράμματα των χρηστών όλα τα 640 KB της conventional memory. Μεγάλα τμήματα της conventional memory χρησιμοποιούνταν από ρουτίνες του DOS, από vector tables, από buffers για file access, από διάφορους οδηγούς (όπως CD-ROM drives, κτλπ). Τα προγράμματα των χρηστών από τα 640 KB της conventional memory είχαν διαθέσιμα μόνο 500-550 KB. Υπήρχαν συστήματα, όπου χωρίς να υπάρχει κάποια διαδιακασία βελτιστοποίσης της conventional memory μετά την εκκίνηση είχαν διαθέσιμα λιγότερα από 400 KB conventional memory. Επίσης, μερικά προγράμματα χρηστών είχαν αναπτυχθεί έτσι ώστε να μην τρέχουν αν η διαθέσιμη conventional memory ήταν λιγότερη από 500 KB. Επομένως, ο περιορισμός στο μέγεθος της conventional memory ήταν η αιτία για την εμφάνιση πολλών προβλημάτων στη λειτουργία του υπολογιστή.



Ο χώρος της Conventional Memory

Η Microsoft χρησιμοποίησε δυο τεχνικές, στην προσπάθειά της να λύσει μερικά από τα προβλήματα που σχετιζόταν με τον περιορισμό στο μέγεθος

της conventional memory. Η πρώτη τεχνική λειτουργούσε με τον εξής τρόπο: το λειτουργικό σύστημα DOS δεν φορτώνονταν στην conventional memory αλλά στην high memory area (τα πρώτα 64 KB της extended memory). Με τον τρόπο αυτό, εξοικονομούνταν για τα προγράμματα των χρηστών 45 KB conventional memory. Όταν χρησιμοποιούνταν η δεύτερη τεχνική, πολλοί οδηγοί (όχι προγράμματα χρηστών) αντί να χρησιμοποιήσουν conventional memory, χρησιμοποιούσαν την upper memory area (τα 384 KB μεταξύ της conventional memory και extended memory).

Η μεγιστοποίηση της διαθέσιμης στα προγράμματα των χρηστών conventional memory, έγινε βασικός στόχος σε κάθε υπολογιστή που χρησιμοποιούσε προγράμματα του DOS. Βελτιστοποιώντας την λειτουργία των high memory area και upper memory area, σε συνδιασμό με τη χρησιμοποίηση διαφόρων τρικ, μπορούμε να εξασφαλίσουμε για τα προγράμματα των χρηστών μέχρι και 620 KB conventional memory.

Στα σύγχρονα λειτουργικά συστήματα (σε όλα πλην του DOS) που λειτουργούν σε protected mode, η conventional memory δεν έχει τόση σημασία όση είχε στα πρώτα συστήματα που χρησιμοποιήθηκε. Τα προγράμματα που τρέχουν κάτω από αυτά τα λειτουργικά συστήματα χρησιμοποιούν την extended memory. Επίσης, τα 32-bit λειτουργικά συστήματα (όπως τα Windows 95) χρησιμοποιούν επίσης την extended memory για τους hardware drivers. Βέβαια, μερικά κομμάτια κώδικα εξακολουθούν να φορτώνονται στην conventional memory, αλλά οι εφαρμογές των χρηστών δεν περιορίζονται από το όριο των 640 KB της conventional memory.

4.9. Ανίχνευση και διόρθωση λαθών

Η μνήμη είναι μια αποθηκευτική συσκεύη ηλεκτρονικού τύπου και όπως συμβαίνει σε όλες τις υπόλοιπες αποθηκευτικές συσκεύες, υπάρχει περίπτωση να επιστραφούν διαφορετικές πληροφορίες από αυτές που είχαν αρχικά αποθηκευτεί. Κάποιες μνήμες είναι πιο εύκολο να παρουσιάσουν λάθη σε σχέση με κάποιες άλλες, εξαιτίας κυρίως της διαφορετικής τεχνολογίας που χρησιμοποιούν. Για παράδειγμα, στις μνήμες DRAMs, τα "0" και "1" αποθηκεύονται με τη μορφή ηλεκτρικών φορτίων σε μικροπυκνωτές. Αυτό έχει ως αποτέλεσμα την ανάγκη για συνεχή αναζωογώνηση των ηλεκτρικών φορτίων, οπότε είναι πιο επιρεπής στην εμφάνιση λαθών σε σχέση με τις μνήμες στατικής αποθήκευσης (SRAMs).

Κάθε bit της μνήμης μπορεί να είναι "1" ή "0" και αντιστοιχεί σε ένα συγκεκριμένο επίπεδο τάσης. Έτσι, σε ένα 5V σύστημα το bit "1" αντιστοιχεί στην τάση +5V, ενώ το bit "0" στην τάση 0V. Αν ο αισθητήρας που διαβάζει την τάση αναγνώσει την τιμή +4.2V, τότε υποθέτει ότι το αντίστοιχο bit θα έχει την τιμή "1", αφού το +4.2V είναι πολύ πιο κοντά στο +5V από ότι στο 0V. Αν όμως μια τάση +5V διαβαστεί από τον αισθητήρα ως +1.9V, τότε αντί για την πραγματική τιμή "1", ο αισθητήρας θα μας επιστρέψει την τιμή "0", οπότε λέμε ότι έχουμε κάποιο λάθος στη μνήμη.

Υπάρχουν δυο τύποι λαθών που εμφανίζονται στη μνήμη ενός συστήματος. Ο πρώτος τύπος λαθών ονομάζεται **επαναλαμβανόμενα λάθη (repeatable errors ή hard errors)** και οφείλονται σε δυσλειτουργίες του hardware. Για παράδειγμα, μπορεί να καταστραφεί κάποιο κομμάτι του κυκλώματος (hard error) και να επιστρέφεται μια μόνο τιμή ("0" ή "1"), ανεξάρτητα με την τιμή που αποθηκεύουμε. Τα λάθη αυτά είναι πολύ εύκολο να εντιπιστούν αφού είναι επαναλαμβανόμενα (repeatable error).

Ο δεύτερος τύπος λαθών ονομάζεται **παροδικά λάθη (transient errors ή soft errors)**. Αυτά τα λάθη εμφανίζονται όταν ένα bit διαβαστεί λάθος μια φορά, αλλά στις επόμενες προσπελάσεις διαβαστεί σωστά. Αυτού του τύπου τα λάθη αν και είναι τα πιο κοινά, είναι πολύ δύσκολο να ανιχνεύτουν. Ο χρόνος εμφάνισης αυτών των λαθών είναι εντελώς τυχαίος και έχουν το χαρακτηριστικό ότι διορθώνονται μόνα τους. Τα soft errors μπορεί να οφείλονται σε προβλήματα του hardware της μνήμης ή στην κακή ποιότητα της μητρικής πλακέτας ή εξαιτίας του κακού χρονισμού του συστήματος ή τέλος λόγω της ραδιοακτινοβολίας που δημιουργείται από τα διάφορα μέρη ενός υπολογιστή.

Το πόσο καταστροφικό μπορεί να είναι ένα λάθος, εξαρτάται από από τη συχνότητα εμφάνισης και το είδος της ζημιάς που μπορεί να προκαλέσει. Για παράδειγμα, αν σε ένα παιχνίδι, αντιστραφεί η τιμή ενός bit που καθορίζει το χρωματισμό ενός pixel στην οθόνη, το αποτέλεσμα δε θα γίνει αντιληπτό από το χρήστη. Αν όμως κατά τη διάρκεια του defragmatation στο σκληρό δίσκο παρουσιαστεί κάποιο λάθος στη μεταφορά πληροφοριών από τη μνήμη στο FAT, τότε θα έχουμε σοβαρό πρόβλημα. Όταν ένα λάθος είναι σημαντικό, ανεξάρτητα από τη συχνότητα εμφάνισης, πρέπει να βρεθεί τρόπος ανίχνευσης και διόρθωσής του. Από την άλλη μεριά, αν ένα λάθος δεν είναι και τόσο σημαντικό, η συχνότητα εμφάνισης δε μας αποσχολεί και δε χρειάζεται να λάβουμε μέτρα αντιμετώπισής του. Πάντως, ανεξάρτητα από το αν ένα λάθος είναι σημαντικό ή όχι, θα πρέπει να χρησιμοποιήσουμε διάφορα πρωτόκολλα ανίχνευσης και διόρθωσης, προκειμένου να ελαχιστοποιήσουμε το κόστος των απωλειών που προκαλούνται από τα λάθη της μνήμης.

4.9.1. ECC μνήμες

Χρήση ή όχι ισοτιμίας και ECC μνήμες

Τα memory modules είναι διαθέσιμα σε δυο τύπους ανάλογα με το αν χρησιμοποιούν ή όχι επιπλέον πληροφορίες για την ανίχνευση και διόρθωση λαθών. Έτσι, έχουμε modules που κάνουν χρήση των **bits ισοτιμίας (parity modules)** και modules που δεν κάνουν χρήση των **bits ισοτιμίας (non-parity modules)**. Στα non-parity modules για κάθε bit αποθηκεύουμε μόνο ένα bit. Για παράδειγμα, για μια ομάδα 8 bits (1 byte) θα αποθηκεύσουμε μόνο τα 8 bits. Αντίθετα, στα parity modules για κάθε ομάδα bits προκαθορισμένου μεγέθος αποθηκεύουμε και μερικά επιλέον bits που μπορούν να χρησιμοποιηθούν για την ανίχνευση και διόρθωση λαθών. Για παράδειγμα, για κάθε μια ομάδα πραγματικών δεδομένων των 8 bits (1 byte) θα αποθηκεύσουμε και ένα επιπλέον bit (συνολικά 9 bits). Ο παρακάτω πίνακας δείχνει το εύρος των bits για SIMM modules με χρήση ή όχι των bits ισοτιμίας:

Τύπος Module	Εύρος σε bits Non-Parity SIMM	Εύρος σε bits Parity SIMM
30-Pin SIMM	8 bits	9 bits
72-Pin SIMM	32 bits	36 bits
168-Pin DIMM	64 bits	72 bits

Τα parity bits χρησιμοποιούνται στον έλεγχο ισοτιμίας **(parity checking)**, που είναι η πιο κοινή μέθοδος ανίχνευσης λαθών. Επίσης, τα parity bits χρησιμοποιούνται και σε πιο προχωρημένες τεχνικές ανίχνευσης και διόρθωσης λαθών, όπως η μέθοδος ECC. Οι non-parity μνήμες δε χρησιμοποιούν καμία τεχνική για την ανίχνευση και διόρθωση λαθών.

Η μεγαλύτερη διαφορά μεταξύ των δυο μεθόδων ανίχνευσης λαθών (parity checking και ECC), είναι ότι η μέθοδος ECC αν και χρησιμοποιεί κάποιου είδους έλεγχο ισοτιμίας, εντούτοις ο τρόπος λειτουργίας της δεν είναι ο ίδιος με αυτόν του συνηθισμένου parity checking. Ο λόγος είναι ότι το επιπλέον bit στα ECC modules δεν μπορεί να προσπελαστεί μόνο του, σε αντίθεση με το αντίστοιχο bit ισοτιμίας στο parity checking. Ανακεφαλαιώνοντας, παραθέτουμε τον παρακάτω πίνακα:

Τύπος Module	Λειτουργία Parity	Λειτουργία ECC
True Parity	Ναι	Ναι
ECC	Όχι	Ναι

4.9.2. Έλεγχος ισοτιμίας

Ο **έλεγχος ισοτιμίας** (**Parity Checking**) είναι μια μέθοδος ανίχνευσης λαθών ενός μόνο bit. Όλοι οι υπολογιστές, από τους αρχικούς IBM PC (1981) μέχρι και αυτούς που κατασκεύαστηκαν ως στις αρχές του 1990, χρησιμοποίησαν τον έλεγχο ισοτιμίας για την ανίχνευση λαθών στη μνήμη. Ο έλεγχος ισοτιμίας απαιτεί τη χρησιμοποίηση μνήμης, τα modules της οποίας παρέχουν ένα επιπλέον bit για κάθε 8 bits πραγματικών δεδομένων που αποθηκεύονται. Η τιμή του επιπλέον bit χρησιμοποιείται για την ανίχνευση μονών λαθών μέσα στην ομάδα των 8 bits. Στα σύγχρονα συστήματα, για να πραγματοποιείται ο

έλεγχος ισοτιμίας, θα πρέπει να είναι ενεργοποιημένη η αντίστοιχη παράμετρος του BIOS.

Κάθε byte πραγματικών δεδομένων αποθηκεύται στην κύρια μνήμη ως μια ομάδα 8 bits, καθένα από τα οποία μπορεί να έχει τιμή "0" ή "1". Όταν είναι ενεργοποιημένος ο έλεγχος ισοτιμίας, κάθε φορά που ένα byte πρόκειται να γραφεί στη μνήμη, ένα λογικό κύκλωμα που ονομάζεται **parity generator/checker** εξετάζει το byte. Το parity generator/checker αφού εξετάσει το byte, προσδιορίζει αν ο αριθμός των άσσων "1" μέσα στο byte είναι περιττός ή άρτιος αριθμός. Αν ο αριθμος των άσσων είναι άρτιος και χρησιμοποιούμε περιττή ισοτιμία θέτουμε στο ένατο bit την τιμή "1", αλλιώς την τιμή "0". Επομένως, στην περιττή ισοτιμία η τιμή του ένατου bit θα είναι τέτοια, ώστε το συνολικό άθροισμα των άσσων στα 9 bits να βγαίνει περιττός αριθμός. Αντίθετα, όταν χρησιμοποιούμε άρτια ισοτιμία, η τιμή που παράγει το κύκλωμα parity generator/checker για το ένατο bit, θα είναι τέτοια ώστε ο συνολικός αριθμός άσσων (και για τα 9 bits), να είναι άρτιος αριθμός. Συνήθως, στους υπολογιστές χρησιμοποιείται η περιττή ισοτιμία για το Parity Checking. Ο παρακάτω πίνακας δείχνει πως υπολογίζεται η τιμή του bit ισοτιμίας (για περιττή ισοτιμία):

Δεδομένα	Πλήθος άσσων στο δεδομένο	Parity Bit	Συνολικός αριθμός άσσων στο σύνολο μετά την προσθήκη του Parity Bit
00000000	0	1	1
10110011	5	0	5
00100100	2	1	3
11111111	8	1	9

Όπως φαίνεται και από τον παραπάνω πίνακα, ο αριθμός των άσσων στην ακολουθία των 9 bits είναι πάντα περιττός αριθμός (για περιττή ισοτιμία). Όταν πρόκειται να διαβαστεί ένα byte από τη μνήμη, το κύκλωμα που παράγει το bit ισοτιμίας λειτουργεί σαν ελεγκτής της ισοτιμίας. Συγκεκριμένα, αφού διαβαστούν και τα 9 bits, καθορίζεται ο αριθμός των άσσων που περιέχονται στην ακολουθία των 9 bits. Αν ο αυτός ο αριθμός είναι περιττός, τότε δεν έχει συμβεί (στην πραγματικότητα μπορεί και να έχουν συμβεί πολλαπλά λάθη) κανένα λάθος στη μνήμη και το byte που διαβάστηκε μεταφέρεται σε αυτόν που το ζήτησε. Στην περίπτωση όμως, που το πλήθος των άσσων είναι άρτιος αριθμός, αυτό σημαίνει ότι έχει συμβεί λάθος σε ένα τουλάχιστον bit, χωρίς όμως να μπορεί να προσδιοριστεί η ταυτότητα αυτού του bit. Όταν ανιχνευτεί ένα λάθος μετά από έναν έλεγχο ισοτιμίας, τότε το κύκλωμα ισοτιμίας παράγει ένα σήμα διακοπής NMI **(non-maskable interrupt)** προς τον επεξεργαστή και διακόπτεται η διαδιακασία ανάγνωσης του συγκεκριμένου byte.

Τι συμβαίνει στην περίπτωση που μεταβληθεί η τιμή σε δυο bits; Ας υποθέσουμε ότι έχουμε το byte δεδομένων "00100100", στο οποίο έχει προστεθεί το bit ισοτιμίας και έχει αποθηκευτεί στην κύρια μνήμη με την μορφή "00100100 1". Αν μεταβληθεί η τιμή ενός bit από "1" σε "0" και ενός άλλου bit από "0" σε "1", τότε το πλήθος των άσσων στην έκφραση των 9 bits εξακολουθεί να είναι περιττός αριθμός αν και έχουν συμβεί δυο λάθη. Δυστυχώς, η μέθοδος του ελέγχου της ισοτιμίας μπορεί να ανιχνεύσει μόνο ένα λάθος που έχει συμβεί σε 1 bit.

Πολλοί άνθρωποι πιστεύουν ότι η δημιουργία / έλεγχος του bit ισοτιμίας κατά την αποθήκευση / ανάγνωση αντίστοιχα των bytes προκαλεί κάποια καθυστέρηση στη μνήμη. Αυτό δεν είναι σωστό, αφού η διαδικασία της δημιουργίας / ελέγχου γίνεται παράλληλα με τη διαδικασία αποθήκευσης / ανάγνωσης, οπότε δεν επιβαρυνόμαστε με επιπλέον χρόνο και έτσι δε μειώνεται η απόδοση της μνήμης.

Ενότητα Γ: Προγραμματισμός Η/Υ και Λειτουργικά συστήματα (Windows XP)

Κεφάλαιο 5: Προγραμματισμός Η/Υ, γλώσσες και είδη προγραμματισμού

5.1. Προγραμματισμός (Ορισμός)

Προγραμματισμός είναι η διαδικασία ανάπτυξης προγράμματος, δηλαδή συνόλου εντολών κατανοητών από τον υπολογιστή που υλοποιούν ένα αλγόριθμο.

Από την αρχή της εμφάνισης των υπολογιστών γίνονται προσπάθειες μεθοδολογιών και τεχνικών προγραμματισμού, που θα εξασφάλιζαν τη δημιουργία απλών και κομψών προγραμμάτων.

Βασικά στοιχεία του προγράμματος είναι τα δεδομένα του καθώς και οι δομές δεδομένων πάνω στα οποία και ενεργεί.

5.1.1. Γλώσσα Προγραμματισμού (τι είναι;)

Οι γλώσσες προγραμματισμού έκαναν την εμφάνιση τους κατά την διάρκεια του $2^{ου}$ παγκόσμιου πολέμου σε αριθμητικά προβλήματα του πυροβολικού (υπολογισμό καμπυλών βολών).

Οι εντολές ήταν ακολουθίες από 0 και 1 (01-patterns). Στην συνέχεια έκαναν την εμφάνιση τους οι assemblers που χρησιμοποιούσαν mnemonics και απευθύνονταν άμεσα στην CPU.

Οι γλώσσες Προγραμματισμού αναπτύχθηκαν με σκοπό την επικοινωνία ανθρώπου και μηχανής. Οι πρώτοι υπολογιστές ήσαν τεράστιοι σε μέγεθος με

περιορισμένη ταχύτητα, σε σχέση με τη σημερινή κατάσταση, που συμβαίνει ακριβώς το αντίθετο. Στον ENIAC (τον 1ο υπολογιστή) γι α να γίνουν λίγοι υπολογισμοί, έπρεπε να αλλάξουν θέση εκατοντάδες διακόπτες και να γίνουν αρκετές ρυθμίσεις και να ελεγχθούν καλωδιώσεις. Την επόμενη φορά, που το σύστημα έπρεπε να εκτελέσει μια άλλη διαδικασία, ο υπολογιστής έπρεπε να αναδιαρθρωθεί.

Στην συνέχεια παρουσιάστηκαν οι ανωτέρου επιπέδου γλώσσες όπως η Basic και η Fortran και μεταγενέστερα η γλώσσα C (που εξέλιξη της αποτελεί η C++ καθώς και οι Visual εκδόσεις της). Οι ανωτέρου επιπέδου γλώσσες έχουν σύνταξη και λεξιλόγιο (keywords) που προσεγγίζουν την ανθρώπινη γλώσσα. Μετά ο πηγαίος κώδικας μεταφράζεται σε γλώσσα μηχανής από interpreters είτε compilers.

• Ένας interpreter μεταφράζει σε γλώσσα μηχανής την εντολή άμεσα την στιγμή που εισάγεται.

• Ο compiler μεταφράζει όλο τον κώδικα πρώτα σε μια ενδιάμεση κατάσταση δίνοντας σαν αποτέλεσμα το object αρχείο (αρχεία με κατάληξη *.obj). Στην συνέχεια είναι η σειρά του linker που μετατρέπει το object αρχείο σε εκτελέσιμο (αρχεία με κατάληξη *.exe). Παράγοντας έτσι τον κώδικα μηχανής.

Αρχικά ο προγραμματιστής έπρεπε να γραφεί μικρά προγράμματα λόγω:

• έλλειψης μνήμης (RAM),
• πανάκριβης υπολογιστικής ισχύς.

Εκείνες τις μέρες χρησιμοποιούνταν η πιο κατώτερη γλώσσα προγραμματισμού η γλώσσα μηχανής (machine language).

Ο υπολογιστής χρησιμοποιεί αρκετά εξειδικευμένη γλώσσα με bits (δυαδικά ψηφία) και bytes (ψηφιολέξεις) που μόνο οι επαγγελματίες προγραμματιστές μπορούν να καταλάβουν. Για το πολλούς ανθρώπους θα πρέπει με κάποιο τρόπο να μεταφράζεται το συγκεκριμένο πρόβλημα σε γλώσσα που μπορεί να καταλάβει ο υπολογιστής. Αυτό τον σκοπό έχουν οι γλώσσες προγραμματισμού, από την στιγμή που το πρόβλημα έχει μεταφρασθεί σε κάποια γλώσσα που μπορεί να καταλάβει η μηχανή, ο υπολογιστής ακολουθεί τις οδηγίες για να επιλύσει το πρόβλημα. Το πρόγραμμα λοιπόν είναι μία σειρά από οδηγίες που ο υπολογιστής μπορεί να χρησιμοποιήσει για την επίλυση κάποιου προβλήματος.

5.1.2. Ιεραρχικός Προγραμματισμός

Η σχεδίαση προγράμματος πρέπει να προχωρεί από πάνω προς τα κάτω. Ξεκινάμε με μια πολύ απλή και σύντομη δήλωση για το τι κάνει το πρόγραμμα, π.χ. με τον υπολογισμό του μέσου όρου βαθμολογίας. Αυτό γίνεται στο αρχικό επίπεδο σχεδίασης του προγράμματος, ενώ στο επόμενο επίπεδο εμφανίζονται με περισσότερες λεπτομέρειες οι επεξεργασίες που εκτελεί το πρόγραμμα. Καθεμιά απ' αυτές δεν είναι τελείως συμπληρωμένη,

αλλά περιγράφεται με περισσότερες λεπτομέρειες σε κατώτερο επίπεδο του προγράμματος.

Η διαδικασία αυτή επαναλαμβάνεται βήμα προς βήμα και, σε κάθε κατώτερο επίπεδο, εμφανίζονται περισσότερες λεπτομέρειες. Η σε κατώτερα επίπεδα ανάλυση σταματάει, όταν η επεξεργασία περιέχει τόσες λεπτομέρειες ώστε να μπορεί να κωδικοποιηθεί σε μια γλώσσα προγραμματισμού.

Τα παραπάνω συνιστούν την τεχνική του ιεραρχικού προγραμματισμού όπου, ξεκινώντας από τα γενικά σε ανώτερα επίπεδα , προχωρούμε προσθέτοντας λεπτομέρειες σε κατωτέρα επίπεδα

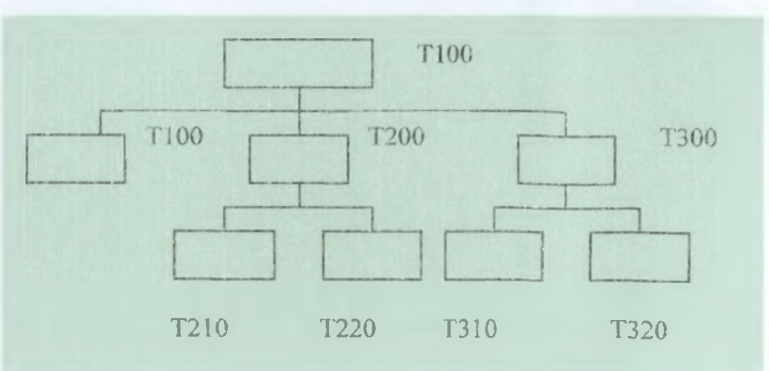
5.1.3. Τμηματικός Προγραμματισμός

Οι διάφορες επεξεργασίες που εμφανίζονται κατά την προηγούμενη ανάπτυξη είναι τμήματα προγράμματος λογικά, πλήρη και ανεξάρτητα μεταξύ τους. Τα τμήματα αυτά έχουν μια **είσοδο** και μια **έξοδο** και αποτελούν λογικά αυτοτελείς προγραμματικές ενότητες, οι οποίες περιέχουν τις βασικές δομές του Δομημένου Προγραμματισμού. Επειδή είναι μικρότερα προγράμματα, είναι δυνατό να κωδικοποιηθούν, να δοκιμαστούν και να διορθωθούν ανεξάρτητα και ο συνδυασμός τους να δώσει το πλήρες πρόγραμμα.

Ο χωρισμός του προγράμματος σε ανεξάρτητες λογικές ενότητες –τμήματα αποτελεί τον **Τμηματικό Προγραμματισμό.** Τα μεμονωμένα τμήματα πρέπει να είναι όσο το δυνατόν μικρότερα, ώστε να διευκολύνεται ο καταμερισμός της λογικής ενότητας του προγράμματος σε μικρότερες, που είναι ευκολότερο να διορθωθούν.

Η τεχνική που χρησιμοποιεί τις αρχές του **Ιεραρχικού** και του **Τμηματικού** προγραμματισμού για την σχεδίαση ενός αλγορίθμου και του αντίστοιχου προγράμματος καλείται **Δομημένος Προγραμματισμός**. Με το συνδυασμό των δυο τεχνικών , κάθε επίπεδο προγράμματος υποστηρίζεται από το αμέσως κατώτερο επίπεδο. Το πρόγραμμα στο κατώτερο επίπεδο επεξεργάζεται τα δεδομένα και συγχρόνως δημιουργεί την πληροφορία η οποία απαιτείται από το υψηλότερο επίπεδο.

Η παράσταση αυτών των τεχνικών γίνεται με τα διαγράμματα **HIPO** (**H**ierarchical **I**nput **O**utput **P**rocessing), που στα Ελληνικά μεταφράζεται ως **Ιεράρχηση Εισόδου – Επεξεργασίας-Εξόδου**.)

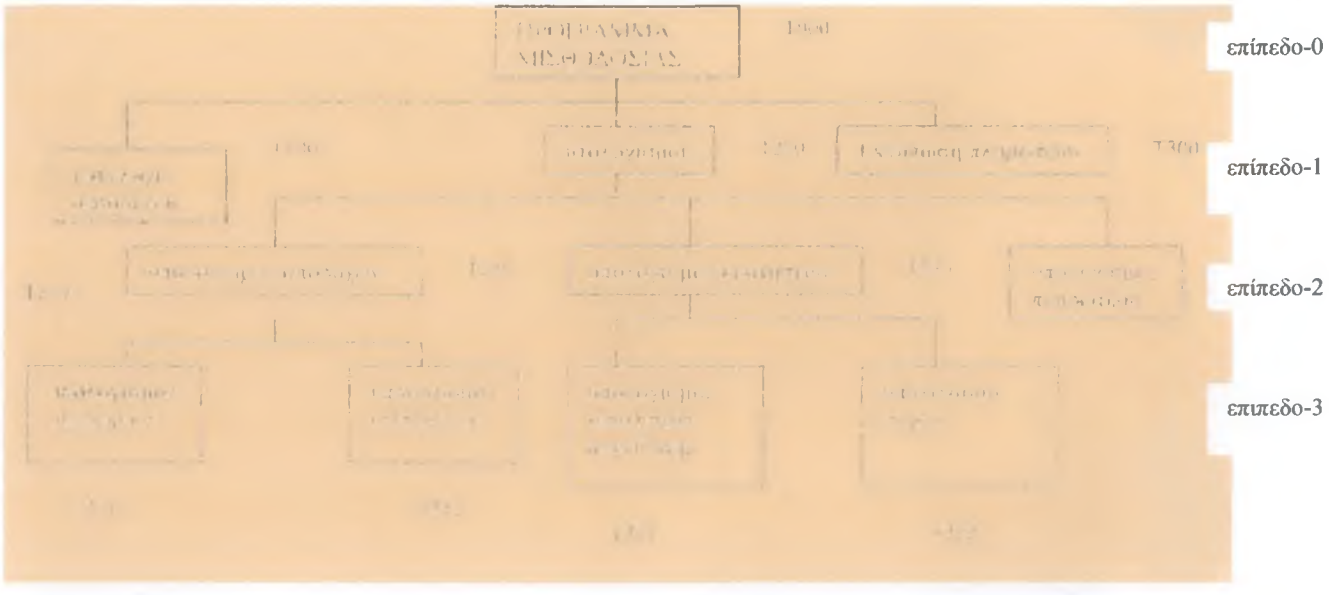


Ένα διάγραμμα HIPO αποτελείται από τρία βασικά τμήματα, τα οποία περιγράφουν την είσοδο δεδομένων , την επεξεργασία και τα αποτελέσματα που απαιτεί ένας αλγόριθμος και το αντίστοιχο πρόγραμμα. Στα διαγράμματα αυτά εμφανίζονται τα διαφορά επίπεδα ιεραρχίας και τα τμήματα του αλγόριθμου –προγράμματος με αριθμό σύμφωνα με τη σειρά εκτέλεσης τους. Στο ανώτερο επίπεδο υπάρχει μια περιγραφή της όλης διαδικασίας που αναπαρίσταται από το διάγραμμα HIPO. Στο αμέσως επόμενο επίπεδο διακρίνονται, σε τμήματα, οι τρεις βασικές λειτουργίες (είσοδος, επεξεργασία και έξοδος) οι οποίες μπορεί να αναλύονται περαιτέρω σε αλλά υποτιμήματα.

Παρακάτω θα αναφέρουμε δυο παραδείγματα σχεδίασης διαγράμματος HIPO.

Το πρώτο παράδειγμα αφορά στον υπολογισμό μισθοδοσίας υπαλλήλων ενώ το δεύτερο στην βαθμολογία μαθητών.

ΥΠΟΛΟΓΙΣΜΟΣ ΜΙΣΘΟΔΟΣΙΑΣ



5.2. Γλώσσες Μηχανής

Γλώσσα μηχανής είναι η γλώσσα προγραμματισμού που αντιλαμβάνεται ο υπολογιστής. Οι εντολές της είναι ακολουθίες δυαδικών ψηφίων 0, 1. Ο τρόπος προγραμματισμού σε γλώσσα μηχανής είναι επίπονος, ακατανόητος για πολλούς και χρονοβόρος.

Αλλά ακόμα και σήμερα οι εντολές των σύγχρονων προγραμμάτων μετατρέπονται από τον σύγχρονο υπολογιστή σε γλώσσα μηχανής, προκειμένου να εκτελεστούν.

Τα προγράμματα που είναι γραμμένα σε γλώσσα μηχανής ενός υπολογιστή είναι εξαρτημένα από η δομή του συγκεκριμένου υπολογιστή και μπορούν να τρέξουν μόνο σε υπολογιστές, που είναι όμοιοι ή έχουν συμβατές ΚΜΕ. Ένα πρόγραμμα γραμμένο σε γλώσσα μηχανής είναι εύκολα κατανοητό από τον υπολογιστή για τον οποίο γράφτηκε αλλά δύσκολα κατανοητό από τον άνθρωπο, ακόμη και από αυτόν που το έγραψε, μετά από κάποιο χρονικό διάστημα.
Ο προγραμματισμός σε γλώσσα μηχανής ήταν ο πρώτος τρόπος προγραμματισμού των υπολογιστών. Ο προγραμματισμός στη γλώσσα αυτή θεωρείται σήμερα ένας άθλος για τον προγραμματιστή της εποχής εκείνης. Οι πρώτοι υπολογιστές προγραμματιζόταν σε γλώσσα μηχανής με τη χρήση διακοπτών, των οποίων η μία θέση αντιπροσώπευε το 0 (μηδέν) και η άλλη το 1 (ένα).
Εάν προσπαθήσετε να προγραμματίσετε με γλώσσα μηχανής, θα συναντήσετε πολλές δυσκολίες.
-- Η γλώσσα και ο κώδικας είναι δύσκολο να διαβαστούν και να μεταφραστούν.
-- Οι εντολές αναφέρονται σε συγκεκριμένες θέσεις μνήμης, που μπορεί να αλλάζουν εάν αλλάξει το πρόγραμμα. Ο προγραμματιστής πρέπει να κανονίσει τις θέσεις μνήμης για την αποθήκευση των δεδομένων.
-- Ο κώδικας ο οποίος εξαρτάται από την μηχανή θα πρέπει να ξαναγραφεί εάν πρόκειται να εκτελεσθεί σε κάποιον διαφορετικό υπολογιστή. Τα περισσότερα προγράμματα περιέχουν πολλές γραμμές από εντολές σε κώδικα μηχανής.

Τα περισσότερα προγράμματα περιέχουν πολλές γραμμές από εντολές σε κώδικα μηχανής.

Επειδή η γλώσσα μηχανής είναι δυσανάγνωστη για τον άνθρωπο αφού οι εντολές της είναι ακολουθίες από δυαδικά ψηφία, είναι χρήσιμος ο ορισμός μιας συμβολικής μορφής των εντολών της ως μνημονικών κωδίκων, αν επιθυμεί κάποιος να γράψει προγράμματα σε αυτή τη γλώσσα. Έτσι ορίστηκαν ευανάγνωστες στον άνθρωπο εκδόσεις γλωσσών μηχανής που ονομάστηκαν συμβολικές γλώσσες (assembly languages). Έτσι μία ακολουθία από bits που στη γλώσσα μηχανής εκφράζει την εντολή που προσθέτει το περιεχόμενο δύο κυκλωμάτων της κεντρικής μονάδας επεξεργασίας, μπορεί σε συμβολική γλώσσα να συμβολίζεται από το A (ADD).

5.2.1. Συμβολικές Γλώσσες

Ο Προγρ/σμος σε γλώσσα μηχανής ήταν και είναι μία πολύ δύσκολη δουλειά για τον προγραμματιστή. Έτσι, πολύ σύντομα οι προγραμματιστές άρχισαν να αντιστοιχίζουν τις διάφορες εντολές του κώδικα μηχανής με συντομογραφίες λέξεων της Αγγλικής γλώσσας. Ταυτόχρονα εκτός από τα μνημονικά ονόματα σε εντολές, χρησιμοποιήθηκαν και μνημονικά ονόματα σε διευθύνσεις μνήμης και έτσι μειώθηκαν οι πιθανότητες λάθους από απροσεξία, τόσο στον καθορισμό των εντολών, όσο και στον καθορισμό των διευθύνσεων που χειριζόταν το πρόγραμμα.

Έτσι δημιουργήθηκε μία άλλη κατηγορία γλωσσών, οι συμβολικές γλώσσες, οι οποίες, επειδή είναι στενά συνδεδεμένες με την αρχιτεκτονική της μηχανής, ονομάστηκαν και χαμηλού επιπέδου γλώσσες. Επειδή όμως αναγνωρίζει μόνο τη γλώσσα μηχανής έπρεπε και τα προγράμματα που γραφόταν σε συμβολική γλώσσα, να μεταφραστούν σε γλώσσα μηχανής. Αρχικά αυτό γινόταν χειρογραφικά από ανθρώπους που έπαιρναν ένα πρόγραμμα γραμμένο σε συμβολική γλώσσα και το μετέφραζαν σε γλώσσα μηχανής. Οι άνθρωποι αυτοί ονομάστηκαν assemblers, (συναρμολογητές).

Γύρω στα 1950 διαπιστώθηκε ότι η μηχανική εργασία της μετάφρασης θα μπορούσε να γίνει από τους υπολογιστές πιο γρήγορα και με περισσότερη ακρίβεια απ' ότι θα γινόταν από τον άνθρωπο. Έτσι γράφτηκαν τα πρώτα μεταφραστικά προγράμματα που μετέτρεπαν ένα πρόγραμμα από συμβολική γλώσσα σε γλώσσα μηχανής.

Τα προγράμματα σε συμβολική (assembly) γλώσσα είναι άμεσα συνδεδεμένη με τον υπολογιστή για τον οποίο γράφτηκαν και δεν μπορούν να μεταφερθούν σε διαφορετικό υπολογιστή. Η κύρια χρήση της γλώσσας χαμηλού επιπέδου προορίζεται για τη συγγραφή προγραμμάτων διαχείρισης του συστήματος, όπως λειτουργικά συστήματα, βοηθητικά προγράμματα (utilities), συστήματα αυτόματου ελέγχου με υπολογιστή κ.λ.π.

Οι συμβολικές γλώσσες είναι το χαμηλότερο επίπεδο στο οποίο ο άνθρωπος μπορεί να προγραμματίσει τον υπολογιστή, λέγονται δε και γλώσσες *χαμηλού* επιπέδου.

Η χρήση των γλωσσών χαμηλού επιπέδου είναι για τη συγγραφή οδηγών συσκευών, τμημάτων του λειτουργικού συστήματος ή τμημάτων εφαρμογών για τα οποία ο προγραμματιστής επιθυμεί να έχει τον απόλυτο έλεγχο της λειτουργίας του επεξεργαστή. Υπάρχουν εργαλεία αυτόματης μετάφρασης προγραμμάτων συμβολικής γλώσσας σε γλώσσα μηχανής που ονομάζονται **συμβολομεταφραστές (assemblers)**.

Υπάρχουν βεβαίως διαφορετικές γλώσσες μηχανής και συμβολικές γλώσσες για διαφορετικές οικογένειες επεξεργαστών. Ένας προγραμματιστής που γράφει κώδικα σε συμβολική γλώσσα ζει σε έναν κόσμο που έχει αρκετή επαφή με την αρχιτεκτονική του επεξεργαστή του υπολογιστή στον οποίο απευθύνεται.

5.2.2. Γλώσσες Υψηλού Επιπέδου

Για να απαλλαγεί ο προγραμματιστής από όλες τις δυσκολίες της συμβολικής γλώσσας, δημιουργήθηκαν γλώσσες προγραμματισμού υψηλού επιπέδου, που είναι προσανατολισμένες προς την διευκόλυνση του ανθρώπου και όχι του υπολογιστή. Μια γλώσσα υψηλού επιπέδου χρησιμοποιεί μόνο μια απλή γραμμή οδηγιών για να αντιπροσωπεύσει πολλές γραμμές συμβολικής γλώσσας ή γλώσσας μηχανής. Αυτό ελαττώνει τον χρόνο ανάπτυξης του προγράμματος και το καθιστά λιγότερο εξαρτημένο από τον συγκεκριμένο επεξεργαστή. Χρησιμοποιούν ως εντολές απλές λέξεις της αγγλικής γλώσσας και ακολουθούν αυστηρούς συντακτικούς κανόνες. Οι εντολές αυτές είναι

περισσότερο κατανοητές και μεταφράζονται από τον υπολογιστή σε γλώσσα μηχανής από ένα ειδικό πρόγραμμα, που ονομάζεται Μεταγλωττιστής ή Μεταφραστής (Compilers) ή Διερμηνευτής (Interpreter), ανάλογα με το προγραμματιστικό περιβάλλον .Τα προγράμματα όμως που είναι γραμμένα σε γλώσσες υψηλού επιπέδου απαιτούν γενικά περισσότερη μνήμη και εκτελούνται με λιγότερη ταχύτητα από τα προγράμματα που είναι γραμμένα σε συμβολική γλώσσα. Δεν χρησιμοποιούν το σύστημα τόσο αποδοτικά όσο τα προγράμματα σε συμβολική γλώσσα.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΓΛΩΣΣΩΝ ΥΨΗΛΟΥ ΕΠΙΠΕΔΟΥ :

Έχουν ένα καθορισμένο σύνολο από λέξεις, σύμβολά και προτάσεις.

Οι εντολές που γράφονται σε γλώσσα υψηλού επιπέδου μεταφράζονται κατά τη διάρκεια της μετάφρασης σε πολλές εντολές του κώδικα μηχανής.

Έχουν ορισμένους γραμματικούς και συντακτικούς κανόνες που πρέπει να γνωρίζει ο προγραμματιστής, όταν γράφει το πρόγραμμα.

Η εξάρτηση της γλώσσας από τη μηχανή είναι ελάχιστη, πολλές φορές μάλιστα η γλώσσα είναι ανεξάρτητη της μηχανής.

Η γλώσσα είναι εφοδιασμένη με ένα μεγάλο αριθμό υποπρογραμμάτων που ονομάζονται βιβλιοθήκη της γλώσσας. Από τη βιβλιοθήκη αυτή ο προγραμματιστής έχει τη δυνατότητα να ενσωματώσει ένα ή περισσότερα υποπρογράμματα μέσα στα προγράμματα εφαρμογών. Επίσης, μπορεί να προσθέσει δικά του υποπρογράμματα.

ΟΙ ΠΙΟ ΔΗΜΟΦΙΛΕΙΣ ΓΛΩΣΣΕΣ ΥΨΗΛΟΥ ΕΠΙΠΕΔΟΥ ΕΙΝΑΙ:

FORTRAN (1957) και **COBOL** (1960), και οι δύο σήμερα είναι ακόμη δημοφιλείς και χρησιμοποιούνται η μεν FORTRAN σε διάφορες επιστημονικές εφαρμογές ενώ η COBOL και η RPG σε εμπορικές εφαρμογές. Το γεγονός ότι υπάρχουν ήδη πολλά προγράμματα σε αυτές τις γλώσσες βεβαιώνει ότι θα υπάρχουν για πολύ καιρό ακόμη.

BASIC (1964) γλώσσα προορισμένη να εισαγάγει τους αρχάριους στον προγραμματισμό. Τα προγράμματα basic όμως είναι γενικά αντιπαραγωγικά και η γλώσσα δεν ενθαρρύνει πολλές εφαρμογές προγραμματισμού.

PASCAL (1968) αναπτύχθηκε από τον Dr. Nicklaus Wirth στο πανεπιστήμιο της Ζυρίχης και πήρε το όνομά της από τον Γάλλο μαθηματικό και φιλόσοφο Blaise Pascal. Είναι απόγονες της ALGOL 60, δημιουργήθηκε κυρίως για διδακτικούς σκοπούς και σχεδιάστηκε για να διδάξει καλές τεχνικές προγραμματισμού. Στην πορεία δέχτηκε μετατροπές και αναθεωρήσεις μέχρι τα τέλη της δεκαετίας του '70 οπότε εμφανίστηκαν μορφές της γλώσσας με δυνατότητα εφαρμογής σε μικροϋπολογιστές. Η πιο γνωστή ήταν η - UCSD Pascal - η χρήση της οποίας αρχικά περιοριζόταν σε mini-υπολογιστές και

αργότερα επεκτάθηκε σε μικροϋπολογιστές. Η σπουδαιότερη από τις εκδόσεις της γλώσσας είναι η Turbo Pascal η οποία εμφανίσθηκε το 1983. Είναι μια γλώσσα γενικής κατεύθυνσης και χαρακτηρίζεται από μεγάλο πλούτο εφαρμογών.

C (1972) καθιερώθηκε ως η κύρια γλώσσα για προγραμματισμό λειτουργικών συστημάτων.

Η **LISP**, η **SMALLTALK** και η **PROLOG** χρησιμοποιούνται για εφαρμογές τεχνητής νοημοσύνης.

ΧΡΗΣΗ	ΓΛΩΣΣΑ
Επιστημονικές Εφαρμογές	FORTRAN, C και APL
Εμπορικές Εφαρμογές	COBOL και RPG
Εκπαιδευτικές Εφαρμογές	BASIC και LOGO
Ειδικές Εφαρμογές	LISP και PROLOG
Επιστημονικές-Εμπορικές Εφαρμογές	PL-1, PASCAL κ.λ.π.

Pascal

Η γλώσσα Pascal σχεδιάστηκε από τον Nicklaus Wirth, διάσημο Ελβετό επιστήμονα της Πληροφορικής, το 1968 και αναθεωρήθηκε το 1972. Ο Nicklaus Wirth σχεδίασε την Pascal προκειμένου να ξεπεραστούν τα μειονεκτήματα των γλωσσών προγραμματισμού της δεκαετίας του1960. Πήρε το όνομά της προς τιμή του μαθηματικού και φιλόσοφου Blaise Pascal.
Η Pascal σχεδιάστηκε με στόχο να χρησιμοποιηθεί ως διδακτικό εργαλείο των αρχών του προγραμματισμού. Λόγω όμως της πληρότητάς της, της απλότητας της και της ευκολίας στην εκμάθησή της χρησιμοποιείται ευρέως στις επιχειρήσεις, τη βιομηχανία και τους προσωπικούς υπολογιστές.
Η Pascal είναι γλώσσα γενικής χρήσης και υποστηρίζει τις αρχές του δομημένου και του τμηματικού προγραμματισμού. Μερικά από τα ιδιαίτερα χαρακτηριστικά της είναι τα εξής:

Η δυνατότητα που δίνεται στον προγραμματιστή να δημιουργεί δικούς του τύπους δεδομένων.

Η χρήση μεταβλητών τύπου δείκτη(pointer) και η δυνατότητα της δυναμικής διαχείρισης της κεντρικής μνήμης.

Η σύνθετη εντολή (compound statement), δηλαδή η χρήση μιας σειράς εντολών ως μία εντολή.

Όπως οι περισσότερες γλώσσες, έτσι κι αυτή, πέρασε από πολλές εκδόσεις διαφόρων κατασκευαστών που κάθε μία εμπλουτιζόταν με περισσότερες δυνατότητες.

Αρχικά είχε μία αδυναμία στον αποτελεσματικό χειρισμό των αρχείων και των αλφαριθμητικών δεδομένων ή συμβολοσειρών (strings) που όμως έχει ήδη ξεπεραστεί σε μεταγενέστερες εκδόσεις. Η έκδοση της για περιβάλλον Windows θεωρείται ως πρότυπο στο χώρο των μικροϋπολογιστών.

Το αλφάβητο της γλώσσας Pascal αποτελείται από βασικά σύμβολα, όπως γράμματα του Ελληνολατινικού αλφαβήτου, τα αριθμητικά ψηφία (0-9) και τα ειδικά σύμβολα, όπως +, -, *, /, \, . , ; κλπ.

Η Pascal μας επιτρέπει να δίνουμε ταυτότητες ή ονόματα τα οποία αναφέρονται σε σταθερές, μεταβλητές, τύπους δεδομένων, διαδικασίες, συναρτήσεις κλπ. Ένα όνομα αποτελείται από μία σειρά χαρακτήρων η οποία πρέπει να αρχίζει πάντοτε με γράμμα και δεν πρέπει να περιέχει κενά.

Προκειμένου να εξασφαλίσουμε ένα ευανάγνωστο πρόγραμμα, δίνουμε ονόματα ενδεικτικά του περιεχομένου τους και αντί για κενό βάζουμε χαρακτήρα "_" . Ο χαρακτήρας αυτός δεν μπορεί να είναι ο τελευταίος του ονόματος. Για παράδειγμα basikos_mistos αυτό είναι σωστό.

ΒΑΣΙΚΟΙ ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Ένα πρόγραμμα επεξεργάζεται δεδομένα τα οποία μπορεί να είναι αποθηκευμένα εσωτερικά στη μνήμη ή εξωτερικά στο δίσκο ή τη δισκέτα ή να γίνεται η εισαγωγή τους από το πληκτρολόγιο ή το σαρωτή κ.λ.π. Στην Pascal κάθε δεδομένο πρέπει να είναι ορισμένου τύπου. Οι τύποι δεδομένων προσδιορίζουν τον τρόπο παράστασης των δεδομένων εσωτερικά στον υπολογιστή καθώς και το είδος της επεξεργασίας τους από τον υπολογιστή. Μερικοί τύποι δεδομένων χρησιμοποιούνται τόσο συχνά που η Pascal τους έχει προσδιορίσει και έτσι υπάρχουν έτοιμοι για το χρήστη. Στον προγραμματισμό ο προγραμματιστής προσδιορίζει και άλλους τύπους δεδομένων ανάλογα με το είδος της επεξεργασίας.

Οι τύποι των μεταβλητών χρησιμοποιούνται για τον ορισμό των μεταβλητών ή των συναρτήσεων που ορίζει ο χρήστης. Μία μεταβλητή θα είναι πάντοτε ενός συγκεκριμένου τύπου.

Οι προσδιορισμένοι από την Pascal απλοί ή στοιχειώδεις τύποι δεδομένων είναι οι :

1. Ακέραιος τύπος
2. Πραγματικός τύπος
3. Λογικός τύπος
4. Χαρακτήρας

Σύνθετοι τύποι δεδομένων είναι αυτοί που ορίζονται από απλούς τύπους ή και από άλλους σύνθετους.

ΑΚΕΡΑΙΟΣ

Οι ακέραιοι τύποι είναι οι γνωστοί μας ακέραιοι που μπορεί να είναι θετικοί ή αρνητικοί, πχ. 10100-10009 + 51-52 κλπ. Θεωρητικά ένας ακέραιος μπορεί να έχει οποιοδήποτε πλήθος ψηφίων. Πρακτικά όμως το πλήθος των ψηφίων περιορίζεται ανάλογα με τον τύπο του υπολογιστή. Η μεταβλητή MaxInt (μέγιστος ακέραιος) προσδιορίζει το εύρος του διαστήματος των ακεραίων από –MaxInt-1 έως MaxInt. Στην Turbo Pascal MaxInt = 32768 δηλαδή, το εύρος του διαστήματος των ακεραίων είναι από –32768 έως 32767.

Έτσι οι ακέραιοι της Pascal είναι ένα υποσύνολο των ακεραίων που γνωρίζουμε από τα Μαθηματικά. Με τους ακεραίους γίνονται όλες οι γνωστές από τα Μαθηματικά πράξεις αλλά τα αποτελέσματα πρέπει να βρίσκονται στο σύνολο τιμών του ακεραίου τύπου.

ΑΚΕΡΑΙΟΙ ΤΥΠΟΙ

Δήλωση τύπου	Διάστημα τιμών	Πρόσημο	Πλήθος bytes
shortint	-128..127	ΝΑΙ	1
integer	-32768..32767	ΝΑΙ	2
longint	-2148483648..2147483647	ΝΑΙ	4
byte	0..255	ΟΧΙ	1
word	0..65535	ΟΧΙ	2

Οι επιτρεπτές πράξεις ακεραίων είναι :	Παράδειγμα
+ πρόσθεση	
- αφαίρεση	
* πολλαπλασιασμός	
Div ακέραια διαίρεση(πηλίκο)	27 mod 6 = 4
Mod υπόλοιπο διαίρεσης	36 mod 6 = 0

ΠΡΑΓΜΑΤΙΚΟΣ

Ο πραγματικός τύπος χρησιμοποιείται εκεί που οι αριθμητικές τιμές δεν είναι ακέραιοι αριθμοί ή οι αναμενόμενες τιμές τους είναι εκτός ορίων του ακεραίου τύπου. Οι πραγματικοί αριθμοί της Pascal ορίζονται ως ένα υποσύνολο των πραγματικών αριθμών που ξέρουμε από τα Μαθηματικά. Περιέχονται στο διάστημα από 10^{-38} μέχρι 10^{+38} περίπου και έχουν από 6 μέχρι και 20 σημαντικά ψηφία ανάλογα με την έκδοση της Pascal και την επιμέρους επιλογή του πραγματικού τύπου.

ΣΧΕΣΙΑΚΟΙ ΤΕΛΕΣΤΕΣ

Περιγραφή	Μαθηματικά	Pascal
Μεγαλύτερο από	>	>
Μικρότερο από	<	<
Μεγαλύτερο ή ίσο	≥	>=
Μικρότερο ή ίσο	≤	<=
Διάφορο	≠	<>
Ανήκει	Є	In

ΧΑΡΑΚΤΗΡΑΣ

Ο τύπος (char) περιγράφει δεδομένα ενός χαρακτήρα μέσα σε ένα υποσύνολο των χαρακτήρων του υπολογιστή. Σε ένα πρόγραμμα Pascal μια τιμή ενός δεδομένου αυτού του τύπου γράφεται : 'A', 'B','$', '&' κλπ.
Ο τύπος χαρακτήρας (char) είναι ένας διατεταγμένος τύπος ο οποίος περιλαμβάνει το σύνολο των χαρακτήρων που διαθέτει ο υπολογιστής μας. Η διάταξη του συνόλου των χαρακτήρων διαφέρει από υπολογιστή σε υπολογιστή. Γενικά πάντως τα ψηφία 0,1,2,...9 είναι συνεχόμενα.

Το ίδιο και τα γράμματα A,B,C,D,.....Z, a,b,c,d,.....z και ακολουθούν οι Ελληνικοί χαρακτήρες Α,Β,Γ,Δ,....Ω, α,β,γ,...ω.

ΑΛΦΑΡΙΘΜΗΤΙΚΟΣ ΤΥΠΟΣ

Ο Αλφαριθμητικός (string)τύπος δεν συνιστάται στην Standard Pascal. Ο αλφαριθμητικός τύπος είναι μία σειρά από 255, το πολύ χαρακτήρες. Εάν στη δήλωση αυτού του τύπου δεν έχει αναφερθεί το μήκος, τότε λαμβάνεται το μέγιστο μήκος, δηλαδή 255 χαρακτήρες. Το περιεχόμενο μιας μεταβλητής αυτού του τύπου μπορεί να είναι μία λέξη ή μία φράση, όπως ονοματεπώνυμο, διεύθυνση κατοικίας κλπ. Τα οποία περιλαμβάνονται μεταξύ εισαγωγικών. Για παράδειγμα επιτρεπτές τιμές αυτού του τύπου μπορεί να είναι :

'Turbo Pascal',
'ΤΕΕ Τεχνικά Επαγγελματικά Εκπαιδευτήρια',
'Τομέας Πληροφορικής' κλπ.

Ένα string μπορεί να αποτελείται από 0-255 το πολύ χαρακτήρες. Ένα string με 0 χαρακτήρες είναι το κενό και δηλώνεται με δύο εισαγωγικά χωρίς κενό μεταξύ τους. Η Turbo Pascal παρέχει λειτουργίες για την συνένωση αλφαριθμητικών τύπων, και για την απομάκρυνση χαρακτήρων από ένα string και για τη σύγκριση των τιμών αλφαριθμητικών τύπων.

Basic

Η **BASIC** είναι η πιο διαδεδεμένη γλώσσα στο χώρο των μικροϋπολογιστών. Το όνομά της προέρχεται απο τα αρχικά των λέξεων **Beginner's All Purpose Symbolic Instruction Code**, που σε ελεύθερη απόδοση σημαίνει Συμβολικός Κώδικας Εντολών Κάθε Χρήσης για τους Αρχάριους. Η γλώσσα άρχισε να αναπτύσεται στο Dartmouth το 1963 για εκπαιδευτικούς σκοπούς. Είναι μια απλή γλώσσα προγραμματισμού. Υπάρχουν περισσότεροι διάλεκτοι της BASIC απ' ότι για οποιαδήποτε άλλη γλώσσα. Μια διάλεκτός της είναι γνωστή Visual Basic.

Η ΓΛΩΣΣΑ VISUAL BASIC

Η γλώσσα Visual Basic είναι μια δομημένη γλώσσα με τύπους που υποστηρίζει, μερικώς, τον προγραμματισμό με αντικείμενα (δεν υποστηρίζει κληρονομικότητα).

Στις επόμενες παραγράφους περιγράφουμε πολύ συνοπτικά ορισμένα βασικά στοιχεία της:

Ορισμός συναρτήσεων

Διαδικασίες (συναρτήσεις που δεν επιστρέφουν τιμή) ορίζονται αντίστοιχα, αλλά με τη λέξη sub αντί για function.

Τύποι

Ορισμένοι βασικοί τύποι είναι:

boolean integer single double string

Ορισμός μεταβλητών

Μεταβλητές ορίζονται με τη σύνταξη:

Εντολές

Οι εντολές τερματίζονται από το τέλος της γραμμής Παράδειγμα:

Σχόλια

Τα σχόλια αρχίζουν με το χαρακτήρα '

Δομές ελέγχου

Η Visual Basic παρέχει τις παρακάτω δομές ελέγχου (σε σχόλιο η αντίστοιχη δομή της C).

Cobol

Η **COBOL** είναι γλώσσα προγραμματισμού, της οποίας το όνομα προήλθε από τα αρχικά των λέξεων **CO**mmon **B**usiness **O**riented **L**anguage (Κοινή Γλώσσα Προσανατολισμένη για Εμπορικές Εφαρμογές). Η COBOL αναπτύχθηκε με σκοπό να καλύψει τις δυσχέρειες που παρουσίαζε η FORTRAN ως γλώσσα προγραμματισμού για εμπορικές εφαρμογές.

Βασικό χαρακτηριστικό της COBOL είναι η χρησιμοποίηση της Αγγλικής γλώσσας με προτάσεις, παραγράφους και σημεία στίξης. Η COBOL είναι μία γλώσσα με πολλά στοιχεία μιας φυσικής γλώσσας και έντονη την ύπαρξη του συντακτικού της. Διαβάζοντας μία παράγραφο σε COBOL, εύκολα καταλαβαίνει κανείς το τι κάνουν οι εντολές που διαβάζει. Για το λόγο αυτό η COBOL έγινε πολύ δημοφιλής στο χώρο των εμπορικών εφαρμογών σε σύντομο χρονικό διάστημα.

Fortran

Η γλώσσα **FORTRAN** (από τα αρχικά **FOR**mulae **TRAN**slator - μεταφραστής τύπων) είναι μία από τις πρώτες γλώσσες υψηλού επιπέδου, η οποία χρησιμοποιήθηκε κυρίως σε επιστημονικές αλλά και σε εμπορικές εφαρμογές. Δημιουργήθηκε τη δεκαετία του 1950 από την IBM και χρησιμοποιείται μέχρι και σήμερα. Αρχικά η FORTRAN ήταν προσανατολισμένη στην επίλυση μαθηματικών προβλημάτων.

Εκδόσεις

Υπάρχουν οι εξής 5 τυποποιημένες εκδόσεις της FORTRAN:

FORTRAN-66
FORTRAN-77
FORTRAN-90
FORTRAN-95
FORTRAN 2003

Βάση για την περιγραφή της FORTRAN παρακάτω θα είναι η έκδοση FORTRAN-77.

Αλφάβητο

Ένα πρόγραμμα γραμμένο σε έκδοση της FORTRAN μέχρι και την FORTRAN-77 μπορεί να χρησιμοποιήσει οποιουσδήποτε από τους εξής χαρακτήρες:

Τα 26 κεφαλαία γράμματα του αγγλικού αλφαβήτου: A, B, ..., Z
Τους 10 αραβικούς αριθμούς: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Τους 12 ειδικούς χαρακτήρες: + - * / = ( ) ' . , $:
Το κενό διάστημα.

Από την έκδοση FORTRAN-90 και μετά, προστέθηκαν επιπλέον ειδικοί χαρακτήρες:

Τα 26 μικρά γράμματα του αγγλικού αλφαβήτου: a, b, ..., z

Τύποι εντολών

Οι εντολές της FORTRAN-77 είναι δηλωτικές ή εκτελέσιμες:

Δηλωτικές εντολές

PROGRAM	FUNCTION	SUBROUTINE	BLOCKDATA	IMPLICIT
PARAMETER	INTEGER	REAL	DOUBLEPRECISION	COMPLEX
LOGICAL	CHARACTER	DIMENSION	COMMON	EXTERNAL
ENTRY	EQUIVALENCE	INTRINSIC		

Ορισμός συνάρτησης – πρότασης

SAVE FORMAT DATA

Δηλωτική εντολή τέλους κειμένου προγράμματος

END

Εκτελέσιμες εντολές

OPEN	REWIND	PUNCH	PRINT	INQUIRE
ENDFILE	BACKSPACE	WRITE	READ	CLOSE

(προσάρτησητιμής)=ASSIGN..TO
προσαρτημένο GOTO

χωρίς συνθήκη GOTO	υπολογιζόμενο GOTO	αριθμητικό IF	λογικό IF

IF ..	THEN ..	ELSE IF ..	ELSE ..	END IF
DO..CONTINUE	CALL	RETURN	PAUSE	STOP

Σε επόμενη έκδοση της γλώσσας προστέθηκαν : END WHILE DO

Κατηγορίες εντολών

Οι εντολές της FORTRAN-77 κατατάσσονται σε 4 κατηγορίες:

- Εντολές εκχώρησης ή αντικατάστασης (assignment statements)
- Εντολές εισόδου/εξόδου (input/output statements)

- Εντολές ελέγχου και λογικής (control statements)
- Πληροφοριακές εντολές και εντολές οργάνωσης (specification statements)

Δομή προγραμμάτων

Κατά στήλες
Ένα αρχείο που περιέχει ένα πρόγραμμα FORTRAN χωρίζεται νοητά σε **τρεις ομάδες στηλών**:

Η πρώτη ομάδα, που περιέχει τις στήλες 1-6, χρησιμοποιείται για την εισαγωγή των **συμβόλων σχολίων** (* και c), των **αριθμών εντολών** και **χαρακτήρων συνέχισης γραμμής**. Ειδικότερα:

Τα σύμβολα * ή c είναι προαιρετικά και εισάγονται στην πρώτη στήλη. Ο μεταγλωττιστής αναγωρίζει έτσι την αντίστοιχη γραμμή ως σχόλιο, και την προσπερνάει. Τα σχόλια δεν έχουν επίδραση στον εκτελέσιμο κώδικα, αλλά βοηθάνε τον προγραμματιστή στην κατανόηση και εκσφαλμάτωση του προγράμματος.

Οι αριθμοί εντολών είναι προαιρετικοί 5-ψήφιοι αριθμοί, που εισάγονται στις στήλες 1 ως 5. Είναι χρήσιμοι για εντολές ανακατεύθυνσης της ροής του προγράμματος, αν και σπάνια πια χρησιμοποιούνται μετά την εισαγωγή της έννοιας του δομημένου προγραμματισμού.

Ένας οποιοσδήποτε χαρακτήρας, εκτός του κενού διαστήματος και του μηδέν, υποδεικνύει στον μεταγλωττιστή ότι η αντίστοιχη γραμμή είναι συνέχεια της προηγούμενης. Καθίσταται έτσι δυνατή η συνέχιση μιας πολύ μεγάλης εντολής σε περισσότερες της μίας γραμμής. Ο μέγιστος αριθμός γραμμών για μια εντολή είναι 40.

Στην δεύτερη, κύρια ομάδα στηλών 7-72, εισάγονται οι **εντολές της FORTRAN**.

Τέλος η τρίτη ομάδα, στήλες 73-80, περιέχει κείμενο που αγνοείται από τον μεταγγλωτιστή, εκτός αν πρόκειται για δεδομένα του προγράμματος.

Σημείωση: Στην FORTRAN-90, δεν υπάρχουν οι περιορισμοί των ομάδων στηλών 7-72 και 73-80. Κάθε γραμμή έχει 132 χαρακτήρες. Η συνέχιση γραμμής γίνεται με τον χαρακτήρα &. Επίσης τα σχόλια γίνονται με την τοποθέτηση του συμβόλου ! σε οποιαδήποτε στήλη, ακολουθούμενο από το σχόλιο.

Κατά γραμμές

Ένα πρόγραμμα FORTRAN μπορεί επίσης να χωριστεί κατά 4 τμήματα ως εξής:

Επικεφαλίδα (heading)

Η **επικεφαλίδα** σε ένα πρόγραμμα είναι προαιρετική και εισάγεται στην πρώτη γραμμή. Προσδιορίζει την αρχή του προγράμματος και το όνομά του. Έχει την εξής συγκεκριμένη μορφή:
PROGRAM [όνομα προγράμματος]
Οι αγκύλες δεν εισάγονται. Προσδιορίζουν μια παράμετρο, όπως και στις υπόλοιπες εντολές παρακάτω.

Τεκμηρίωση (documentation)

Ακολουθεί η **τεκμηρίωση**. Είναι επίσης προαιρετικό τμήμα, το οποίο περιέχει σχόλια σχετικά με το πρόγραμμα, όπως:
Σκοπό του προγράμματος,
Σημασία των διαφόρων μεταβλητών,
Πληροφορίες για την είσοδο/έξοδο του προγράμματος,
Περιγραφή τυχόν τυποποιημένων αλγορίθμων,
Όνομα/ονόματα προγραμματιστή/προγραμματιστών,
Ημερομηνία σύνταξης, τροποποίησης, κ.τ.λ.
Σχόλια τεκμηρίωσης μπορεί να βρίσκονται και ανάμεσα από τις εκτελέσιμες εντολές.

Τμήμα προδιαγραφών/Τμήμα δηλώσεων (specification part)

Στο **τμήμα προδιαγραφών/δηλώσεων** δηλώνονται τα ονόματα και οι τύποι των μεταβλητών ή των σταθερών που θα χρησιμοποιηθούν, οι πίνακες, κ.τ.λ.. Οι δηλωτικές εντολές γράφονται όλες πριν από τις εκτελέσιμες εντολές. (Εξαίρεση είναι η εντολή ENTRY).

Εκτελέσιμο τμήμα (execution part)

Τέλος, το σημαντικότερο τμήμα είναι το **εκτελέσιμο**. Περιέχει τις εκτελέσιμες εντολές, με τις οποίες τα δεδομένα της εισόδου μετατρέπονται σε αποτελέσματα.

Τέλος του προγράμματος

Το τέλος του προγράμματος δηλώνεται με την εντολή END.

C++

Από τα μέσα της 10ετίας του 1980 έχει κάνει την εμφάνισή της η C++ (αργότερα, όταν θα έχετε μάθει λίγη C θα δείτε γιατί αυτή η περίεργη ονομασία) που είναι μια επέκταση της C προς την κατεύθυνση του *object-oriented* προγραμματισμού. Το μερίδιο της C++ στους προγραμματιστές αυξάνει διαρκώς, μια και προσφέρεται ιδιαίτερα για δουλειές όπως την κατασκευή φιλικών *user interfaces* (δηλ. περιβαλλόντων αλληλεπίδρασης με το χρήστη), και το object oriented programming προσφέρει ένα πολύ καλό πρότυπο οργάνωσης μεγάλων προγραμμάτων που έχουν να κάνουν όχι τόσο με αριθμητικούς υπολογισμούς αλλά με διαχείριση πολλών διαφορετικών τύπων δεδομένων πάνω στα οποία θέλουμε κάπως να κάνουμε ίδιες εργασίες.

Για παράδειγμα, αν καθήσετε μπροστά σε ένα υπολογιστή με γραφικό περιβάλλον και παράθυρα, όπως τα Miscrosoft Windows ή τα X Windows (το standard γραφικό περιβάλλον για Unix συστήματα) θα δείτε ότι όλα τα αντικείμενα που βρίσκονται πάνω στην οθόνη, είτε παράθυρα είτε διαφόρων τύπων εικονίδια, υποστηρίζουν την έννοια του ``ανοίγματος'' (open). Όταν κάνετε open σε ένα παράθυρο που είναι ελαχιστοποιημένο αυτό μεγαλώνει, όταν κάνετε open σε ένα εικονίδιο που αντιπροσωπεύει ένα modem[2.1]τότε το modem ενεργοποιείται και παίρνει τηλέφωνο, κλπ.

Τα διάφορα εικονίδια δηλ. ανταποκρίνονται στο ίδιο ``σήμα'' (open) με διαφορετικό τρόπο, που εν γένει δεν είναι γνωστός στο περιβάλλον σύστημα αλλά υλοποιείται από το κάθε αντικείμενο (παράθυρο, modem, πίνακα ελέγχου, κλπ) με δικό του ξεχωριστό τρόπο.

Εναλλακτικά θα έπρεπε το περιβάλλον σύστημα να γνωρίζει τα ``εσωτερικά'' του κάθε αντικειμένου που ``φιλοξενεί'', πράγμα ανέφικτο. Αυτού του είδους η προσέγγιση είναι χαρακτηριστική των object oriented συστημάτων.

Το παρακάτω πρόγραμμα τυπώνει "hello, world" στην οθόνη.

Στοιχεία του προγράμματος

Τα προγράμματα της C αποτελούνται από ορισμένες βασικές τάξεις στοιχείων:
Τα κενά, όπως ο χαρακτήρας ' ' και η αλλαγή της γραμμής. Αυτά γενικά δεν επηρεάζουν καθόλου τη συμπεριφορά του προγράμματος. Για παράδειγμα μπορούσαμε να γράψουμε το πρόγραμμα ως εξής:

Χρησιμοποιούμε κενά (*spaces*) για να κάνουμε το πρόγραμμα πιο εύληπτο.

Οι εντολές του προεπεξεργαστή. Αυτές είναι όλες οι γραμμές που αρχίζουν με το χαρακτήρα #. Για την ώρα απλώς τις γράφουμε στην αρχή του προγράμματος για να μπορέσουμε να εκτελέσουμε ορισμένες βασικές λειτουργίες, όπως το να τυπώσουμε τα αποτελέσματά μας.

Τα ονόματα των συναρτήσεων (*functions*) και των μεταβλητών. Αυτά αρχίζουν με έναν λατινικό αλφαβητικό χαρακτήρα (a-z) και μπορούν να ακολουθούνται από άλλους ή/και ψηφία.

Οι δηλώσεις των συναρτήσεων. Αυτές γράφονται ως το όνομα τη συνάρτησης ακολουθούμενο από παρενθέσεις και το περιεχόμενο της συνάρτησης μέσα σε άγκιστρα ({ }). Η συνάρτηση main έχει ειδικό νόημα. Ορίζει το σημείο από το οποία θα αρχίσει να εκτελείται το πρόγραμμα.

Το περιεχόμενο των συναρτήσεων. Αυτό αποτελείται από εντολές. Κάθε εντολή *τερματίζεται* με μια λατινική άνω τελεία (*semicolon*) (; - ελληνικό ερωτηματικό).

Οι εντολές μέσα στη συνάρτηση εκτελούνται με τη σειρά από πάνω προς τα κάτω εκτός να έχουμε ορίσει κάτι διαφορετικό.

Οι κλήσεις άλλων συναρτήσεων. Αυτές αποτελούν ένα βασικό είδος εντολής.

Η κλήση μιας συνάρτησης αποτελείται από το όνομα της συνάρτησης ακολουθούμενο από το όρισμα (*argument*) της συνάρτησης μέσα σε παρενθέσεις.

Αν η συνάρτηση δέχεται πολλαπλές παραμέτρους, τότε αυτές χωρίζονται με κόμματα.

Ορισμός απλών συναρτήσεων

Στο πρόγραμμα που εξετάσαμε εμφανίζονται δύο συναρτήσεις:

η συνάρτηση main την οποία ορίσαμε εμείς και
η συνάρτηση printf την καλέσαμε.

Σε ένα πρόγραμμα είναι δυνατός ο ορισμός (*definition*) και η κλήση (*call*) και άλλων συναρτήσεων. Το παρακάτω πρόγραμμα τυπώνει και αυτό "hello, world" στην οθόνη.

Με τον τρόπο αυτό μπορούμε να ομαδοποιήσουμε τα τμήματα του προγράμματός μας και να χρησιμοποιήσουμε ορισμένα τμήματα πολλές φορές.

Όταν καλείται μια συνάρτηση, αυτή εκτελείται βήμα προς βήμα μέχρι να τελειώσουν οι εντολές που την απαρτίζουν. Τότε η εκτέλεση του προγράμματος επιστρέφει (*returns*) στην επόμενη εντολή από αυτή που κάλεσε τη συνάρτηση.

Τυπική περιγραφή γλωσσών

Σύνταξη (*Syntax*)

Ο τρόπος με τον οποίο τοποθετούνται στη σειρά τα συστατικά στοιχεία της γλώσσας για να αποτελέσουν ένα πρόγραμμα.
Σημασιολογία (*Semantics*)
Η σημασία που αποδίδεται στα συστατικά στοιχεία ενός προγράμματος κατά τη μετάφραση και την εκτέλεσή του.

Παράδειγμα γραμματικής BNF

Μια αριθμητική έκφραση μπορεί να αποτελείται από:
'Αθροισμα ή Διαφορά (ΑΔ)
Πηλίκο ή Γινόμενο (ΠΓ)
Βασικό στοιχείο (Β)

Ο τρόπος που αυτά συνδυάζονται μεταξύ τους εκφράζεται σε BFN ως εξής:

Lisp

Ο συναρτησιακός προγραμματισμός (*functional programming*) βασίζεται στην αποτίμηση συναρτήσεων αντί για την εκτέλεση εντολών. Προγράμματα βασισμένα στο συναρτησιακό προγραμματισμό είναι γραμμένα με βάση συναρτήσεις που αποτιμούν βασικές τιμές. Μια συναρτησιακή γλώσσα προγραμματισμού υποστηρίζει και ενθαρρύνει το συναρτησιακό προγραμματισμό.

Βασικά χαρακτηριστικά του συναρτησιακού προγραμματισμού είναι:

- οι συναρτήσεις ως παράμετροι,
- ράθυμος υπολογισμός (*lazy evaluation*),
- η αναφορική διαφάνεια (*referential transparency*).

Ορισμένες γνωστές συναρτησιακές γλώσσες προγραμματισμού είναι οι:

Erlang Lisp Hope Haskell Miranda ML Scheme

Οι βασικές ιδέες της γλώσσας Lisp αναπτύχθηκαν από τον John McCarthy το 1956 σε ένα ερευνητικό πρόγραμμα για τεχνητή νοημοσύνη. Στόχος του McCarthy ήταν η ανάπτυξη μιας αλγευρικής γλώσσας επεξεργασίας λιστών για έρευνα στην τεχνητή νοημοσύνη. Ανάμεσα στα έτη 1960 και 1965 υλοποιήθηκε η διάλεκτος Lisp 1.5 η οποία τη δεκαετία του 1970 είχε οδηγήσει στις διαλέκτους MacLisp και InterLisp. Στα μέσα της δεκαετίας του 1970 οι Sussman και Steele Jr. ανέπτυξαν τη διάλεκτο Scheme με βάση ιδέες από τη σημασιολογία των γλωσσών προγραμματισμού. Στη δεκαετία του 1980 έγινε προσπάθεια για ενοποίηση των διαλέκτων της Lisp κάτω από το πρότυπο της Common Lisp.

Συναρτήσεις στη Lisp ορίζονται με τη σύνταξη:

(defun όνομα_συνάρτησης (παράμετροι) τιμή)
Υπολογισμός της τιμής μιας συνάρτησης στη Lisp γίνεται με τη σύνταξη:
(όνομα_συνάρτησης παράμετροι)

Ορισμένες χρήσιμες συναρτήσεις είναι οι:

Παράδειγμα (ορισμός της συνάρτησης του παραγοντικού):

Λίστες

Στη Lisp μπορούμε να ορίσουμε μια λίστα (*list*) από τιμές με τη σύνταξη

'(τιμή1 τιμή2 τιμή3 ...)

Η ειδική τιμή nil παριστάνει την κενή λίστα.

Μπορούμε να αναφερθούμε σε σύμβολα με τη σύνταξη 'όνομα.

Παράδειγμα:

Βασικές συναρτήσεις επεξεργασίας λιστών είναι οι παρακάτω:

- (null list) επιστρέφει αληθές αν η λίστα είναι κενή
- (cons val list) επιστρέφει τη λίστα list με πρώτο το στοιχείο val,

- (car list) επιστρέφει το πρώτο στοιχείο μιας λίστας,
- (cdr list) επιστρέφει τα υπόλοιπα (όλα εκτός από το πρώτο) στοιχεία μιας λίστας ή nil αν αυτά δεν υπάρχουν.

Με βάση τις συναρτήσεις αυτές μπορούμε να ορίσουμε πιο σύνθετες συναρτήσεις.

Length

Επιστρέφει το μήκος μιας λίστας.

Append

Ενώνει δύο λίστες.

Reverse

Αντιστρέφει μια λίστα.

Java

Τα χαρακτηριστικά της Java

Ένα από τα βασικά πλεονεκτήματα της Java έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας.

Τα προγράμματα που είναι γραμμένα σε *Java* τρέχουνε ακριβώς το ίδιο σε Windows, Linux, Unix και Macintosh (σύντομα θα τρέχουν και σε Playstation καθώς και σε άλλες παιχνιδομηχανές) χωρίς να χρειαστεί να ξαναγίνει διάταξη αποδελτίωσης (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα.

Για να επιτευχθεί όμως αυτό χρειαζόταν κάποιος τρόπος ετσι ώστε τα προγράμματα γραμμενα σε Java να μπορουν να είναι «κατανοητά» από κάθε υπολογιστή αναξάρτητα του είδους επεξεργαστή (Intel x86, IBM, Sun SPARC, Motorola) αλλα και λειτουργικού συστήματος (Windows, Unix, Linux, Unix, MacOS). Ο λόγος είναι ότι κάθε κεντρική μονάδα επεξεργασίας μπορεί και «καταλαβαίνει» διαφορετικό *assembly* κώδικα.

Ο συναρμολογούμενος (*assembly*) κώδικας που τρέχει σε Windows είναι διαφορετικός από αυτόν που τρέχει σε ένα Macintosh. Η λύση δόθηκε με την ανάπτυξη της *Εικονικής Μηχανής* (*Virtual Machine* ή VM ή ΕΜ στα ελληνικά).

Μεταγλώττιση και εκτέλεση

Για να μεταγλωττίσουμε το πρόγραμμα πρέπει να το αποθηκεύσουμε σε ένα αρχείο με όνομα ίδιο με το όνομα της κλάσης που περιέχει τη συνάρτηση main. Το επίθεμα (*suffix*) του αρχείου πρέπει να είναι .java.

Με την εντολή javac *όνομα αρχείου* μεταγλωττίζουμε το πρόγραμμα από Java σε μορφή που να μπορεί να εκτελεστεί.

Για να εκτελέσουμε το πρόγραμμα χρησιμοποιούμε την εντολή java *όνομα κλάσης*.

Παράδειγμα:

Μηχανισμός μεταγλώττισης και εκτέλεσης

Ο μεταγλωττιστής της Java (javac) μετατρέπει το πηγαίο πρόγραμμα από Java σε εντολές της ιδεατής μηχανής Java (*Java virtual machine*) (JVM)

Το περιβάλλον εκτέλεσης της Java (java)
φορτώνει την κλάση που ορίζουμε,
φορτώνει όποιες ακόμα κλάσεις απαιτούνται,
τις συνδέει με την κλάση που ζητήσαμε να εκτελεστεί ορίσαμε, και
εκτελεί τις εντολές JVM αρχίζοντας από τη μέθοδο main.

Συνάρτηση σε Java

Αντίστοιχες εντολές JVM

Στοιχεία του προγράμματος

Τα προγράμματα της Java αποτελούνται από ορισμένες βασικές τάξεις στοιχείων:

Τα κενά, όπως ο χαρακτήρας ' ' και η αλλαγή της γραμμής. Αυτά γενικά δεν επηρεάζουν καθόλου τη συμπεριφορά του προγράμματος. Για παράδειγμα μπορούσαμε να γράψουμε το πρόγραμμα ως εξής:

Χρησιμοποιούμε κενά (*spaces*) για να κάνουμε το πρόγραμμα πιο εύληπτο.
Τα ονόματα των συναρτήσεων (*functions*), κλάσεων (*classes*) και των μεταβλητών. Αυτά αρχίζουν με έναν λατινικό αλφαβητικό χαρακτήρα και μπορούν να ακολουθούνται από άλλους ή/και ψηφία.

Η δήλωση της κλάσης. Αυτή γράφεται ως το όνομα τη κλάσης ακολουθούμενο από το περιεχόμενο της κλάσης μέσα σε άγκιστρα ({ }). Το όνομα του αρχείου πρέπει να είναι ίδιο με το όνομα της κλάσης. Για την ώρα χρησιμοποιούμε την κλάση για να ορίσουμε μέσα τις κάποιες συναρτήσεις. Σε επόμενα μαθήματα θα μάθουμε τι ακριβώς είναι η κλάση και πως οι συναρτήσεις που ορίζονται στην κλάση αυτή είναι στην πραγματικότητα μέθοδοι.

Οι δηλώσεις των συναρτήσεων. Αυτές γράφονται ως το όνομα τη συνάρτησης ακολουθούμενο από παρενθέσεις και το περιεχόμενο της συνάρτησης μέσα σε άγκιστρα ({ }). Αν μια συνάρτηση δέχεται ορίσματα, αυτά δηλώνονται μέσα στις παρενθέσεις που ακολουθούν το όνομα τις συνάρτησης.

Η συνάρτηση main έχει ειδικό νόημα. Ορίζει το σημείο από το οποία θα αρχίσει να εκτελείται το πρόγραμμα.

Το περιεχόμενο των συναρτήσεων. Αυτό αποτελείται από εντολές. Κάθε εντολή *τερματίζεται* με μια λατινική άνω τελεία (*semicolon*) (; - ελληνικό ερωτηματικό).

Οι εντολές μέσα στη συνάρτηση εκτελούνται με τη σειρά από πάνω προς τα κάτω εκτός να έχουμε ορίσει κάτι διαφορετικό.

Οι κλήσεις άλλων συναρτήσεων. Αυτές αποτελούν ένα βασικό είδος εντολής.

Η κλήση μιας συνάρτησης αποτελείται από το όνομα της συνάρτησης ακολουθούμενο από το όρισμα (*argument*) της συνάρτησης μέσα σε παρενθέσεις. Αν η συνάρτηση δέχεται πολλαπλές παραμέτρους, τότε αυτές χωρίζονται με κόμματα.

Ορισμός απλών συναρτήσεων

τον άνθρωπο. Μία συμβολική γλώσσα που επιτρέπει στον προγραμματιστή να χρησιμοποιεί αλφαριθμητικούς κωδικούς πράξεων με μνημονική σημασία, να ορίζει συμβολικές σημασίες της εκλογής του για καταχωριτές μηχανής και μνήμης και να καθορίζει τρόπους απόδοσης διεύθυνσης που τον εξυπηρετούν (π.χ. με δείκτες, με ανακατεύθυνση). Επιτρέπει επίσης τη χρήση διαφόρων αριθμητικών συστημάτων (π.χ. δεκαδικό, δεκαεξαδικό) για αριθμητικές σταθερές, καθώς και την προσάρτηση ετικετών σε γραμμές προγράμματος, ώστε να είναι δυνατή η αναφορά σε αυτές με συμβολικό τρόπο από άλλα σημεία του προγράμματος (συνήθως ως προορισμός μιας μεταφοράς ελέγχου ή ενός άλματος).
Η γλώσσα μηχανής είναι η αναπαράσταση ενός προγράμματος σε οποιαδήποτε γλώσσα προγραμματισμού (basic, c, c++) σε δυαδική μορφή 0 και 1.

Τα πρόωρα συγκροτήματα ηλεκτρονικών υπολογιστών προγραμματίστηκαν κυριολεκτικά με το χέρι. Οι μπροστινοί διακόπτες στο ταμπλό χρησιμοποιήθηκαν για να εισαγάγουν τις οδηγίες και τα στοιχεία. Αυτοί οι διακόπτες αντιπροσώπευσαν τις γραμμές διευθύνσεων, στοιχείων και ελέγχου του συγκροτήματος ηλεκτρονικών υπολογιστών. Για να εισαγάγουν τα στοιχεία στη μνήμη, οι διακόπτες διευθύνσεων επιλέχτηκαν στη σωστή διεύθυνση, οι διακόπτες στοιχείων επιλέχτηκαν στη συνέχεια και τελικά επιλέχτηκε ο WRite διακόπτης. Αυτό έγραψε τη δυαδική αξία στις μπροστινές αλλαγές στοιχείων επιτροπής στη διεύθυνση που διευκρινίστηκε. Μόλις εισήχθησαν όλα τα στοιχεία και οι οδηγίες, ο διακόπτης εκκινήσεις επιλέχτηκε για να τρέξει το πρόγραμμα.

Ο προγραμματιστής έπρεπε επίσης να ξέρει το σύνολο των οδηγιών το επεξεργαστή. Κάθε οδηγία έπρεπε να μετατραπεί με το χέρι στα σχέδια κομματιών από τον προγραμματιστή έτσι οι μπροστινοί διακόπτες εκκίνησης να μπορούσαν να τεθούν σωστά. Αυτό οδήγησε σε λάθη κατά τη μετάφραση, όπως ο προγραμματιστής μπόρεσε εύκολα να παερμηνεύσει το 8 ως αξία Β. Έγινε προφανές ότι τέτοιες μέθοδοι ήταν αργές και λάθος επιρρεπές.

Με την εμφάνιση του καλύτερου υλικού που θα μπορούσε να εξετάσει τη μεγαλύτερη μνήμη, και την αύξηση στο μέγεθος μνήμης (λόγω των καλύτερων τεχνικών παραγωγής και του χαμηλότερου κόστους), τα προγράμματα γράφτηκαν για να εκτελέσουν μερικές από χειροκίνητες εισόδους. Τα μικρά προγράμματα οργάνων ελέγχου έγιναν δημοφιλή, τα οποία επέτρεψαν την είσοδο των οδηγιών και των στοιχείων μέσω των αριθμητικών πλήκτρων ή των τερματικών δεκαεξαδικού. Οι πρόσθετες συσκευές όπως η ταινία εγγράφου και οι τρυπημένες με διατρητική μηχανή κάρτες έγιναν δημοφιλείς ως μεθόδους αποθήκευσης για τα προγράμματα.

Τα προγράμματα ήταν ακόμα κωδικοποιημένα με το χέρι, δεδομένου ότι η μετατροπή από το μνημονικό στις οδηγίες εκτελούνταν ακόμα με το χέρι. Για να αυξήσει την παραγωγικότητα προγραμματιστών, η ιδέα του γραψίματος ενός προγράμματος για να ερμηνευτή κάποιο άλλο ήταν μια σημαντική ανακάλυψη. Αυτό θα οργανωνόταν από τον υπολογιστή και θα μετέφραζε το πραγματικό μνημονικό στις οδηγίες. Τα οφέλη ενός τέτοιου προγράμματος θα ήταν:

• μειωμένα λάθη
• γρηγορότεροι χρόνοι μεταφράσεων
• οι αλλαγές θα μπορούσαν να γίνουν ευκολότερες και γρηγορότερες

Δεδομένου ότι οι προγραμματιστές έγραφαν τον κώδικα πηγής στο μνημονικό οπωσδήποτε, φάνηκε το λογικό επόμενο βήμα. Το αρχείο πηγής τροφοδοτήθηκε δεδομένου ότι η εισαγωγή στο πρόγραμμα, που μετέφρασε το μνημονικό στις οδηγίες, κατόπιν έγραψε την παραγωγή στην επιθυμητή θέση (χαρτί-ταινία κλπ.). Αυτή η ακολουθία γίνεται αποδεκτή τώρα ως κοινή θέση.

Οι μόνες πρόοδοι ήταν η αυξανόμενη χρήση των υψηλού επιπέδου γλωσσών για την αύξηση της παραγωγικότητας των προγραμματιστών.

Ο προγραμματισμός Συμβολικής γλώσσας (Assembly) γράφει τις οδηγίες μηχανών με μνημονική μορφή, χρησιμοποιώντας μια assembler για να μετατρέψει αυτό το μνημονικό στις πραγματικές οδηγίες επεξεργαστών και τα σχετικά στοιχεία.

Τα μειονεκτήματα του προγραμματισμού Συμβολικής γλώσσας (Assembly) είναι:

• ο προγραμματιστής απαιτεί τη γνώση του συνόλου αρχιτεκτονικής και επιμόρφωσης επεξεργαστών
• πολλές οδηγίες απαιτούνται να επιτύχουν μικρούς στόχους
• τα προγράμματα πηγής τείνουν να είναι μεγάλα και δύσκολο να ακολουθηθούν
• τα προγράμματα είναι μηχανή εξαρτώμενη από τις αλλαγές που πιθανόν να γίνουν στο υλικό και απαιτούν να επανεγγραφούν, εάν αλλάξει αυτό.

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΥΜΒΟΛΙΚΗΣ ΓΛΩΣΣΑΣ (ASSEMBLY)

Asemblers είναι προγράμματα που παράγουν τις οδηγίες κώδικα μηχανών από ένα πρόγραμμα κώδικα πηγής που γράφεται στη Συμβολική Γλώσσα (Assembly).

Τα χαρακτηριστικά γνωρίσματα που παρέχονται από μια assembler είναι:

• επιτρέπει στον προγραμματιστή να χρησιμοποιήσει το μνημονικό όταν γράφει κώδικα πηγής προγραμμάτων.

• οι μεταβλητές αντιπροσωπεύονται από τα συμβολικά ονόματα, και όχι ως θέσεις μνήμης

• ο συμβολικός κώδικας είναι ευκολότερος να διαβάσει και να ακολουθήσει

• ο έλεγχος λάθους παρέχεται

• οι αλλαγές μπορούν γρήγορα και εύκολα να ενσωματωθούν με μια επανασυναρμολόγηση

• οι ενισχύσεις προγραμματισμού συμπεριλαμβάνονται για τον επανεντοπισμό και την αξιολόγηση έκφρασης.

Στο γράψιμο των προγραμμάτων Συμβολικής γλώσσας (Assembly) για τους μικροϋπολογιστές, είναι ουσιαστικό ότι ένα τυποποιημένο σχήμα ακολουθείται.

Οι περισσότεροι κατασκευαστές παρέχουν assembler, οι οποίες είναι προγράμματα που χρησιμοποιούνται για να παράγουν τις οδηγίες κώδικα μηχανών για τον πραγματικό επεξεργαστή. (για να εκτελέσουν.)

Η assembler μετατρέπει το γραπτό πρόγραμμα πηγής Συμβολικής γλώσσας (Assembly) σε ένα σχήμα που τρέχει στον επεξεργαστή. Κάθε οδηγία κώδικα μηχανών (η αξία δυαδικών ή δεκαεξαδικού) αντικαθίσταται από ένα **μνημονικό**.

Ένας μνημονικός είναι μια σύντμηση που αντιπροσωπεύει την πραγματική οδηγία.

Binary	Hex	Mnemonic	
01001111	4F	CLRA	Clears the A accumulator
00110110	36	PSHA	Saves A acc on stack

Οι δηλώσεις Συμβολικής γλώσσας (Assembly) γράφονται μια ανά γραμμή. Ένα πρόγραμμα κώδικα μηχανών αποτελείται από μια ακολουθία δηλώσεων Συμβολικής γλώσσας (Assembly), όπου κάθε δήλωση περιέχει ένα μνημονικό. Κάθε γραμμή ενός προγράμματος Συμβολικής γλώσσας (Assembly) είναι χωρισμένη σε τέσσερις τομείς, όπως παρουσιάζονται πιο κάτω.

LABEL OPCODE OPERAND COMMENTS

Ο τομέας ετικετών είναι προαιρετικός. Μια ετικέτα είναι προσδιοριστική (ή σύμβολο σειρά κειμένων). Οι ετικέτες χρησιμοποιούνται εκτενώς στα προγράμματα για να μειωθεί η εμπιστοσύνη επάνω στους προγραμματιστές που θυμούνται που το στοιχείο ή ο κώδικας βρίσκεται. Μια ετικέτα μπορεί να χρησιμοποιηθεί για να αναφέρει

μια θέση μνήμης, την αξία ενός κομματιού των στοιχείων, την διεύθυνση ενός προγράμματος, μιας υπορουτίνας, μιας μερίδας κλπ. κώδικα.

Το μέγιστο μήκος μιας ετικέτας διαφέρει μεταξύ των assembler. Μερικοί δέχονται μέχρι 32 χαρακτήρες μακροχρόνιους, άλλοι μόνο τέσσερις χαρακτήρες. Μια ετικέτα, όταν δηλώνεται, είναι από μια άνω και κάτω τελεία, και αρχίζει με έναν έγκυρο χαρακτήρα (A.Z). Εξετάστε το ακόλουθο παράδειγμα.

START: LDAA #24H

Εδώ, η ΕΝΑΡΞΗ ετικετών είναι ίση με τη διεύθυνση της οδηγίας LDAA #24H. Η ετικέτα χρησιμοποιείται στο πρόγραμμα ως αναφορά, π.χ.,

JMP START
Αυτό θα οδηγούσε στον επεξεργαστή που πηδά στη θέση (διεύθυνση) που συνδέεται με την ΕΝΑΡΞΗ ετικετών, εκτελώντας κατά συνέπεια την οδηγία LDAA #24H αμέσως μετά από την οδηγία JMP. Όταν μια ετικέτα παραπέμπεται αργότερα στο πρόγραμμα, γίνεται έτσι χωρίς το επίθημα άνω και κάτω τελειών.

Ένα πλεονέκτημα της χρήσης ετικετών είναι ότι η παρεμβολή ή η εκ νέου ρύθμιση των δηλώσεων κώδικα δεν απαιτεί την επανάληψη των πραγματικών οδηγιών μηχανών. Μια απλή επανασυναρμολόγηση είναι όλα αυτό που απαιτείται. Στην χέρι-κωδικοποίηση, τέτοιες αλλαγές μπορούν να πάρουν τις ώρες που εκτελούν.

Κάθε οδηγία αποτελείται από ένα opcode και πιθανόν από έναν ή περισσότερους τελεστές. Στην ανωτέρω οδηγία

JMP START

το opcode είναι JMP και ο τελεστέος είναι η διεύθυνση της ΈΝΑΡΞΗΣ ετικετών.

Ο τομέας opcode περιέχει έναν μνημονικό. Το Opcode αντιπροσωπεύει τον κώδικα λειτουργίας, δηλ., μια οδηγία κώδικα μηχανών.

Το opcode μπορεί επίσης να επιθυμήσει τις πρόσθετες πληροφορίες (τελεστέοι). Αυτές οι πρόσθετες πληροφορίες χωρίζονται από το opcode με τη χρησιμοποίηση ενός διαστήματος (ή της στάσης ετικετών).

Ο τομέας τελεστέου αποτελείται από τις πρόσθετα πληροφορίες ή τα στοιχεία που το opcode επιθυμεί. Σε ορισμένους τύπους εξετάσεων των τρόπων, ο τελευταίως χρησιμοποιείται για να διευκρινίσει

- σταθερές ή ετικέτες
- άμεσα στοιχεία
- στοιχεία που περιλαμβάνονται σε έναν άλλο συσσωρευτή ή κατάλογο
- μια διεύθυνση

Σημειώστε ότι ο προγραμματιστής δεν πρέπει να ανησυχήσει για τα σχέδια κομματιών, τις τιμές δεκαεξαδικού και τις διευθύνσεις της ΘΕΣΗΣ ή του ΚΩΔΙΚΑ. Η assembler, όταν τροφοδοτείτε το ανωτέρω πρόγραμμα, θα παραγάγει το σωστό κώδικα.

Η παραγωγή κώδικα από τη assembler θα είναι:

Location 0100 holds the value associated with the label STATUS

Locations 0101 to 0103 perform the LDAA STATUS instruction

Locations 0104 to 0106 perform the JMP CODE instruction

Η δήλωση ORG 0100H στο ανωτέρω πρόγραμμα δεν είναι μια οδηγία κώδικα μηχανών. Είναι μια οδηγία στη assembler, η οποία καθοδηγεί τη assembler για να παραγάγει τον κώδικα που τρέχει στην οριζόμενη διεύθυνση προέλευσης. Οι οδηγίες στις assembler καλούνται ψευδοκώδικας. Αυτοί χρησιμοποιούνται για :

εφαρμογών (προσωπικού, μισθοδοσίας), μεταφορών (κράτηση θέσεων, έκδοση εισιτηρίων) κλπ.

Στο διαδικασιακό προγραμματισμό το πρόγραμμα είναι μια διαδικασία η οποία εκτελεί ένα προς ένα τα βήματα του αλγόριθμου επίλυσης του προβλήματος. Οι εντολές του προγράμματος εκτελούνται διαδοχικά, εκτός αν υπάρχουν συνθήκες, οπότε κάποιες εντολές παραλείπονται και δημιουργούνται διακλαδώσεις ή κάποιες εντολές επαναλαμβάνονται.

Ανάλογα φαινόμενα επικρατούν για παράδειγμα κατά την κίνηση των οχημάτων όπου σύμφωνα με τις κυκλοφοριακές συνθήκες εκτελούνται διακλαδώσεις και επαναλήψεις δρομολογίων κατά ορισμένα χρονικά διαστήματα.

Οι γλώσσες της τρίτης γενιάς κατάφεραν να απομακρύνουν τον προγραμματισμό από το επίπεδο της μηχανής προς το επίπεδο του ανθρώπου. Το αποτέλεσμα ήταν να γίνει ο προγραμματισμός μια ευχάριστη και δημιουργική εργασία, στην οποία οι προγραμματιστές απελευθερωμένοι από δουλειές ρουτίνας πέτυχαν να δημιουργήσουν νέα εργαλεία και τεχνικές και η εξέλιξη της Πληροφορικής να συνεχίζεται από τότε με γρήγορους ρυθμούς.

5.3.2. Δομημένος Προγραμματισμός

Ο Δομημένος Προγραμματισμός (structural Programming) προυποθέτει δομημένη σχεδίαση και έλεγχο ενός δομημένου προγράμματος, που αποτελείται από ανεξάρτητα τμήματα (modules), με βάση ένα προκαθορισμένο σχέδιο.

Χρησιμοποιεί μία θεωρητική βάση για την κωδικοποίηση των προγραμμάτων, όπου η κύρια ιδέα είναι η χρησιμοποίηση των βασικών αλγοριθμικών δομών για τη δημιουργία πολύπλοκων προγραμμάτων. Οι δομές αυτές είναι η διαδοχή, η απλή επιλογή και η επανάληψη.

Εκτός από τις δομές αυτές χρησιμοποιούνται και τεχνικές ανάπτυξης και σχεδίασης όπως του ιεραρχικού και του τμηματικού προγραμματισμού.

5.3.3. Παράλληλος Προγραμματισμός

Όπως στις περισσότερες περιπτώσεις εξέλιξης του λογισμικού, ο Παράλληλος προγραμματισμός (Parallel Programming) οφείλει την καθιέρωσή του στην εξέλιξη του υλικού. Η εμφάνιση της αρχιτεκτονικής των πολλών επεξεργαστών οι οποίοι χρησιμοποιούν κοινή μνήμη είχε ως συνέπεια την ανάπτυξη των παραλλήλων αλγορίθμων οι οποίοι επέβαλαν την καθιέρωση του παράλληλου προγραμματισμού.

Ο Παράλληλος Προγραμματισμός εκτός από τις δομές του παραδοσιακού διαδικασιακού προγραμματισμού διαθέτει δομές που επιτρέπουν την ταυτόχρονη εκτέλεση διαδικασιών από διαφορετικούς επεξεργαστές.

Έτσι δίνεται η δυνατότητα ώστε διάφορα υποπρογράμματα ενός προγράμματος να εκτελούνται παράλληλα από δύο ή περισσότερους επεξεργαστές του υπολογιστή.

Η ανάγκη της ταυτόχρονης εκτέλεσης καθώς και της επικοινωνίας μεταξύ των εκτελουμένων διεργασιών καθορίζει και τα πλαίσια των απαιτήσεων στο χώρο του προγραμματισμού. Για την κάλυψη των νέων αναγκών έγινε επέκταση είτε σε κάποιες κλασσικές γλώσσες διαδικασιακού προγραμματισμού όπως οι Ada, Modula 2 Concurrent C είτε στο σχεδιασμό νέων γλωσσών.

Ταυτόχρονα δόθηκε έμφαση στον παράλληλο προγραμματισμό για πιο αποτελεσματική εκμετάλλευση των νέων δυνατοτήτων στα πλαίσια της αρχιτεκτονικής των παραλλήλων επεξεργαστών. Αντιπροσωπευτική γλώσσα σχεδιασμένη για παράλληλη επεξεργασία είναι η γλώσσα Occam.

5.3.4. Αντικειμενοστρεφής Προγραμματισμός

Ο Αντικειμενοστρεφής Προγραμματισμός (Object Oriented Programming) είναι μια τεχνική στην οποία υπάρχει ενσωμάτωση των δεδομένων και του τρόπου χειρισμού αυτών μέσα από την έννοια του αντικειμένου.

Ένα αντικείμενο αποτελείται από μια σειρά δεδομένων που αποτελούν τα χαρακτηριστικά του και μία σειρά μεθόδων ή ενεργειών που σχετίζονται με την επεξεργασία των δεδομένων και καθορίζουν τη συμπεριφορά του αντικειμένου στο πρόγραμμα.

Οι μέθοδοι χειρισμού των δεδομένων μπορεί να είναι διαδικασίες ή συναρτήσεις του χρήστη στο κυρίως πρόγραμμα.

Τα αντικείμενα σε ένα αντικειμενοστρεφές πρόγραμμα μπορεί να σχηματίζουν κλάσεις ιεραρχικά δομημένες. Με την ιεραρχική δόμηση των κλάσεων οι υποκλάσεις κληρονομούν χαρακτηριστικά και ιδιότητες των κλάσεων από τις οποίες προήλθαν, έχοντας επιπλέον την δυνατότητα να προστεθούν σε αυτές (τις υποκλάσεις) νέες ιδιότητες και χαρακτηριστικά.

5.3.5. Συναρτησιακός Προγραμματισμός

Στον Συναρτησιακό Προγραμματισμό (Functional Programming) οι εντολές και οι δομές ελέγχου είναι συναρτήσεις. Ως ορίσματα των συναρτήσεων μπορεί να είναι δεδομένα ή άλλες συναρτήσεις. Έτσι ο συναρτησιακός προγραμματισμός βασίζεται στην έννοια της συνάρτησης όπως την έχουμε γνωρίσει στα μαθηματικά.

Η συνάρτηση δίνει τη δυνατότητα απεικόνισης του πεδίου ορισμού σ' ένα πεδίο τιμών.

Μέσα από τις συναρτήσεις στις οποίες εφαρμόζονται δεδομένα, υλοποιούνται οι διάφορες δομές όπως η διακλάδωση, η επανάληψη κλπ. Οι έννοιες της μεταβλητής και της εκχώρησης του διαδικασιακού προγραμματισμού αντικαθίστανται από την έννοια της δέσμευσης ονομάτων σε τιμές κατά την εφαρμογή των συναρτήσεων.

ιδιαίτερα φιλικό τρόπο. Επιτρέπει την εμφάνιση των δεδομένων σε μορφή κειμένου, εικόνας, κινούμενης εικόνας, ήχου και βίντεο. Οι πληροφορίες στο WWW είναι οργανωμένες σε μορφή υπερκειμένου (hypertext) Ιστοσελίδα. Η ιστοσελίδα (Web page) είναι ένα έγγραφο του ιστού που περιλαμβάνει πληροφορίες σε πολλές μορφές: κείμενο, εικόνα, κινούμενη εικόνα, ήχος, Video.

6.4.2. Υπηρεσία του Ηλεκτρονικού Ταχυδρομείου

Το *ηλεκτρονικό ταχυδρομείο (E-mail)* είναι η πιο διαδεδομένη υπηρεσία του διαδικτύου και αποτελεί έναν ταχύτατο, φθηνό και αποδοτικό τρόπο επικοινωνίας μεταξύ χρηστών του Internet σε ολόκληρο τον κόσμο. Είναι μια μορφή επικοινωνίας η οποία επιτρέπει στους χρήστες του διαδικτύου να στείλουν ένα μήνυμα σε άλλους χρήστες, που έχουν *ηλεκτρονική διεύθυνση* (*e-mail address*) με τρόπο που μμοιάζει με αυτόν του κλασικού ταχυδρομείου.

Κάθε μήνυμα χαρακτηρίζεται από την ηλεκτρονική διεύθυνση του αποστολέα, το περιεχόμενο (που μπορεί να είναι απλό κείμενο, εικόνα, επισυναπτόμενο αρχείο κ.ά.) και την ηλεκτρονική διεύθυνση του παραλήπτη. Τα μηνύματα φυλάσσονται σε ηλεκτρονικά γραμματοκιβώτια (mailboxes) μέχρι την ανάκτησή τους.

6.4.3. Υπηρεσία Μεταφοράς Αρχείων

Τα αρχεία στο Internet μπορούν να διακινούνται από τον ένα υπολογιστή στον άλλο, επικοινωνώντας με μια κοινή γλώσσα *(πρωτόκολλο)* που ονομάζεται File Transfer Protocol (FTP). Στο Διαδίκτυο υπάρχει πλήθος από τοποθεσίες FTP (FTP sites) από τα οποία μπορεί κάποιος να «κατεβάσει» (Download) αρχεία, δηλαδή, να τα μεταφέρει από τον απομακρυσμένο υπολογιστή στον υπολογιστή του ή να «ανεβάσει» αρχεία, δηλαδή να τα στείλει στον απομακρυσμένο υπολογιστή.

Το FTP ήταν ο βασικός τρόπος μμεταφοράς αρχείων, αλλά σήμερα χρησιμοποιείται ευρέως και ο παγκόσμιος ιστός. Υπάρχουν πολλές τέτοιες μμεγάλες «αποθήκες αρχείων» και έχουν σχεδιαστεί, έτσι ώστε να μπορούν οι χρήστες να αναζητούν και να βρίσκουν τα προγράμματα που τους ενδιαφέρουν. Για παράδειγμα μπορούμε να βρούμε έναν screen saver, ένα νεότερο οδηγό (driver) για μια συσκευή μας ή ένα πρόγραμμα ελέγχου ιών. Αυτές οι «αποθήκες» αρχείων ονομάζονται FTP sites. Όταν δεν χρειάζεται ο χρήστης να δηλώσει τα στοιχεία του, για να έχει πρόσβαση σ' αυτές ονομάζονται ανώνυμες (anonymous). Υπάρχουν εκατοντάδες anonymous FTP sites στον κόσμο που προσφέρουν έναν πλούτο πληροφοριών και προγραμμάτων.

6.4.4. Υπηρεσίες Telnet

Όπως αναφέραμε ήδη μπορούμε μέσω του Internet να εκτελέσουμε προγράμματα σε άλλους υπολογιστές. Η διαδικασία αυτή έχει ως εξής: Κάνουμε κλικ στο μενού *Έναρξη* ▸ *Εκτέλεση*. Στο παράθυρο διάλογου *Εκτέλεση,* στο πεδίο *Άνοιγμα:* πληκτρολογούμε *telnet* και πατάμε το κουμπί

OK. Στο παράθυρο διαλόγου που ακολουθεί επιλέουμε το μενού *Σύνδεση Απομακρυσμένο σύστημα*. Στο πλαίσιο *Όνομα κεντρικού υπολογιστή*, πληκτρολογούμε ή επιλέγουμε το όνομα του απομακρυσμένου συστήματος με το οποίο θέλουμε να συνδεθούμε. Κατόπιν κάνουμε κλικ στο κουμπί *Σύνδεση*.

Το **Telnet** είναι η υπηρεσία του Internet που μας επιτρέπει να συνδεόμαστε με έναν απομακρυσμένο υπολογιστή και να δουλεύουμε αλληλεπιδραστικά στον υπολογιστή αυτόν χρησιμοποιώντας τα προγράμματά του σαν να είμαστε άμεσα συνδεδεμένοι μαζί του. Ο υπολογιστής μας, μετατρέπεται σε *τερματικό του απομακρυσμένου υπολογιστή*, ο οποίος ανταποκρίνεται στις εντολές μας. Μέσω του Telnet, μπορούμε να συνδεόμαστε με υπολογιστές σε ολόκληρο τον κόσμο και να εκμεταλλευόμαστε την ισχύ τους και τις υπηρεσίες που μας προσφέρουν. Έτσι μπορούμε να χρησιμοποιούμε απομακρυσμένες, για παράδειγμα, βάσεις δεδομένων και άλλες πηγές πληροφόρησης, για να αναζητήσουμε πληροφορίες σε βιβλιογραφικούς καταλόγους διαφόρων βιβλιοθηκών. Για παράδειγμα, μπορούμε να συνδεθούμε με τη βιβλιοθήκη του Παντείου Πανεπιστήμιου στην διεύθυνση www.panteion.gr και ακολουθώντας τις οδηγίες να μπούμε στη βιβλιοθήκη του. Μπορούμε ακόμη να συνδεθούμε με έναν *υπερυπολογιστή (supercomputer)* και να χρησιμοποιήσουμε την ισχύ του για την εκτέλεση πολύπλοκων αλγορίθμων. Όταν συνδεθούμε με τον απομακρυσμένο υπολογιστή, μας ζητείται *όνομα χρήστη (login name)* και *συνθηματικό (password)*. Επομένως, θα πρέπει να έχουμε λογαριασμό (δηλ. δικαίωμα πρόσβασης) στον υπολογιστή αυτό.

Μερικές φορές, για υπηρεσίες που διατίθενται δημόσια, μας υποδεικνύεται από τον απομακρυσμένο υπολογιστή κάποιο ειδικό login name (π.χ. guest), ώστε να μπορέσουμε να συνδεθούμε ακόμη κι αν δε διαθέτουμε λογαριασμό.

6.4.5. Υπηρεσία συνομιλιών με άλλους χρήστες

Η *συνομιλία IRC (Internet Relay Chat)* είναι ένα μέσο γραπτής επικοινωνίας (σε πραγματικό χρόνο) με ανθρώπους από όλο τον κόσμο. Αποτελείται από διάφορα ξεχωριστά δίκτυα από *IRC servers*, μηχανήματα, δηλαδή, τα οποία χρησιμοποιούν οι χρήστες, για να συνδεθούν στο IRC.

Η *υπηρεσία τηλεδιάσκεψης (videoconference)*, επιτρέπει στους χρήστες, διαχειριζόμενοι κατάλληλο υλικό και λογισμικό, να έχουν τη δυνατότητα να συνομιλούν και να ανταλλάσσουν δεδομένα κειμένου, φωνής και εικόνας σε πραγματικό χρόνο (real time). Η ποιότητα αυτών των υπηρεσιών αρχικά δεν ήταν αυτή που θα επέτρεπε ευρεία χρήση τέτοιου είδους εφαρμογών. Όμως η αύξηση του εύρους των δικτύων και η αύξηση της ταχύτητας πρόσβασης των χρηστών δίνουν πλέον τη δυνατότητα να χρησιμοποιούμε εφαρμογές *τηλεδιάσκεψης και τηλεκπαίδευσης*, οι οποίες απαιτούν οπτική και ηχητική επικοινωνία πραγματικού χρόνου μεταξύ των ατόμων που συμμετέχουν. Αναλυτικά θα αναφερθούμε στο οικείο κεφάλαιο της τηλεκπαίδευσης.

6.4.6. Υπηρεσία συζητήσεων

Η υπηρεσία *συζητήσεων (Usenet news ή Newsgroups)* δίνει τη δυνατότητα σε ανθρώπους από όλο τον κόσμο, να συμμετέχουν σε ανοιχτές συζητήσεις πάνω σε θέματα που τους ενδιαφέρουν. Οι συζητήσεις αυτές

πραγματοποιούνται σε χώρους, που λειτουργούν σαν πίνακες ανακοινώσεων. Κάθε χρήστης μπορεί να στείλει το μήνυμά του (άρθρο) και οι άλλοι χρήστες μπορούν να διαβάσουν το άρθρο του και, αν επιθυμούν, να απαντήσουν σε αυτό. Οι απαντήσεις στέλνονται και αυτές στον ίδιο χώρο, ώστε να μπορούν και αυτές με τη σειρά τους να διαβαστούν από όλους τους υπόλοιπους χρήστες.

Οι *λίστες ηλεκτρονικού ταχυδρομείου (mailing lists)* είναι ένας αυτοματοποιημένος μηχανισμός που διανέμει ένα μήνυμα (mail) σε πολλούς χρήστες ταυτόχρονα, χωρίς ο αποστολέας να γράφει τις διευθύνσεις των παραληπτών. Δηλαδή, ο χρήστης στέλνει το email σε μια «εικονική» διεύθυνση (π.χ. *gr-schools@pi-schools.gr)* και αυτό καταλήγει σε όλους τους εγγεγραμμένους στη συγκεκριμένη λίστα. Σε μια τέτοια λίστα, όπως είναι φυσικό, υπάρχουν άτομα με κοινά ενδιαφέροντα που επιθυμούν να ανταλλάσσουν e-mail για θέματα που τους απασχολούν.

6.4.7. Υπηρεσία Αναζήτησης Πληροφοριών

Η μηχανή αναζήτησης είναι μια υπηρεσία που διαθέτει μια βάση δεδομένων με καταγεγραμμένα στοιχεία για τις πληροφορίες που υπάρχουν στο Internet. Ο χρήστης αναζητεί αυτό που θέλει με βάση κάποια συγκεκριμένα κριτήρια - *λέξεις κλειδιά (keywords)* και η μηχανή αναζήτησης του παρουσιάζει τις διευθύνσεις εκείνες στις οποίες έχουν βρεθεί οι λέξεις κλειδιά. Μια πολύ διαδεδομένη μηχανή αναζήτησης είναι το *www.google.com,* το *www.yahoo.com* και άλλα που έχουμε ήδη αναφέρει.

6.5. Internet και λειτουργία του

Το Internet λειτουργεί με το μοντέλο *πελάτη – εξυπηρετητή (client-server),* όπου ο πελάτης πρέπει να αντιλαμβάνεται τα πρωτόκολλα, δηλαδή τις συμφωνημένες παραδοχές επικοινωνίας, των υπηρεσιών που ζητά. Το πρόγραμμα *πελάτης (client)* στο δικό μας υπολογιστή είναι υπεύθυνο για τη μεταφορά των πληροφοριών στον υπολογιστή μας και την παρουσίασή τους με ένα γνωστό και προκαθορισμένο τρόπο. Έτσι απελευθερώνεται ο *εξυπηρετητής (server)* από αυτό το φορτίο και αφιερώνει περισσότερα μέσα για την επεξεργασία των πληροφοριών (εργασία που μόνο αυτός μπορεί να κάνει).

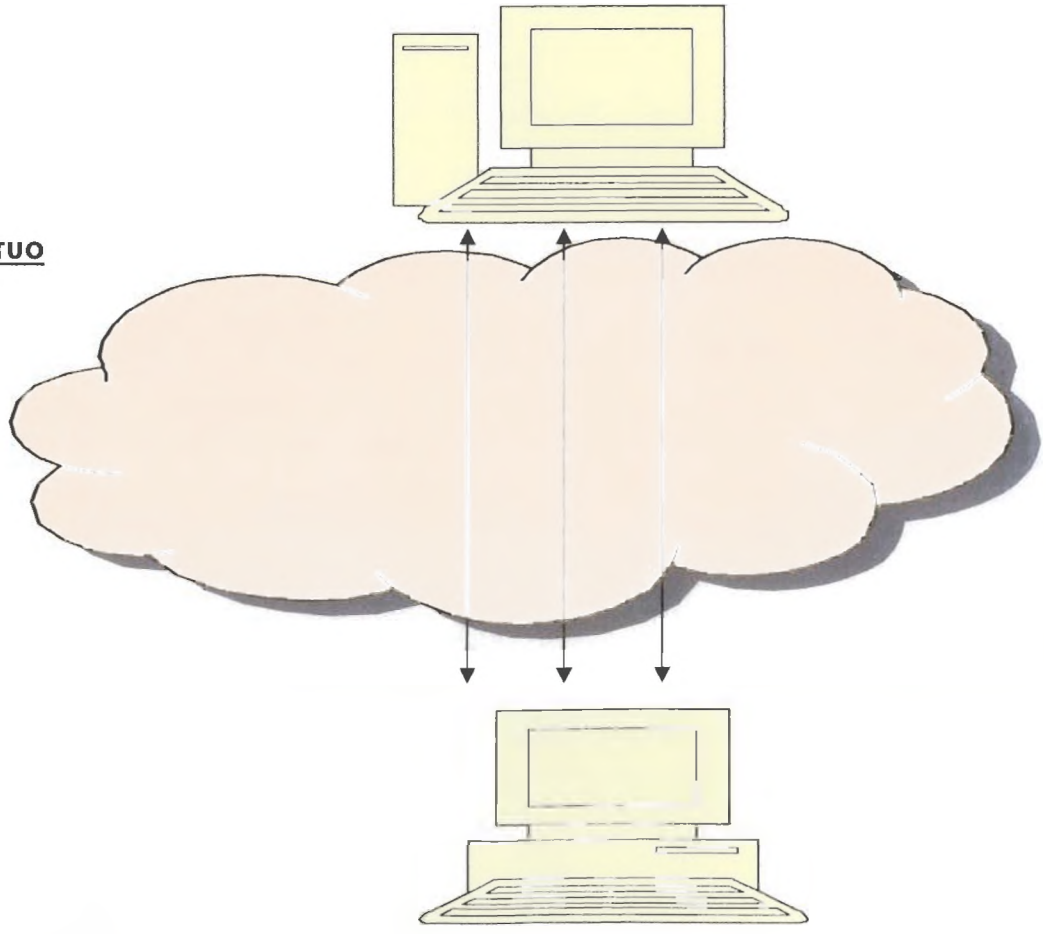
Η ιδέα του διαδικτύου είναι πολύ απλή. Ένας υπολογιστής που ονομάζεται *πελάτης (client)* παίρνει υπηρεσίες και πληροφορίες από έναν άλλο απομακρυσμένο υπολογιστή που ονομάζεται *εξυπηρετητής (server).* Στην καθημερινή μας ζωή, πρωτόκολλο είναι ένα σύνολο από συμβάσεις που καθορίζουν το πώς πρέπει να πραγματοποιηθεί κάποια διαδικασία. Σε ένα δίκτυο, πρωτόκολλο είναι ένα σύνολο από συμβάσεις που καθορίζουν πώς οι υπολογιστές του δικτύου ανταλλάσσουν μεταξύ τους δεδομένα, πώς γίνεται ο έλεγχος και ο χειρισμός λαθών. Το διαδίκτυο χρειάζεται ένα σύνολο από συμβάσεις, οι οποίες να καθορίζουν το πώς ανταλλάσσουν μεταξύ τους οι υπολογιστές δεδομένα, που μπορεί να είναι διαφορετικού τύπου και να ανήκουν σε διαφορετικής τεχνολογίας δίκτυα.

Το TCP/IP, όπως είδαμε και πιο πάνω, είναι το πρωτόκολλο που επιτρέπει στους υπολογιστές που είναι συνδεδεμένοι στα χιλιάδες μικρότερα δίκτυα του Internet να μιλούν μια κοινή γλώσσα, για να συνεννοούνται παρά

τις όποιες διαφορές τους. Το Internet χρησιμοποιεί την τεχνολογία μεταγωγής πακέτων για τη μεταφορά πληροφοριών. Τα δεδομένα κόβονται σε κομμάτια που ονομάζονται πακέτα. Σε κάθε πακέτο μπαίνει μια «επικεφαλίδα» με τις διευθύνσεις του υπολογιστή - αποστολέα και του υπολογιστή – παραλήπτη, για παράδειγμα, 128.17.42.96 255.8.12.45 ΚΑΛΗΜΕΡΑ

(Από: Προς: Δεδομένα:)

Κάθε πακέτο δεδομένων αριθμείται. Ο υπολογιστής – παραλήπτης και ο υπολογιστής - αποστολέας, αλλά όχι οι ενδιάμεσοι υπολογιστές, παρακολουθούν τους αριθμούς των πακέτων και ανταλλάσσουν μεταξύ τους πληροφορίες. Ο παραλήπτης λαμβάνει το πρώτο πακέτο, το δεύτερο, το τρίτο κτλ. Σε περίπτωση που παρουσιαστεί κάποιο πρόβλημα στο δίκτυο, είτε χαθεί κάποιο πακέτο κατά τη διάρκεια της μμετάδοσης, το ξαναζητάει, και ο αποστολέας είναι υπεύθυνος για την αναμετάδοση του. Ο παραλήπτης ελέγχει, επίσης, αν το περιεχόμενο των πακέτων φτάνει σωστά. Η διαδρομή που ακολουθεί ένα πακέτο μέσα από το «σύννεφο» των συνδέσεων δεν είναι προκαθορισμένη.



Σχήμα

Τα δίκτυα στο Internet συνδέονται μεταξύ τους με ειδικούς υπολογιστές που ονομάζονται *δρομολογητές (routers) ή πύλες (gateways).* Ο

δρομολογητής είναι λοιπόν ένας υπολογιστής που συνδέει δύο ή περισσότερα δίκτυα (ίδιου ή διαφορετικού τύπου). Η δουλειά τους είναι να δρομολογούν τα πακέτα των δεδομένων μέσα από τα διάφορα δίκτυα που αποτελούν το Internet, μέχρις ότου τα επιδώσουν στον προορισμό τους. Ο δρομολογητής ελέγχει την επικεφαλίδα του πακέτου και, αν ο παραλήπτης βρίσκεται στο ίδιο δίκτυο με τον αποστολέα, στέλνει κατευθείαν το πακέτο στον παραλήπτη, χωρίς να χρειαστεί να διαβεί τα όρια του δικτύου. Διαφορετικά, το προωθεί στον επόμενο δρομολογητή, που είναι συνδεμένος με το δίκτυο κ.ο.κ, μέχρις ότου το πακέτο προωθηθεί τελικά στο δρομολογητή που είναι συνδεδεμένος στο ίδιο δίκτυο με τον παραλήπτη.

Οι δρομολογητές διατηρούν πίνακες από τους οποίους προσδιορίζουν την κατεύθυνση που πρέπει να πάρει ένα πακέτο, προκειμένου να φτάσει στον προορισμό του. Κάθε φορά, το πακέτο μετακινείται όλο και πιο κοντά προς τον προορισμό του, έως ότου, τελικά, τον φτάσει. Ανάλογα με την κίνηση, ή σε περίπτωση που ένα τμήμα του δικτύου παρουσιάζει πρόβλημα και βρίσκεται προσωρινά σε αχρηστία, οι δρομολογητές επιλέγουν εναλλακτικούς δρόμους.

Κάθε υπολογιστής, κάθε τόπος, κάθε σελίδα με πληροφορίες, κάθε χρήστης που συνδέεται στο Internet έχει τη δική του διεύθυνση. Το σύστημα διευθύνσεων του Internet επιτρέπει σε κάποιον υπολογιστή να συνδέεται με κάποιον άλλον, λόγω του ότι κάθε υπολογιστής έχει το δικό του μοναδικό όνομα και διεύθυνση, όπως κάτι ανάλογο γίνεται και στο διεθνές τηλεφωνικό δίκτυο, όπου κάθε χρήστης έχει ένα μοναδικό αριθμό τηλεφώνου.

Η διεύθυνση ενός υπολογιστή στο Internet αναφέρεται και ως IP διεύθυνση (IP Address) και αποτελεί την «ταυτότητα» του στο διαδίκτυο. Κάθε διεύθυνση αποτελείται από τέσσερα νούμερα (αριθμός δικτύου και αριθμός υπολογιστή μέσα στο συγκεκριμένο δίκτυο). Εκτός της αριθμητικής διεύθυνσης μπορεί να προσδιορίζεται και με ένα όνομα. Τα ονόματα αυτά αποδίδονται με κάποιες συμβάσεις μέσω του συστήματος *DNS (Domain Name System).*

Για παράδειγμα στην *IP Address 147.52.16.2* αντιστοιχεί το όνομα *crete.csd.uch.gr.*

Έχουμε, λοιπόν, ως δεδομένο ότι: Κάθε υπολογιστής είναι μοναδικός στο Internet.

Κάθε οργανισμός που θέλει να συνδέσει στο Internet τους υπολογιστές του, ζητά έναν αριθμό δικτύου από κάποιον επίσημο οργανισμό, που ασχολείται με την κατανομή των διευθύνσεων στο Internet, έτσι ώστε να εξασφαλίζεται η μμοναδικότητα τους.

Σε έναν υπολογιστή του Internet, μπορούν να έχουν πρόσβαση περισσότεροι από ένας χρήστες. Κάθε χρήστης διαθέτει ένα λογαριασμό (account) που περιλαμβάνει *το όνομα χρήστη* και τον *κωδικό πρόσβασης.* Δηλαδή, κάθε χρήστης έχει τη δική του ταυτότητα, το δικό του όνομα χρήστη (user ID), το οποίο του επιτρέπει να ξεχωρίζει από τους άλλους χρήστες που χρησιμοποιούν τον ίδιο υπολογιστή. Για να πάρει άδεια να συνδεθεί, πρέπει να επιβεβαιώσει την ταυτότητα του, παρέχοντας στο σύστημα το προσωπικό του μυστικό κωδικό (password).

Συνδυάζοντας το γεγονός ότι κάθε υπολογιστής είναι μοναδικός στο Internet και κάθε χρήστης μοναδικός στον υπολογιστή που χρησιμοποιεί, καταλήγουμε στο ότι:

και ο Internet Explorer της εταιρείας Microsoft που υπάρχει ενσωματωμένος στις τελευταίες εκδόσεις των Windows, γεγονός που πυροδότησε μια μεγάλη δικαστική αντιμονοπωλιακή διαμάχη στις ΗΠΑ.

6.5.3. Περιήγηση στο Web

Μερικές από τις βασικές δυνατότητες που προσφέρουν σήμερα τα προγράμματα περιήγησης στο Internet είναι και οι εξής:

Πύλη (portal) εισόδου στο Web είναι μια σελίδα που έχει σχεδιαστεί για να ξεκινάμε την περιήγηση στο Διαδίκτυο. Συνήθως περιέχει ειδήσεις και θεματικούς καταλόγους με διευθύνσεις άλλων σελίδων.

Όταν αρχίζουμε να πληκτρολογούμε στη γραμμή διευθύνσεων μια διεύθυνση Web που χρησιμοποιούμε συχνά, εμφανίζεται μια λίστα παρόμοιων διευθύνσεων από τις οποίες μπορούμε να επιλέξουμε.

Μπορούμε να αναζητήσουμε τοποθεσίες Web, κάνοντας κλικ στο κουμπί Αναζήτηση της γραμμής εργαλείων. Στη συνέχεια, στη γραμμή αναζήτησης, πληκτρολογούμε μια λέξη ή μια φράση που να περιγράφει αυτό που ζητάμε.

Όταν εμφανίζονται τα αποτελέσματα της αναζήτησής μας, μπορούμε να προβάλουμε κάθε ιστοσελίδα ξεχωριστά, χωρίς να χάσουμε τη λίστα με τα αποτελέσματα της αναζήτησης.

Μπορούμε να μεταβούμε σε ιστοσελίδες παρόμοιες με αυτή που έχει προβληθεί, χωρίς αναζήτηση.

Για να δούμε μια λίστα ιστοσελίδων που επισκεφθήκαμε πρόσφατα, μπορούμε να κάνουμε κλικ στο κουμπί Ιστορικό της γραμμής εργαλείων.

Για να εμφανισθεί μια ιστοσελίδα, αρκεί να έχουμε πληκτρολογήσει σωστά τη διεύθυνσή της.

Εάν γνωρίζουμε τη διεύθυνση μιας σελίδας, μπορούμε να την πληκτρολογήσουμε στο πεδίο Διεύθυνση: και να πατήσουμε το πλήκτρο Enter, για να εμφανισθεί το περιεχόμενό της.

6.6. Μηχανές Αναζήτησης

Οι *Μηχανές Αναζήτησης (Search Engines)* αποθηκεύουν πληροφορίες για εκατομμύρια σελίδες WEB σε μια τεράστια βάση δεδομένων. Από τα αρχεία που συγκεντρώνονται (με βάση τον τίτλο τους, το πλήρες κείμενο, το μέγεθος τους, τη διεύθυνσή τους κτλ.) δημιουργείται ένα ευρετήριο. Μπορούμε να κάνουμε αναζητήσεις σ' αυτές τις βάσεις δεδομένων, εισάγοντας λέξεις-κλειδιά (keywords).

Παραδείγματα για: Μηχανές Αναζήτησης

www.yahoo.com
www.altavista.com
www.infoseek.com
www.lycos.com
www.google.com
www.askjeeves.com
www.directhit.com
www.infind.com
www.metafind.com
www.cyber411.com

www.profusion.com
www.hack.gr

Μηχανή αναζήτησης είναι ένα λογισμικό που βρίσκει και ταξινομεί τα αποτελέσματα ανάλογα με το ποσοστό συνάφειας του περιεχομένου των ιστοσελίδων σε σχέση με τους όρους της έρευνας. Από τα αποτελέσματα μπορούμε να ανακαλούμε τα έγγραφα (τις σελίδες) που ικανοποιούν τις λέξεις κλειδιά που έχουμε εισάγει. Οι μηχανές αναζήτησης τελευταίας γενιάς επιπλέον, ομαδοποιούν τα αποτελέσματά τους σύμφωνα με το περιεχόμενο, τη δημοτικότητα και το είδος των τόπων, ενώ ορισμένες από αυτές μπορούν να δεχθούν ερωτήσεις και σε φυσική γλώσσα (π.χ. Αγγλικά).

Μεταμηχανές είναι μηχανές αναζήτησης οι οποίες ερευνούν τα ευρετήρια πολλών άλλων μηχανών ταυτόχρονα. Είναι ιδιαίτερα χρήσιμες όταν ψάχνουμε για ένα μμοναδικό όρο ή φράση.

Δεν υπάρχει υπηρεσία ή μηχανή αναζήτησης που να είναι πλήρης. Γενικά, μια μηχανή αναζήτησης μπορεί να περιέχει διευθύνσεις από όλες τις υπηρεσίες του Internet, όπως FTP, World Wide Web, Usenet, Telnet κτλ.

Οι περισσότερες μηχανές αναζήτησης περιορίζονται στην «κατηγοριοποίηση σε καταλόγους» των πληροφοριών εκείνων, που μπορούν να προβληθούν μόνο μέσω του World Wide Web, δηλαδή με βάση το πρωτόκολλο HTTP κατά κύριο λόγο, ενώ ορισμένες υποστηρίζουν επιπλέον και FTP διευθύνσεις του δικτύου.

Τις περισσότερες φορές οι ιστοσελίδες που ανακτώνται ως αποτέλεσμα μιας αναζήτησης, είναι εν μέρει μόνο σχετικές με αυτό που ζητάμε. Πολλές φορές ακόμη στα αποτελέσματα περιλαμβάνονται πολλές ιστοσελίδες από τον ίδιο τόπο.

Παραδείγματα Ελληνικών Μηχανών Αναζήτησης – Ευρετηρίων - Πυλών

www.forthnet.gr
www.hol.gr
www.in.gr
www.flash.gr
www.e-go.gr
www.phantis.com
www.pathfinder.gr
www.robby.gr
www.directory.gr
www.greekpromo.com
www.toxo.gr

6.6.1. Λειτουργία μιας μηχανής αναζήτησης

Το πρώτο ερώτημα που λογικά προκύπτει είναι πώς εισάγονται οι διευθύνσεις σε κάθε μηχανή αναζήτησης, έτσι ώστε να δημιουργηθεί η βάση διευθύνσεων στην οποία κάνουν αναζητήσεις οι χρήστες. Η συνηθέστερη διαδικασία για την εισαγωγή των διευθύνσεων σε θεματικούς καταλόγους είναι

βαρύτητα από τις κοινές λέξεις. Κάθε κείμενο λαμβάνει μια βαρύτητα η οποία ισούται με το άθροισμα των βαρυτήτων κάθε λέξης ερωτήματος που εμφανίζεται στο κείμενο.

6.6.3. Απλή αναζήτηση με λέξεις κλειδιά

Στην πιο απλή μορφή της, η αναζήτηση μπορεί γίνει μόνο με μια μονάχα λέξη ή μια φράση. Όσο πιο λεπτομερής όμως είναι η περιγραφή της αναζήτησης, τόσο καλύτερα αποτελέσματα θα αντλούνται από τη μηχανή αναζήτησης. Οι απλές αναζητήσεις χρησιμοποιούν γενικούς συντακτικούς κανόνες σχετικά με το σχηματισμό φράσεων, τη διάκριση πεζών-κεφαλαίων, και τη χρήση του αστερίσκου (*) ως χαρακτήρα μπαλαντέρ που υποδηλώνει ότι θέλουμε να βρούμε όλες τις λέξεις που περιέχουν μια σύμπτωση για το καθοριζόμενο σύνολο γραμμάτων. Η μηχανή αναζήτησης κατατάσσει τα αποτελέσματα αυτόματα βάσει μιας σειράς παραγόντων που εξασφαλίζουν ότι τα κείμενα που είναι πιθανότερο να είναι σχετικά, εμφανίζονται στην κορυφή της λίστας αποτελεσμάτων. Κατά πόσο πολλαπλές λέξεις ή φράσεις ερωτήματος βρίσκονται κοντά η μια στην άλλη σε ένα κείμενο. Για να αυξήσουμε την πιθανότητα εμφάνισης των πιο σχετικών κειμένων στην κορυφή της λίστας, εισάγουμε πολλά συνώνυμα του θέματος που αναζητάμε. Η εισαγωγή πολλών λέξεων που διαχωρίζονται με κενά, υποδηλώνει ότι θέλουμε να βρούμε κείμενα που περιέχουν κάποιες ή όλες τις λέξεις (τα κείμενα που περιέχουν όλες τις λέξεις εμφανίζονται πρώτα). Μπορούμε να εισάγουμε τη σειρά λέξεων, για παράδειγμα επαγγελματικά δικαιώματα πτυχιούχων ΤΕΙ, αν για παράδειγμα θέλουμε να βρούμε το σχετικό θέμα. Εάν κάποιο κείμενο περιέχει και τις τέσσερις λέξεις, η αυτόματη κατάταξη τοποθετεί αυτό το κείμενο στην κορυφή της λίστας των αποτελεσμάτων. Ακολουθούν τα κείμενα που περιέχουν μόνο κάποιες από τις λέξεις, ενώ τα κείμενα που περιέχουν μόνο μια από τις λέξεις κατατάσσονται τελευταία.

Μπορούμε να εισάγουμε ένα ερώτημα με τη μορφή μιας ερώτησης: για παράδειγμα, *Ποια είναι τα επαγγελματικά δικαιώματα των πτυχιούχων των ΤΕΙ;*
και η μηχανή αναζήτησης ξεχωρίζει τις σημαντικότερες λέξεις της φράσης και επιστρέφει μια λίστα κειμένων που περιέχουν αυτές τις λέξεις. Μπορούμε να χρησιμοποιήσουμε τα σύμβολα + και – ως απλούς τελεστές που απαιτούν την ύπαρξη ή την απουσία κάποιων λέξεων σε μια αναζήτηση. Μπορούμε να χρησιμοποιήσουμε έναν αστερίσκο για την αναζήτηση λέξεων που ξεκινούν με τα ίδια γράμματα. Αυτό εξυπηρετεί στην εύρεση παράγωγων και παραλλαγών της ίδιας λέξης. Για παράδειγμα, εάν εισάγουμε «δικαι*» θα βρούμε σελίδες που περιέχουν λέξεις όπως δίκαιο, δικαιοσύνη, δικαιολογία κτλ. Μπορούμε να χρησιμοποιήσουμε διπλό αστερίσκο για την αναζήτηση όλων των μορφών μιας λέξης.

6.6.4. Το συντακτικό στις μηχανές αναζήτησης – τελεστές και αποτελέσματα που συνεπάγονται

Οι μηχανές αναζήτησης ορίζουν ως λέξη μια σειρά γραμμάτων και ψηφίων που διαχωρίζονται είτε από ειδικούς χαρακτήρες (όπως το κενό, το tab, το τέλος γραμμής, ή αρχή ή το τέλος ενός κειμένου) είτε ειδικά σημεία στίξης (όπως τα : %, $, /, #, _) Για να βρούμε φράσεις, ή ομάδες

συσχετισμένων λέξεων που εμφανίζονται δίπλα η μια στην άλλη, πρέπει να περιβάλλουμε τις λέξεις με διπλά εισαγωγικά. Για παράδειγμα, εάν εισάγουμε «επαγγελματικά δικαιώματα», θα βρούμε όλες τις σελίδες που περιέχουν τη φράση επαγγελματικά δικαιώματα. Ο σχηματισμός φράσεων εξασφαλίζει ότι η μμηχανή αναζήτησης θα βρει τις λέξεις μαζί, αντί να αναζητήσει ανεξάρτητες εμφανίσεις κάθε λέξης. Εάν δε χρησιμοποιήσουμε τα διπλά εισαγωγικά, η μηχανή αναζήτησης βρίσκει τις εμφανίσεις της λέξης «επαγγελματικά» και της λέξης «δικαιώματα», όπως και τις τυχόν εμφανίσεις των δύο λέξεων μαζί. Εάν περιβάλλουμε τις λέξεις σε εισαγωγικά, υποδηλώνει ότι θέλουμε να βρούμε μόνο τις εμφανίσεις και των δύο λέξεων μαζί.

Αποτελέσματα

+ Συμπεριλαμβάνει στα αποτελέσματα αναζήτησης μόνο τα κείμενα που περιέχουν όλες τις καθοριζόμενες λέξεις ή φράσεις

- Εξαιρεί από τα αποτελέσματα αναζήτησης κείμενα που περιέχουν την καθοριζόμενη λέξη ή φράση.

Οι κανόνες αναζήτησης διαφέρουν από μηχανή σε μηχανή. Για παράδειγμα, οι χαρακτήρες «*» και «**» δε χρησιμοποιούνται στη μηχανή αναζήτησης της πύλης www.in.gr, ενώ χρησιμοποιούνται στη μμηχανή αναζήτησης της www.forthnet.gr και της www.hol.gr

Η διάκριση πεζών και κεφαλαίων διαφέρει από μμηχανή σε μμηχανή. Για παράδειγμα, οι google, in.gr, δεν διακρίνουν εάν όλα είναι μμικρά ή κεφαλαία, όμως η Altavista κάνει διάκριση αν το πρώτο γράμμα είναι κεφαλαίο.

Οι μμηχανές αναζήτησης αγνοούν συνήθως τα σημεία στίξης, εκτός της παύλας (π.χ. συλλαβισμένες λέξεις, όπως το CD-ROM, σχηματίζουν αυτόματα μια φράση λόγω της παύλας). Η τοποθέτηση σημείων στίξης ή ειδικών χαρακτήρων ανάμεσα σε κάθε λέξη, χωρίς κενά μεταξύ των χαρακτήρων των λέξεων, είναι ένας άλλος τρόπος υπόδειξης μιας φράσης. (π.χ. η αναζήτηση ενός τηλεφωνικού αριθμού 22280-99-212 είναι ευκολότερη από την εισαγωγή του 22280-99-212, η οποία είναι μια εξίσου αποδεκτή σύνταξη, αλλά λιγότερο φυσική).

6.6.5. Προχωρημένη αναζήτηση με λογικούς τελεστές (τελεστής – σύμβολο – ενέργεια)

Η προχωρημένη αναζήτηση, μας παρέχει μεγαλύτερο έλεγχο στα αποτελέσματα της αναζήτησής μας. Πρέπει, επίσης, να είμαστε πιο ακριβείς, ώστε να έχουμε τα αποτελέσματα που θέλουμε. Μπορούμε να ομαδοποιήσουμε λέξεις σε φράσεις, όπως θα κάναμε για μια απλή αναζήτηση, αλλά επιπλέον μπορούμε να χρησιμοποιήσουμε και λογικούς τελεστές, για να συνδυάσουμε πολλές λέξεις ή φράσεις στην ίδια αναζήτηση. Οι περισσότερες μμηχανές αναζήτησης δίνουν τη δυνατότητα χρήσης λογικών (OR, AND, NOT, NEAR) τελεστών, για να ερευνήσουμε βάσει κανόνων, έτσι ώστε να περιορίσουμε τις πιθανότητες εμφάνισης «άχρηστων» για σας ιστοσελίδων.

Τελεστές

αναζήτησης, όπως και στους θεματικούς καταλόγους, σελιδοποιούνται και μπορούμε να κάνουμε κλικ σε έναν αριθμό στο κάτω μέρος της σελίδας, ή στο κουμπί Επόμενη ή Προηγούμενη σελίδα, για να εμφανισθούν τα επόμενα ή τα προηγούμενα αποτελέσματα.

6.7. Περιγραφή συγκεκριμένων πακέτων

6.7.1. Word, Excel

Το Word της Microsoft είναι ένα πακέτο επεξεργασίας κειμένου, χρησιμοποιείται για τη δημιουργία, διόρθωση, εκτύπωση, αποθήκευση, ανάκτηση και γενικά χειρισμό κειμένου, με τρόπο εύκολο και απαλλαγμένο από όλα τα μειονεκτήματα της κλασικής γραφομηχανής.

Η συγγραφή με τη βοήθεια ενός πακέτου του είδους είναι ιδιαίτερα απλή: με την εκκίνηση τους, εισάγεται ο χρήστης σε μια κενή οθόνη, την οποία μπορεί να θεωρήσει ως το χαρτί της γραφομηχανής και στην οποία μπορεί να αρχίσει αμέσως να γράφει. Η θέση που βρίσκεται στο 'ηλεκτρονικό χαρτί', φαίνεται από το δρομέα που κινείται γράφοντας και αντανακλώντας τις πληκτρολόγησης του. Σε κάποιο άκρο της οθόνης, αναγράφεται η θέση του δρομέα καθώς και κάποιες συμπληρωματικές πληροφορίες.

Ο χρήστης καθορίζει το μήκος της γραμμής, έτσι η μεταπήδηση στην επομένη γραμμή γίνεται αυτόματα. Φυσικά και εδώ μπορούν να οριστούν στηλογνώμονες. Αν, ο χρήστης θέλει να δώσει έμφαση σε μια λέξη ή περιοχή του κειμένου του, μπορεί πολύ εύκολα να την ορίσει έτσι, ώστε να εκτυπωθεί με έντονα γράμματα ή πλάγια ή υπογραμμισμένη ή και σε συνδυασμό των πιο πάνω. Επιπλέον, μπορεί να χρησιμοποιήσει διαφορετικές γραμματοσειρές και σε διάφορα μεγέθη.

Τυχόν λάθη που θα γίνουν κατά τη συγγραφή, δεν απαιτούν γράψιμο από την αρχή. Το μόνο που έχει να κάνει ο χρήστης, είναι να τοποθετήσει το δρομέα πάνω ή δίπλα στο λάθος, να το σβήσει με το πλήκτρο Del ή Backspace και να γράψει το σωστό. Αν χρειαστεί να προσθέσει ένα γραμμα στο μέσον μιας λέξης, μπορεί να πάει στο επόμενο και απλά να το γράψει. Τα δεξιότερα από αυτό θα μεταφερθούν αυτόματα μια θέση δεξιά.

Με το πακέτο αυτό μπορούν να γίνουν ομαδικοί χειρισμοί σε ένα έγγραφο. Αν για παράδειγμα μια πρόταση ή παράγραφος (ή και λέξη) επαναλαμβάνεται, δεν χρειάζεται να γραφτεί πολλές φορές. Αρκεί μια και ένας απλούστατος χειρισμός για να αντιγραφεί στις επιθυμητές θέσεις.

Μια άλλη πολύ χρήσιμη και τυπικά διαθέσιμη δυνατότητα των επεξεργαστών κειμένου, είναι η αναζήτηση μιας λέξης μέσα στο κείμενο. Η αναζήτηση γίνεται με απλούς χειρισμούς. Όμοια με απλούς χειρισμούς μπορεί να γίνει αναζήτηση και αντικατάσταση με μια άλλη. Θα πρέπει να σημειωθεί ότι, σχεδόν όλα τα πακέτα επεξεργασίας κειμένου διαθέτουν πλέον και δυνατότητα ορθογραφικού ελέγχου των κειμένων.

Excel

Το Excel είναι ένα πακέτο φύλλων υπολογισμών, παρέχει στο χρήστη τη δυνατότητα ανάλυσης και πολύπλοκων υπολογισμών αριθμητικών δεδομένων, υπό την προϋπόθεση ότι αυτά μπορούν να οργανωθούν σε

μορφή πίνακα. Τα σημερινά πακέτα περιέχουν επιπλέον τη δυνατότητα γραφικής παράστασης των δεδομένων που περιέχουν.

Οπτικά ένα φύλλο υπολογισμών, είναι μια απλή μεταφορά των κλασικών λογισμικών φύλλων που περιέχουν τεμνόμενες γραμμές και στήλες στην οθόνη του υπολογιστή, δηλαδή στις γραμμές του φύλλου με αριθμούς, ενώ στις στήλες με γράμματα της λατινικής αλφαβήτου. Οι τομές των γραμμών με τις στήλες σχηματίζουν παραλληλόγραμμα που λέγονται κελιά.

Κάθε κελί παίρνει την ονομασία του από τη γραμμή και τη στήλη που το σχηματίζουν. Για παράδειγμα, το κελί που σχηματίζεται από την τομή της γραμμής 1 με τη στήλη Α ονομάζεται Α1 κ.λ.π. σε κάθε κελί μπορούν να τεθούν αριθμητικά δεδομένα, ημερομηνίες ή ακόμα και σχέσεις.

Για παράδειγμα, αν θέλουμε στο κελί Α5 να εμφανίζεται το άθροισμα των κελιών Α1, Α2, Α3, τοποθετούμε ως περιεχόμενο την παράσταση : =Α1+Α2+Α3

Το σημαντικό είναι, ότι οποιεσδήποτε αλλαγές στα κελιά που λαμβάνουν μέρος στους ορούς του αθροίσματος(Α1,Α2,Α3), αντανακλώνται αυτόματα και στο κελί Α5. Με τον τρόπο αυτό, μπορεί κανείς να δοκιμάσει εύκολα και γρήγορα εναλλακτικές λύσεις, που φυσικά μπορεί να περιλαμβάνουν πολύ συνθετότερες εκφράσεις.

Τα πακέτα των φύλλων υπολογισμών έχουν εξελιχθεί σήμερα σε ισχυρά εργαλεία επίλυσης προβλημάτων. Με τη χρήση σημερινών πακέτων του είδους μπορεί κανείς να παραστήσει ένα μοντέλο του προβλήματος με βάση το οποίο να διατυπώσει ερωτήσεις του τύπου 'τι θα γίνει αν ελέγχοντας πρακτικά και άμεσα τι θα συμβεί σε αριθμούς, αν μεταβληθούν κάποιοι άλλοι. Θα πρέπει να σημειωθεί ότι ο όρος πρόβλημα εδώ χρησιμοποιείται με την ευρεία έννοια και αναφέρεται σε οποιοδήποτε τύπο πληροφορίας μπορεί να προσαρμοστεί σε ένα φύλλο, όπως οικονομικά μιας επιχείρησης ή ενός σπιτιού, ένα τιμολόγιο, προϋπολογισμός ενός έργου κ.λπ.

Τα σύγχρονα φύλλα υπολογισμών παρέχουν εκτεταμένες δυνατότητές αριθμητικών υπολογισμών, γραφικών και διαγραμμάτων και ανεπτυγμένα συστήματα μακροεντολών και τελευταία ενσωματωμένη γλώσσα προγραμματισμού π.χ. Visual Basic στο πακέτο Excel. Στα σημερινά φύλλα υπολογισμών έχει κανείς στη διάθεση του ισχυρές μεθόδους ανάλυσης, καθώς και μια βιβλιοθήκη από πολλές συναρτήσεις απλές αλλά και ιδιαίτερα πολύπλοκες. Τα φύλλα υπολογισμών συνδέονται εύκολα μεταξύ τους, ενώ περιλαμβάνουν πολλαπλές δυνατότητες μορφοποίησης του φύλλου και ανάπτυξης διαγραμμάτων.

6.7.2. Access, PowerPoint

Το Access είναι ένα πακέτο βάσης δεδομένων που επιτρέπει στο χρήστη του να αποθηκεύσει πληροφορίες με κάποια δομημένη μορφή, με τρόπο ώστε να μπορεί να τις ανάκτησή ή να τις χειριστεί (όλες ή ορισμένες) εύκολα και γρήγορα.

Η δουλειά με αυτό το πακέτο ξεκινάει κατά κανόνα με τον ορισμό της δομής του αρχείου ή των αρχείων τα οποία απαιτεί ο χρήστης. Όλο αυτό το πακέτο παρέχει κάποια ευκολία(λειτουργία) με την οποία γίνεται αυτό. Όταν δημιουργούμε ένα απλό αρχείο πελάτη. Έστω ότι έχει αποφασιστεί να χρειάζεται ένας 6-ψήφιος κωδικός πελάτη, ο οποίος θα είναι αλφαριθμητικού τύπου. Η γραμμογράφηση περιλαμβάνει επίσης το ονοματεπώνυμο του

6.7.3. Outlook, Frontpage

Το Outlook εντάσσεται στη κατηγορία των προγραμμάτων που συμβάλουν στην οργάνωση των διάφορων εργασιών ενός γραφείου. Με τα πακέτα αυτά ο χρήστης μπορεί να χειριστεί το ημερολόγιο του, δηλαδή να καταγράψει τα ραντεβού του για μια ή περισσότερες ημέρες και να καθορίζει υπενθυμίσεις γι' αυτά. Επίσης να διαχειρίζεται την ηλεκτρονική αλληλογραφία με συνεργάτες στην ίδια επιχείρηση για τον καθορισμό των συναντήσεων αποφεύγοντας τα κουραστικά και επαναλαμβανόμενα τηλέφωνα.

Συνήθως τα προγράμματα αυτά έχουν και δυνατότητα χρονικού προγραμματισμού, με την οποία μπορούν να επιλέγονται τα κενά χρόνου των συναδέλφων και να προγραμματίζονται οι συνεδριάσεις στην καταλληλότερη στιγμή.

Μπορούν να κρατούν τα προσωπικά αρχεία του χρήστη ή άλλα αρχεία πελατών, συνεργατών, προμηθευτών, διαφημιζομένων, προσωπικού στους οποίους είναι δυνατή και εύκολη η αποστολή και λήψη μηνυμάτων ηλεκτρονικού ταχυδρομείου.

Τέλος διάφορες άλλες λειτουργίες όπως διαχείριση σημειώσεων, χρονικά διάφορων ενεργειών, αναζήτηση διάφορων πληροφοριών, εορτολόγιο, τήρηση πρωτοκόλλου κ.α. μπορούν να συμπεριλαμβάνονται στις δυνατότητες των πακέτων αυτών.

Όλες οι προηγούμενες δυνατότητες εντάσσονται στις προσπάθειες των κατασκευαστών λογισμικού προς την πορεία για το 'γραφείο χωρίς χαρτιά'.

Frontpage

Έχει ήδη γίνει αναφορά στο ρόλο και τη σημασία του διαδικτύου. Σήμερα ο κύριος όγκος πληροφοριών που μπορούν να αναζητηθούν περιέχονται στο λεγόμενο Παγκόσμιο Ιστό. Οι πληροφορίες στον ιστό είναι διασπασμένες σε τόπους, που στην ουσία αποτελούν ηλεκτρονικές σελίδες. Υπάρχει λοιπόν σοβαρό κίνητρο για κάθε επιχείρηση να διαθέτει σελίδα στον ιστό, η οποία πρέπει να είναι ελκυστική και να ενημερώνεται τακτικά. Η δημιουργία ιστοσελίδων αποτελεί πλέον μια ακόμη (νέα) εργασία γραφείου, που έχει ως κύριο εργαλείο το πακέτο του Frontpage.

Για τη δημιουργία μιας ιστοσελίδας μπορεί να χρησιμοποιηθεί κάποια ειδική γλωσσά προγραμματισμού(π.χ. HTML, Java) ή κάποιο πακέτο.

Αυτά τα πακέτα αποτελούν ένα ολοκληρωμένο περιβάλλον ανάπτυξης και συντήρησης τόπων. Παρέχουν πολλές ευκολίες και εργαλεία για την ένθεση κειμένου, εικόνων, ήχου και βίντεο. Ο χρήστης μπορεί να δημιουργήσει εύκολα υπερσυνδέσεις που παραπέμπουν σε άλλες σελίδες ή τόπους. Η αποδοτική εργασία με τα πακέτα αυτά καθίσταται ευχερής μετά από μικρή εκπαίδευση και εξοικείωση.

6.8. OpenOffice

Το OpenOffice είναι μια σουίτα εφαρμογών γραφείου(εξέλιξη του Office της Microsoft). Περιλαμβάνει ελληνικό ορθογράφο και συλλαβισμό και προσφέρει συμβατότητα με το Microsoft Office. Αποτελεί μια πολύ καλή λύση για επιχειρήσεις, σχολεία και οργανισμούς, καθώς μπορούν να το αποκτήσουν δωρεάν. Το OpenOffice μπορεί να διαβάζει και να γράφει, μεταξύ των άλλων, αρχεία Word, Excel και PowerPoint. Το OpenOffice (έκδοση 2.0) περιλαμβάνει τα εξής προγράμματα:

1) Write (Επεξεργασίας κειμένου).
   Ανοίγει, αποθηκεύει και επεξεργάζεται αρχεία τύπου Word.
2) Calt (Λογιστικά φύλλα).
   Ανοίγει, αποθηκεύει και επεξεργάζεται αρχεία τύπου Excel.
3) Impress (Δημιουργία παρουσιάσεων).
   Ανοίγει, αποθηκεύει και επεξεργάζεται αρχεία τύπου PowerPoint
4) Base (Δημιουργία και επεξεργασία βάσεων δεδομένων).
5) Math (Επεξεργαστής μαθηματικών παραστάσεων).
6) Draw (Επεξεργασία γραφικών ,σχεδίων και διαγραμμάτων).

Αυτά τα κύρια μέρη, συνοδεύονται από διάφορα βοηθητικά όπως ο επεξεργαστής μαθηματικών παραστάσεων (*Math),* ο δημιουργός σχεδιαγραμμάτων (*Chart),* προγράμματα δηλαδή τα οποία θα μπορούσαν να είναι αυτόνομα, καθώς και μία ποικιλία Αυτόματων Πιλότων (*AutoPilots* και *Wizards*) που αυτοματοποιούν κάποιες ενέργειες και διαδικασίες, προσθέτοντας στη σουίτα μία πραγματικά εντυπωσιακή ευελιξία. Όλα αυτά συνοδεύονται από ένα πανίσχυρο προγραμματιστικό interface που δίνει τη δυνατότητα αλληλεπίδρασης με τοπρόγραμμα από διάφορες γλώσσες προγραμματισμού όπως Basic, Java, C, Perl κ.λ.π.

✓ Κοινά χαρακτηριστικά για όλη τη σουΐτα:

- Εγγενής XML τύπος αρχείων για αρχεία που παρότι έχουν όλες τις δυνατότητες μορφοποίησης, έχουν και πολύ μικρό μέγεθος.
- Δυνατότητα *εισαγωγής τύπων εγγράφων* και προτύπων από διάφορα άλλα προγράμματα.
- Δυνατότητα *εξαγωγής των εγγράφων* σε διάφορους τύπους αρχείων, από απλή HTML μέχρι και MS Office XP, Docbook, PDF, Flash (swf), LaTex κλπ.
- Λειτουργικότητα *εισαγωγής διευθυνσιολογίου* από το Mozilla ή το Netscape έκδοσης 6, το MS Outlook ή ακόμα και εξυπηρετητές LDAP.
- Εγγενής *υποστήριξη Unicode*, πάνω από 20 εκδόσεις σε τοπικές γλώσσες είναι διαθέσιμες.
- Εξαιρετικά ενισχυμένες *επιλογές εκτύπωσης*.

- Χαρακτηριστικά του επεξεργαστή κειμένου *(Writer)*:
- Απευθείας *σύνδεση με προγράμματα e-mail* της επιλογής σας για την απροβλημάτιστη αποστολή των κειμένων σας σε όλο τον κόσμο.
- *Στυλίστας* που επιτρέπει την αυτόματη και δυναμική αλλαγή της εμφάνισης του κειμένου σας.
- *Διαχείριση πεδίων* για την αυτοματοποιημένη αποστολή επιστολών σε διευθύνσεις που λαμβάνονται από μία προέλευση δεδομένων.
- *Σύστημα εκδόσεων (versions)* για καλή συνεργασία με τους συναδέλφους σας, και σύγκριση αλλαγών.
- Αυτόματη διόρθωση και αυτόματη συμπλήρωση λέξεων, που επιταχύνουν τη δακτυλογράφηση χωρίς να ενοχλούν.
- Δυνατότητα αρχειοθέτησης και λειτουργίες βιβλιογραφίας, που διευκολύνουν τη μετακίνηση μέσα στο κείμενο.
- «Έξυπνη» διαχείριση μορφοποίησης που σας επιτρέπει να απεικονίσετε οτιδήποτε θέλετε, εύκολα και ξεκάθαρα.
- Χαρακτηριστικά του υπολογιστικού φύλλου **(Calc)**:
- Δημιουργία μακροεντολών StarBASIC ή «έξυπνων» πρόσθετων λειτουργιών σε Java ή C/C++ χρησιμοποιώντας το προγραμματιστικό interface του OpenOffice.org.
- Εισαγωγή και χειρισμός πληροφοριών από εξωτερικές πηγές (όπως βάσεις δεδομένων ODBC και JDBC) ή άλλα προγράμματα του OpenOffice.org χρησιμοποιώντας το DataPilot.
- Φόρμουλες φυσικής γλώσσας (στα αγγλικά).
- Συνδυασμός των στοιχείων των κελιών από διάφορα υπολογιστικά φύλλα με το εργαλείο data consolidation.
- Διαχειριστής σεναρίων για να κάνετε αναλύσεις «αν - τι» (what-if-then).
- Εργαλεία δημιουργίας σχεδιαγραμμάτων για να απεικονίσετε δισδιάστατα ή τρισδιάστατα τα δεδομένα σας.
- Χαρακτηριστικά του τμήματος σχεδίασης (Draw):
- Το πρόγραμμα FontWorks, που σας επιτρέπει να τροποποιήσετε το κείμενο που έχετε χρησιμοποιήσει, σε διάφορα σχήματα για επιπλέον εφέ.
- Γρήγορη και εύκολη δημιουργία σύνθετων τρισδιάστατων σχημάτων.
- Καμπύλες Bezier για ρεαλιστικά ομαλές καμπύλες γραμμές.
- Σταγονόμετρο, για την αλλαγή του χρώματος οποιουδήποτε εικονοστοιχείου σε κάποιο άλλο.
- Εργαλεία και εφέ Raster (πλαισίων κουκίδων).
- Χαρακτηριστικά του τμήματος παρουσιάσεων (Impress):
- Χρησιμοποιεί όλα τα εφέ και γραφικά εργαλεία από το Draw.
- Αυτόματος Πιλότος (AutoPilot), που σας βοηθάει να απεικονίσετε τις ιδέες σας ακριβώς όπως τις θέλετε.
- Αυτόματη μορφοποίηση, που σας εξασφαλίζει ότι οι παρουσιάσεις σας θα δείχνουν επαγγελματικές.
- Ένας μεγάλος αριθμός από εφέ μετάβασης (slide effects), που προσθέτουν ένα επιπλέον ενδιαφέρον στην παρουσίασή σας.

6.8.1. Δημιουργία εγγράφων

Στο OpenOffice.org υπάρχουν τρεις τρόποι να δημιουργήσετε ένα νέο έγγραφο. Ο απλούστερος τρόπος είναι χρησιμοποιώντας τη συντόμευση

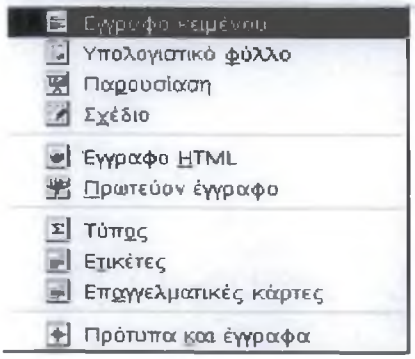
πληκτρολογίου (*Ctrl+N*), οπότε θα ανοίξει ένα ίδιο έγγραφο με αυτό στο οποίο εργαζόμαστε.

Αν για παράδειγμα εργαζόμαστε σε ένα έγγραφο κειμένου, θα ανοίξει ένα νέο έγγραφο κειμένου ενώ αν εργαζόμαστε σε ένα φύλλο υπολογισμού, θα ανοίξει ένα νέο φύλλο υπολογισμού.

Εναλλακτικά μπορούμε να πατήσουμε το εικονίδιο στη γραμμή εργαλείων. Ένα απλό πάτημα έχει το ίδιο αποτέλεσμα με τη συντόμευση πληκτρολογίου, ενώ ένα παρατεταμένο πάτημα εμφανίζει ένα υπομενού από το οποίο μπορούμε να επιλέξουμε τον τύπο του νέου εγγράφου, μεταξύ των τεσσάρων βασικών (έγγραφο κειμένου, υπολογιστικό φύλλο, σχέδιο, παρουσίαση), εγγράφου για το διαδίκτυο (HTML), πρωτεύοντος εγγράφου, μαθηματικού τύπου, αρχείου ετικετών ή επαγγελματικών καρτών, και τέλος δημιουργία εγγράφου με χρήση πρότυπου.

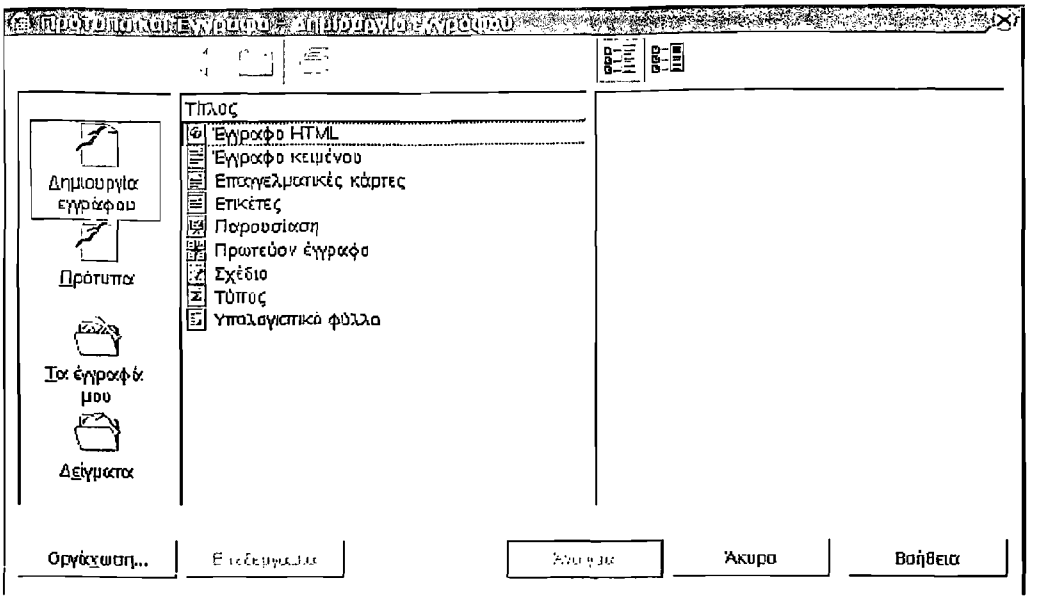
Το ίδιο υπομενού θα δούμε και αν ακολουθήσουμε τον τρίτο τρόπο, αν επιλέξουμε δηλαδή από το μενού *Αρχείο (File)* την επιλογή *Δημιουργία (New)*.

Εικόνα 1: Μενού δημιουργίας νέου εγγράφου



α) Δημιουργία εγγράφων από πρότυπο

Αν επιλέξουμε να δημιουργήσουμε ένα έγγραφο με χρήση προτύπου, θα ανοίξει ένας διάλογος ο οποίος θα μας μεταφέρει στον προκαθορισμένο κατάλογο προτύπων του υπολογιστή μας.



Από εκεί, μπορούμε να δούμε μία προεπισκόπηση του προτύπου ή πληροφορίες για το πρότυπο, επιλέγοντας έτσι το πρότυπο το οποίο μας ταιριάζει καλύτερα.

Μόλις επιλέξουμε το πρότυπο της αρεσκείας μας, ένα αντίγραφό του ανοίγει στον υπολογιστή μας και μας επιτρέπει να εργαστούμε σε αυτό το αρχείο.

	Σε μία εταιρεία, ο διαχειριστής θα δημιουργήσει ένα πρότυπο, το οποίο θα μπορεί ο οποιοσδήποτε χρήστης να χρησιμοποιήσει για να δημιουργήσει μία εταιρική επιστολή. Κατ' αυτόν τον τρόπο, όλες οι επιστολές της εταιρείας θα έχουν κοινή μορφή.

β) Άνοιγμα εγγράφων

Για να ανοίξουμε ένα έγγραφο που έχουμε ήδη στο σύστημά μας, μπορούμε να χρησιμοποιήσουμε τη συντόμευση πληκτρολογίου *(Ctrl+O)*, το εικονίδιο της γραμμής εργαλείων  ή από το μενού *Αρχείο (File)*, την επιλογή *Άνοιγμα (Open)*.

Σε κάθε περίπτωση, ένας νέος διάλογος θα εμφανιστεί ο οποίος θα μας μεταφέρει στον προκαθορισμένο κατάλογο αρχείων του υπολογιστή μας όπως και στην περίπτωση της δημιουργίας εγγράφων. Στο διάλογο αυτό έχουμε την επιλογή να περιορίσουμε τα αρχεία που θα εμφανίζονται, με βάση τον τύπο τους, χρησιμοποιώντας τη λίστα επιλογής *Τύπος Αρχείου (File Type)*. Η προεπιλεγμένη ρύθμιση για τον τύπο αρχείου εμφανίζει όλα τα αρχεία που βρίσκονται στον κατάλογο αυτό.

γ) Αποθήκευση εγγράφων

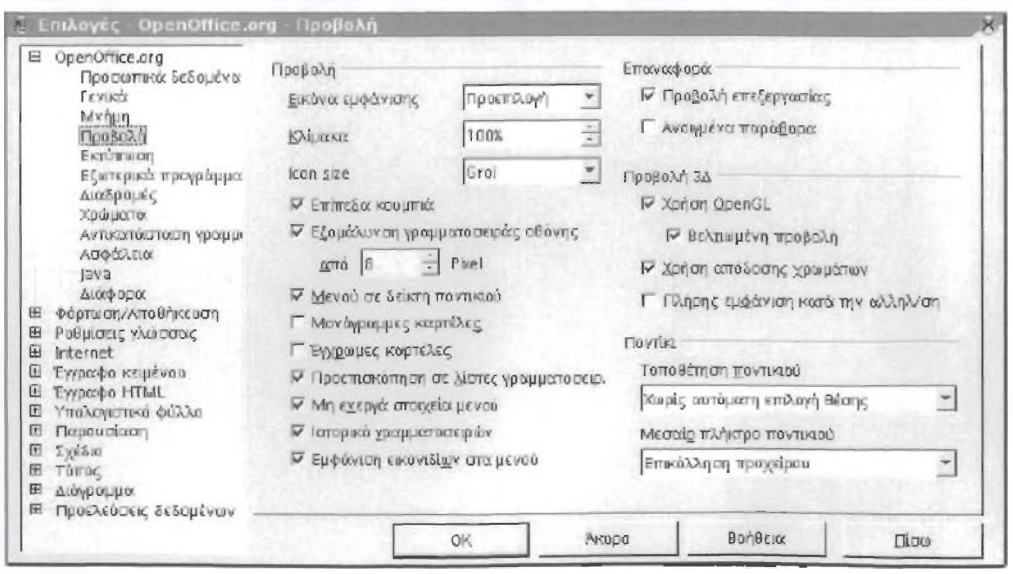
Η αποθήκευση ενός εγγράφου μπορεί να γίνει είτε με τη συντόμευση πληκτρολογίου *(Ctrl+S)*, το εικονίδιο της γραμμής εργαλείων  ή από το μενού *Αρχείο (File)*, την επιλογή *Αποθήκευση (Save)*.

Αν το αρχείο αυτό έχει ήδη αποθηκευθεί μία φορά (ή έχει ανοιχθεί, οπότε έχει ήδη όνομα) τότε αποθηκεύεται, και δεν εμφανίζεται κανένα μήνυμα. Αν το αρχείο δεν έχει αποθηκευθεί ποτέ αλλά είναι νέο αρχείο, τότε η επιλογή *Αποθήκευση (Save)* στο μενού *Αρχείο (File)* δεν είναι διαθέσιμη. Σε αυτή την περίπτωση, τόσο η συντόμευση πληκτρολογίου όσο και το πάτημα του εικονιδίου της γραμμής εργαλείων έχουν το ίδιο αποτέλεσμα με την επιλογή *Αποθήκευση Ως (Save As)* του μενού *Αρχείο (File)*. Η επιλογή αυτή ανοίγει ένα διάλογο στον οποίο ζητείται ένα όνομα (και τύπος κατά περίπτωση) για το αρχείο που θα αποθηκευτεί. Ο προεπιλεγμένος τύπος του αρχείου που θα προταθεί κατά περίπτωση, ορίζεται από τις *Επιλογές (Options)* του προγράμματος.

δ) Επιλογές

Όπως αναφέραμε νωρίτερα, το OpenOffice.org δεν είναι απλά μία συλλογή προγραμμάτων, αλλά είναι μία ολοκληρωμένη σουίτα και έχει έτσι όλα τα πλεονεκτήματα της σουίτας. Ένα σημαντικό τέτοιο πλεονέκτημα είναι οι επιλογές, οι οποίες δεν χρειάζεται να οριστούν για κάθε πρόγραμμα ξεχωριστά αλλά μπορούν να οριστούν σε οποιοδήποτε πρόγραμμα και αυτόματα να έχουν ισχύ και για όλα τα άλλα προγράμματα (όπου αυτό έχει νόημα). Τον διάλογο των επιλογών των προσπελαύνουμε από το μενού *Εργαλεία* (*Tools*) και την επιλογή *Επιλογές* (*Options*).

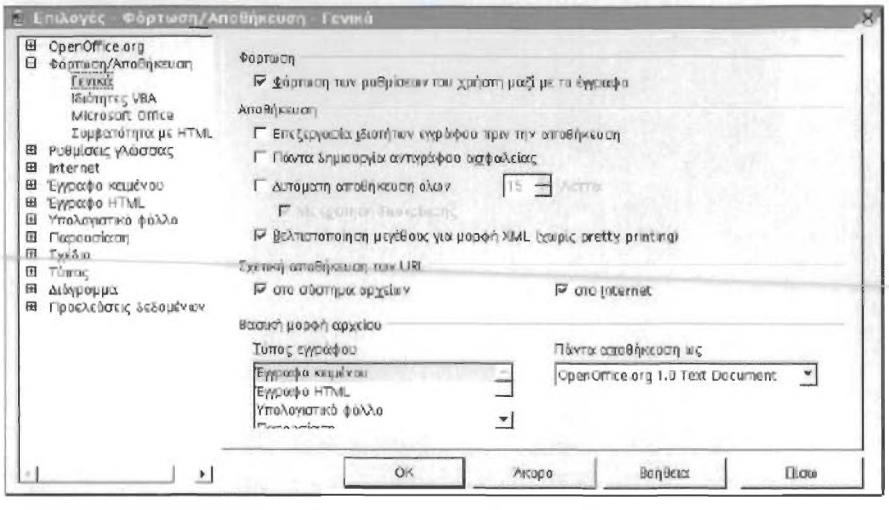
Εικόνα 3: Επιλογές του προγράμματος OpenOffice.org



ε) Άνοιγμα / Αποθήκευση

Από το δέντρο επιλογών στο αριστερό τμήμα του διαλόγου, επιλέγουμε το *Φόρτωση / Αποθήκευση* (*Load / Save*) και στη συνέχεια το τμήμα *Γενικά*(*General*). Στο κάτω μέρος του διαλόγου, μπορούμε να επιλέξουμε τον προεπιλεγμένο τύπο αποθήκευσης των αρχείων για κάθε κύρια υπομονάδα του OpenOffice.org.

Εικόνα 4: Επιλογές αποθήκευσης του προγράμματος OpenOffice.org

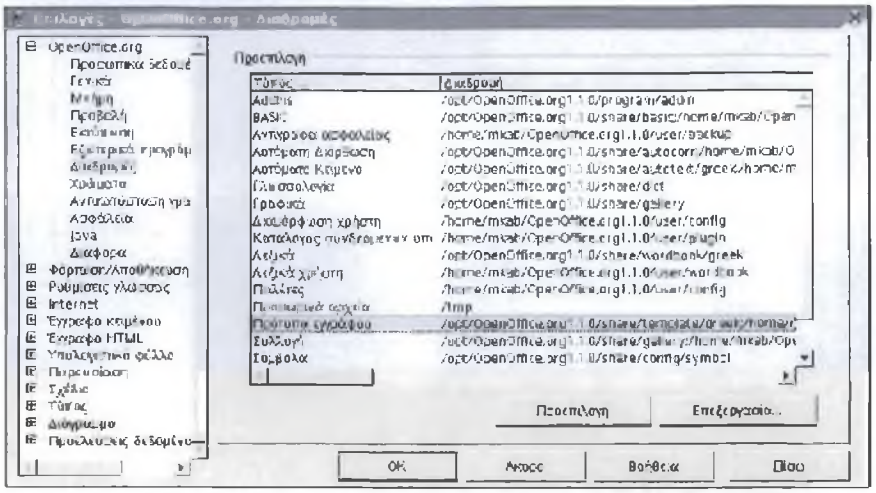


Σε μία εταιρεία που κάνει μετάβαση από MS Office σε OpenOffice.org και μεταφέρει σταδιακά τα συστήματά της, θα μπορούσαν οι χρήστες του OpenOffice.org να ορίσουν ως προεπιλεγμένο τύπο αποθήκευσης υπολογιστικών φύλλων αυτόν του MS Excel, για να υπάρχει διάφανη συνεργασία μεταξύ των διαφορετικών προγραμμάτων οργάνωσης γραφείου. Όταν μεταφερθούν όλα τα συστήματα σε OpenOffice.org, μπορεί να χρησιμοποιηθεί ο Αυτόματος Μετατροπέας Εγγράφων για να μετατραπούν τα αρχεία από την ιδιωτική και δύσχρηστη μορφή MS Excel στην ευέλικτη και ανοιχτή μορφή XML του OpenOffice.org. Το ίδιο και για τα αρχεία του MS Word και του MS PowerPoint.

στ) Διαδρομές Αρχείων

Από το δέντρο επιλογών στο αριστερό τμήμα του διαλόγου, επιλέγουμε το OpenOffice.org και από τις επί μέρους επιλογές το Διαδρομές (Paths). Σε αυτό μπορούμε να ορίσουμε τις προκαθορισμένες διαδρομές στο σύστημά μας, στις οποίες μπορούν να βρεθούν χρήσιμα στοιχεία. Για να καθορίσουμε το που θα βρίσκονται τα πρότυπα των εγγράφων μας, από το δεξί τμήμα του διαλόγου επιλέγουμε το Πρότυπα Εγγράφου (Templates) και πατάμε το κουμπί Επεξεργασία (Edit). Στο διάλογο που θα ανοίξει, ορίζουμε τη διαδρομή των προτύπων μας και όταν ξαναζητήσουμε τη δημιουργία εγγράφου από πρότυπο θα ανοίξει
αυτόματα ο διάλογος επιλογής προτύπων σε αυτή τη διαδρομή.

Εικόνα 5: Προεπιλεγμένες διαδρομές αρχείων



Σε μία εταιρεία με κεντρικό σύστημα εξυπηρέτησης αρχείων (file server) μπορούν να δημιουργηθούν κατάλογοι στο σύστημα αυτό που να έχουν όλοι οι χρήστες δικαιώματα ανάγνωσης, και να περιέχουν τα πρότυπα της εταιρείας και μία συλλογή εικόνων (cliparts). Στη συνέχεια, ορίζοντας τις προκαθορισμένες διαδρομές για τα πρότυπα και τις συλλογές εικόνων, μπορούμε να είμαστε σίγουροι ότι όλοι οι υπάλληλοι της εταιρείας θα έχουν πρόσβαση στα ίδια πρότυπα και στις ίδιες εικόνες.

ζ) Αυτοματισμοί

Το OpenOffice.org παρέχει στον τελικό χρήστη αρκετούς αυτοματισμούς γιασυνηθισμένες εργασίες, όπως είναι η *δημιουργία εξωφύλλων Fax, Επαγγελματικών καρτών, Φόρμας αλληλεπίδρασης με βάσεις δεδομένων, ετικετών, εγκυκλίων επιστολών κλπ.* Οι περισσότεροι από αυτούς τους αυτοματισμούς είναι άμεσα προσβάσιμοι από το μενού Αρχείο (File) στην επιλογή Αυτόματος Πιλότος (AutoPilot).

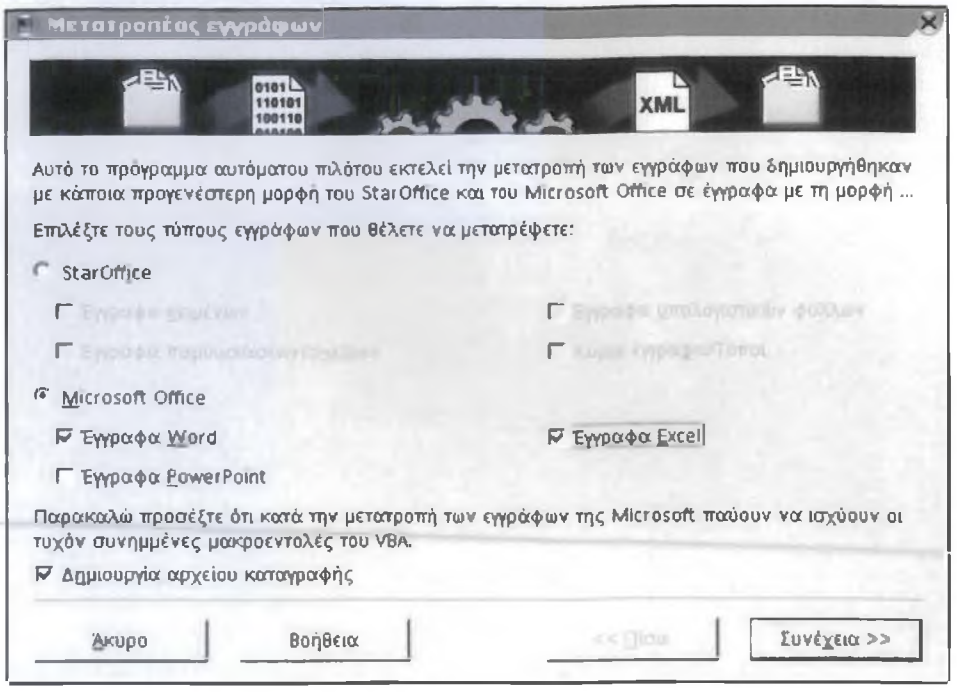
Για ορισμένους αυτοματισμούς πρέπει να έχει ήδη οριστεί μία προέλευση δεδομένων (database).

η) Μετατροπέας εγγράφων

Ένας από τους αυτοματισμούς που παρέχει το OpenOffice.org είναι η αυτόματη μετατροπή εγγράφων και προτύπων του MS Office σε μορφή OpenOffice.org.

Τα αρχεία (και πρότυπα) του MS Word μετατρέπονται σε αρχεία (και πρότυπα) του Writer, αυτά του MS Excel μετατρέπονται σε μορφή Calc ενώ τέλος οι παρουσιάσεις του MS PowerPoint μετατρέπονται σε παρουσιάσεις Impress.

Εικόνα 5: Αυτόματος μετατροπέας εγγράφων MS Office σε OpenOffice.org



Μπορούμε να ξεκινήσουμε την αυτόματη μετατροπή εγγράφων από το μενού Αρχείο (File), επιλογή Αυτόματος Πιλότος (AutoPilot), υποεπιλογή Μετατροπέας Εγγράφων (Document Converter). Ο Αυτοπιλότος έχει τρία απλά στάδια:

i. Επιλέγουμε τον τύπο των αρχείων που θέλουμε να μετατρέψουμε (Star Office ή Microsoft Office) καθώς και όλες τις επιμέρους μορφές εγγράφων (Έγγραφα Word, έγγραφα PowerPoint, έγγραφα Excel), αν βέβαια έχουμε και θέλουμε να μετατρέψουμε αντίστοιχα έγγραφα. Αν επιθυμούμε να καταγραφούν οι ενέργειες που γίνονται για να τις ελέγξουμε μετά, μπορούμε να επιλέξουμε και το πεδίο Δημιουργία Αρχείου Καταγραφής (Create Log File). Πατάμε το *Συνέχεια (Next)*.

ii. Χρησιμοποιούμε τα πλαίσια ελέγχου στην πρώτη περιοχή που λέγεται Πρότυπα (Templates) και στη δεύτερη περιοχή, που λέγεται Έγγραφα (Documents) ανάλογα με τα αρχεία που έχουμε να μετατρέψουμε. Στο πεδίο Εισαγωγή από (Import from) γράφουμε τον αρχικό κατάλογο στον οποίο βρίσκονται τα πρότυπα και αρχεία που θέλουμε να μετατραπούν. Αν αυτός περιέχει υποκαταλόγους, επιλέγουμε και το πλαίσιο Συμπεριλαμβ. υποκαταλόγους (Including Subdirectories). Στο πεδίο Αποθήκευση σε (Save To) γράφουμε τον κατάλογο στον οποίο θα δημιουργηθούν τα αρχεία της μορφής OpenOffice.org. Ο κατάλογος αυτός πρέπει να υπάρχει ήδη και είναι καλό να είναι κενός για να μη ρωτά συνεχώς το OpenOffice.org για αντικατάσταση τυχόν υπαρχόντων αρχείων. Δεν είναι ανάγκη να δημιουργήσετε
υποκαταλόγους, γιατί δημιουργούνται αυτόματα αν χρειάζονται. Πατάμε το

Συνέχεια (Next). Επαναλαμβάνουμε την ίδια διαδικασία για τα έγγραφα κάθε τύπου αρχείου που έχουμε επιλέξει να μετατρέψουμε (π.χ. MS Excel) αν απαιτείται. Πατάμε το *Συνέχεια (Next)*.
iii. Το OpenOffice.org εμφανίζει μία περίληψη των επιλογών μας. Ένας σύντομος έλεγχος είναι αρκετός για να διαπιστωθεί αν υπάρχει κάποιο λάθος. Επιβεβαιώνουμε με το *Μετατροπή (Convert)*.

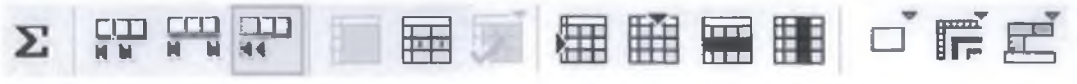
6.8.2. Επεξεργασία κειμένου

Το πρόγραμμα επεξεργασίας κειμένου (Writer) του OpenOffice.org δεν είναι απλά ένας κειμενογράφος, μία εξελιγμένη γραφομηχανή, αλλά ένα ισχυρό εργαλείο που μπορεί να σας βοηθήσει να γράψετε ό,τι επιθυμείτε, από μία απλή υποσημείωση μέχρι μία ολοκληρωμένη διατριβή, ένα επιστημονικό σύγγραμα ή ένα διαφημιστικό φυλλάδιο. Οι δυνατότητές του είναι πρακτικά απεριόριστες, ενώ έχει δοθεί ιδιαίτερη σημασία στην ευκολία χρήσης και την αυτοματοποίηση συχνών εργασιών. Τα έγγραφά σας υποστηρίζονται από αυτόματο ορθογραφικό έλεγχο και συλλαβισμό, ενώ μπορείτε να εισάγετε σε κάθε έγγραφο αντικείμενα τα οποία είναι αποθηκευμένα στο δίσκο σας, ή μπορεί να δημιουργούνται από άλλα τμήματα του OpenOffice.org, ένα υπολογιστικό φύλλο του Calc για παράδειγμα ή ένα σχέδιο που σχεδιάσατε στο Draw.

1) Γραμμές εργαλείων

Αν και εμείς θα αναλύσουμε τους διαλόγους και τις μεθόδους που χρησιμοποιούνται για να γίνουν διάφορες εργασίες στο Writer, το ίδιο το Writer έχει διάφορες γραμμές εργαλείων οι οποίες παρέχουν κουμπιά για εύκολη και γρήγορη πρόσβαση σε συνηθισμένες λειτουργίες. Οι περισσότερες, αν όχι όλες, οι εργασίες και μορφοποιήσεις που αναφέρουμε σε αυτόν τον οδηγό, μπορούν να γίνουν από αυτές τις γραμμές εργαλείων, με ένα γρήγορο και λίγο – πολύ αυτονόητο τρόπο. Δύο από αυτές, η **Γραμμή Εργαλείων (Main Toolbar)** στην αριστερή πλευρά της οθόνης και η **Γραμμή Λειτουργιών (Function Bar)** στην πάνω πλευρά, είναι πάντα ορατές (εκτός αν επιλέξει ο χρήστης να τις κρύψει). Μία τουλάχιστον γραμμή εργαλείων, η **Γραμμή Αντικειμένων (Object Bar)** εμφανίζεται ακόμα, η οποία όμως εξαρτάται από τη θέση του δείκτη. Αν ο δείκτης βρίσκεται σε κάποιο κείμενο, πάνω από ένα χαρακτήρα για παράδειγμα, τότε είναι ορατή η **Γραμμή Αντικειμένων Κειμένου (Text Object Bar)**. Αν όμως ο δείκτης βρίσκεται στο κελί κάποιου πίνακα, τότε είναι ορατή η **Γραμμή Αντικειμένων Πίνακα (Table Object Bar)**. Ωστόσο, το κελί του πίνακα μπορεί να περιέχει κείμενο, οπότε έχει νόημα και η **Γραμμή Αντικειμένων Κειμένου**. Ο Writer φροντίζει να προσθέσει όλες τις γραμμές αντικειμένων που μπορεί να έχουν νόημα για το συγκεκριμένο περιεχόμενο, και ο χρήστης μπορεί να τις ανακαλέσει, απότο κουμπί  στην αριστερή πλευρά της εκάστοτε ενεργής γραμμής αντικειμένων. Όποτε βλέπετε αυτό το κουμπί στη γραμμή αντικειμένων, θα γνωρίζετε ότι υπάρχουν «κρυμμένες» και άλλες γραμμές αντικειμένων που μπορούν να χρησιμοποιηθούν για τη μορφοποίηση του εγγράφου σας.

Εικόνα 1: Τμήμα της γραμμής αντικειμένων πίνακα



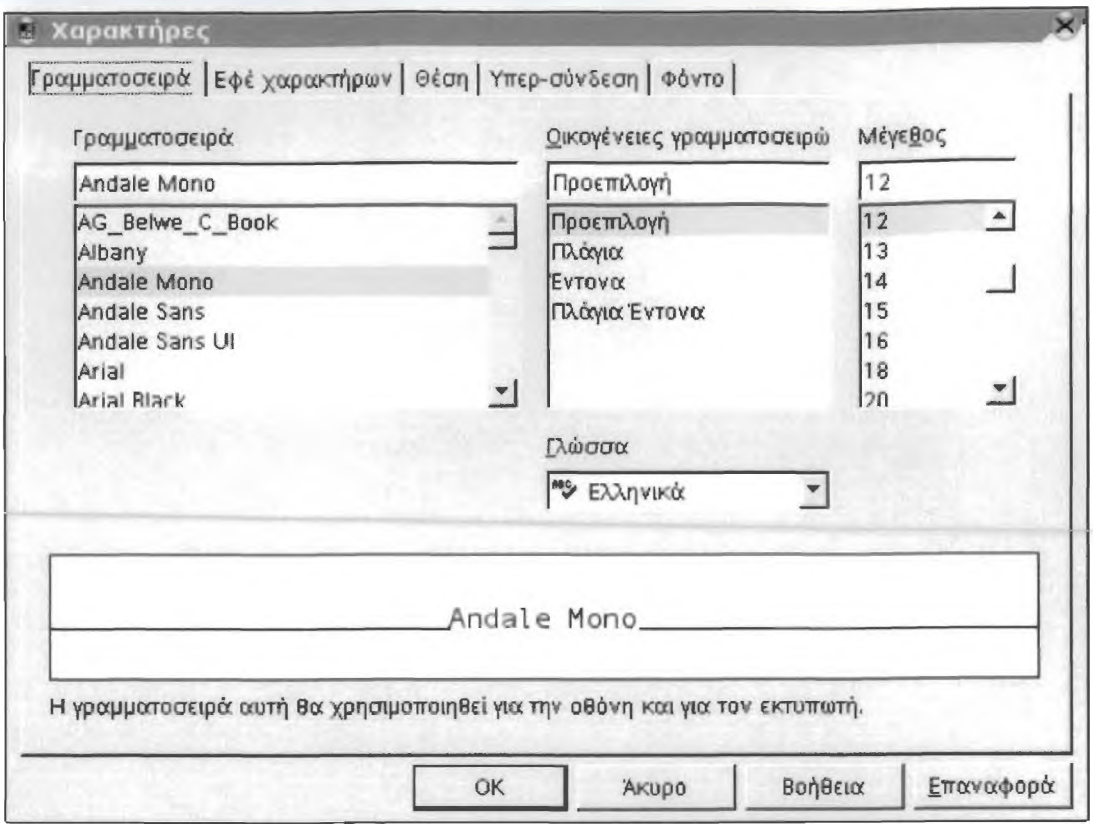
Η μορφοποίηση στο OpenOffice.org μπορεί να γίνει σε διάφορα επίπεδα, όπως χαρακτήρες, παράγραφοι, σελίδες, πίνακες κ.λ.π. Η μία μορφοποίηση δεν αναιρεί την άλλη, ακόμα και αν εφαρμόζονται διαδοχικά, και αν πρόκειται να χρησιμοποιηθεί αυτή η μέθοδος για τη μορφοποίηση, προτείνεται να εφαρμόζεται από τη γενικότερη μορφοποίηση (σελίδα) προς την ειδικότερη (χαρακτήρας).
Ωστόσο, η καλύτερη λύση είναι η μορφοποίηση με τα **Στυλ (Styles)**, την οποία θα αναλύσουμε στο τέλος του κεφαλαίου.

2) Μορφοποίηση χαρακτήρων

Το κείμενο που γράφει κανείς στο OpenOffice.org μπορεί να διαμορφωθεί για να ταιριάξει στο προσωπικό γούστο του τελικού χρήστη.

Για να μορφοποιήσουμε κάποιους χαρακτήρες που έχουμε ήδη γράψει, πρέπει να τους επιλέξουμε με το δείκτη και να πατήσουμε το δεξί κουμπί του ποντικιού. Από το μενού που εμφανίζεται, το οποίο ονομάζεται *μενού περιεχομένου (contentmenu)*, επιλέγουμε το **Χαρακτήρας (Character)** οπότε και εμφανίζεται ένας διάλογος με πέντε καρτέλες. Εναλλακτικά, μπορούμε να καλέσουμε τον ίδιο διάλογο, επιλέγοντας από το μενού **Μορφή (Format)** την επιλογή **Χαρακτήρες (Character)**.

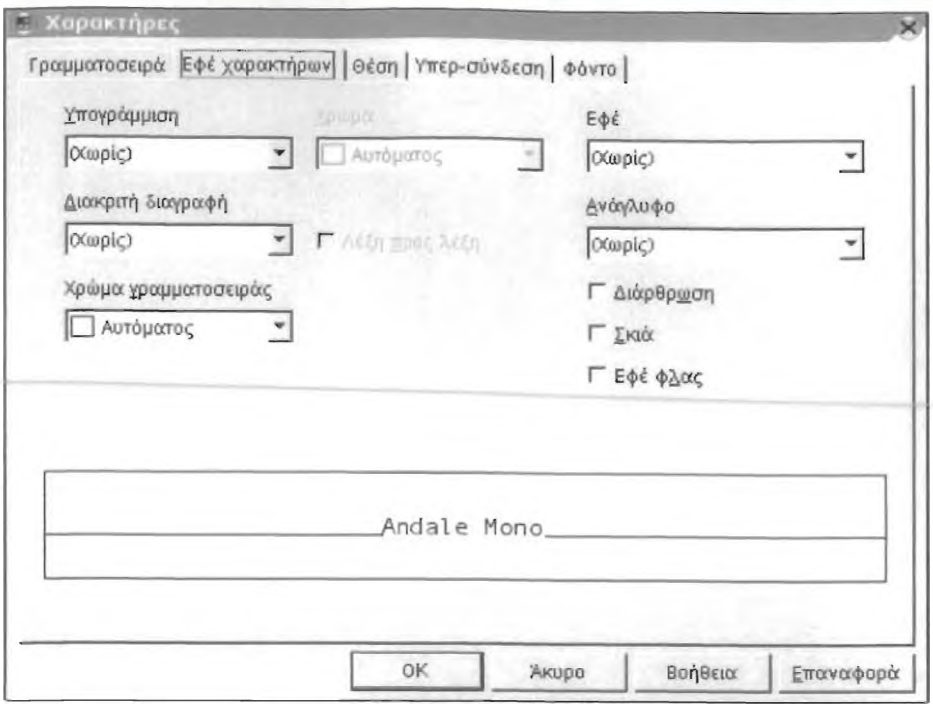
Εικόνα 8: Μορφοποίηση χαρακτήρων - Γραμματοσειρά



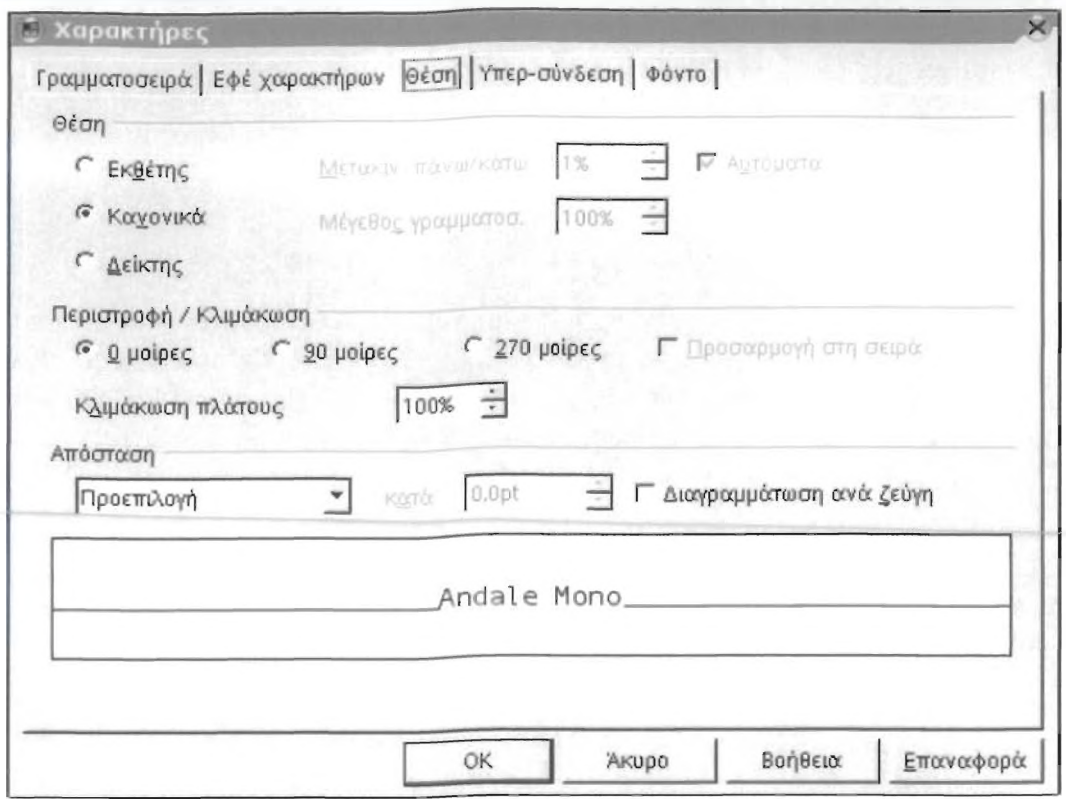
Από την καρτέλα **Γραμματοσειρά (Font)** μπορούμε να επιλέξουμε τη γραμματοσειρά στην οποία θα εμφανίζεται το κείμενό μας, την **Οικογένεια Γραμματοσειρών (Typeface)** της (κανονικά, πλάγια, έντονα...) και το **Μέγεθος (Size)** των γραμμάτων, ενώ παράλληλα βλέπουμε και μία προεπισκόπηση των επιλεγμένων ρυθμίσεων. Καλό είναι να επιλέξουμε και τη **Γλώσσα (Language)** στην οποία είναι γραμμένοι αυτοί οι χαρακτήρες, για να εφαρμοστούν τα αντίστοιχα βοηθήματα γλωσσολογίας (συλλαβισμός, ορθογραφικός έλεγχος) που μπορεί να είναι ενεργοποιημένα, ή να ενεργοποιηθούν αργότερα στο έγγραφό μας. Στις γλώσσες για τις οποίες έχει εγκατασταθεί ο ορθογραφικός έλεγχος, εμφανίζεται μπροστά από το όνομά τους ένα σύμβολο που υποδηλώνει αυτήν ακριβώς τη δυνατότητα.

Όλες οι ρυθμίσεις της καρτέλας Γραμματοσειρά (Font) εκτός από την επιλογή γλώσσας, είναι διαθέσιμες στη Γραμμή Αντικειμένων Κειμένου. Έτσι, για απλές και γρήγορες αλλαγές, δεν χρειάζεται να χρησιμοποιούμε το διάλογο για τη Μορφοποίηση χαρακτήρων.

Επιλογές Μορφοποίησης χαρακτήρων – Εφέ χαρακτήρων



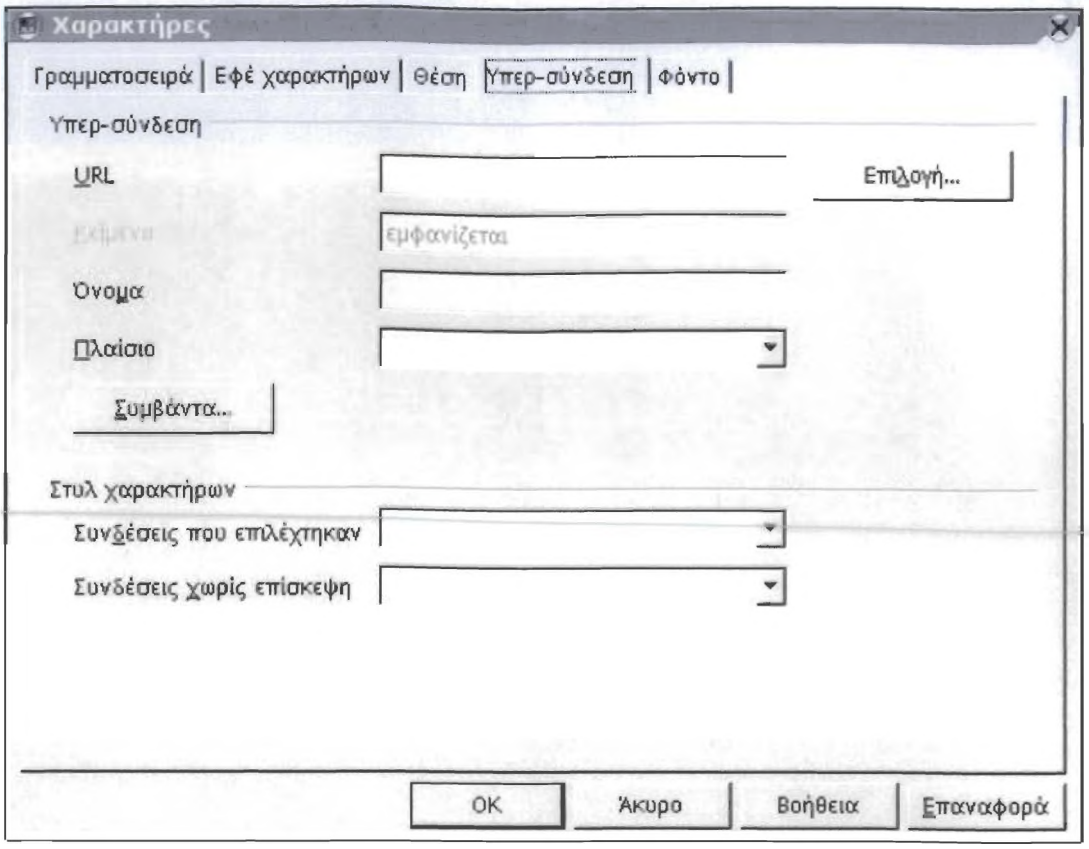
Από την καρτέλα **Εφέ χαρακτήρων (Font Effects)** μπορούμε να επιλέξουμε αν θέλουμε οι επιλεγμένοι χαρακτήρες να έχουν **Υπογράμμιση (underline)**. Η υπογράμμιση μπορεί να είναι σε διάφορες παραλλαγές όπως απλή (μία μόνο γραμμή), διπλή (δύο γραμμές), διάστικτη (κουκκίδες), διάφοροι συνδυασμοί κουκίδων και παυλών κ.ο.κ. Μπορούμε επίσης να ορίσουμε σε τι χρώμα θα είναι η υπογράμμιση, μία ρύθμιση που δεν πρέπει να μπερδεύεται με το χρώμα των χαρακτήρων. Οι δύο αυτές ρυθμίσεις είναι ανεξάρτητες μεταξύ τους. Υπάρχει επίσης η δυνατότητα για **διακριτή διαγραφή (strikethrough)** με διάφορους
τρόπους και χρώματα. Σε αυτή την καρτέλα ορίζουμε επίσης το χρώμα των γραμμάτων, καθώς και άλλα εφέ που χρησιμοποιούνται σπανιότερα όπως αν θα έχουν **σκιά (shadow)** οι χαρακτήρες, αν θα φαίνεται μόνο το **περίγραμμά** τους **(outline)**, αν θα είναι όλοι **κεφαλαίοι (capitals)** ή **μικρά κεφαλαία (καπιταλάκια, small caps)** άσχετα με το πως τους γράφουμε. Στον ίδιο διάλογο ορίζουμε και το **Αποτύπωμα (Relief)** των χαρακτήρων, αν θα είναι δηλαδή **Ανάγλυφοι(Embossed)** ή **Χαραγμένοι (Engraved)**.



Στην καρτέλα **Θέση (Position)** ορίζονται επιλογές που έχουν να κάνουν με τη θέση των χαρακτήρων. Αν δηλαδή θέλουμε οι χαρακτήρες να τοποθετηθούν ως **εκθέτες (superscript)** ή **δείκτες (subscript)**. Σε αυτή την περίπτωση μπορούμε να επιλέξουμε και τι απόσταση θα έχουν από τη **βασική γραμμή (baseline)** καθώς και τι ποσοστό του κανονικού μεγέθους θα έχουν οι χαρακτήρες αυτοί.

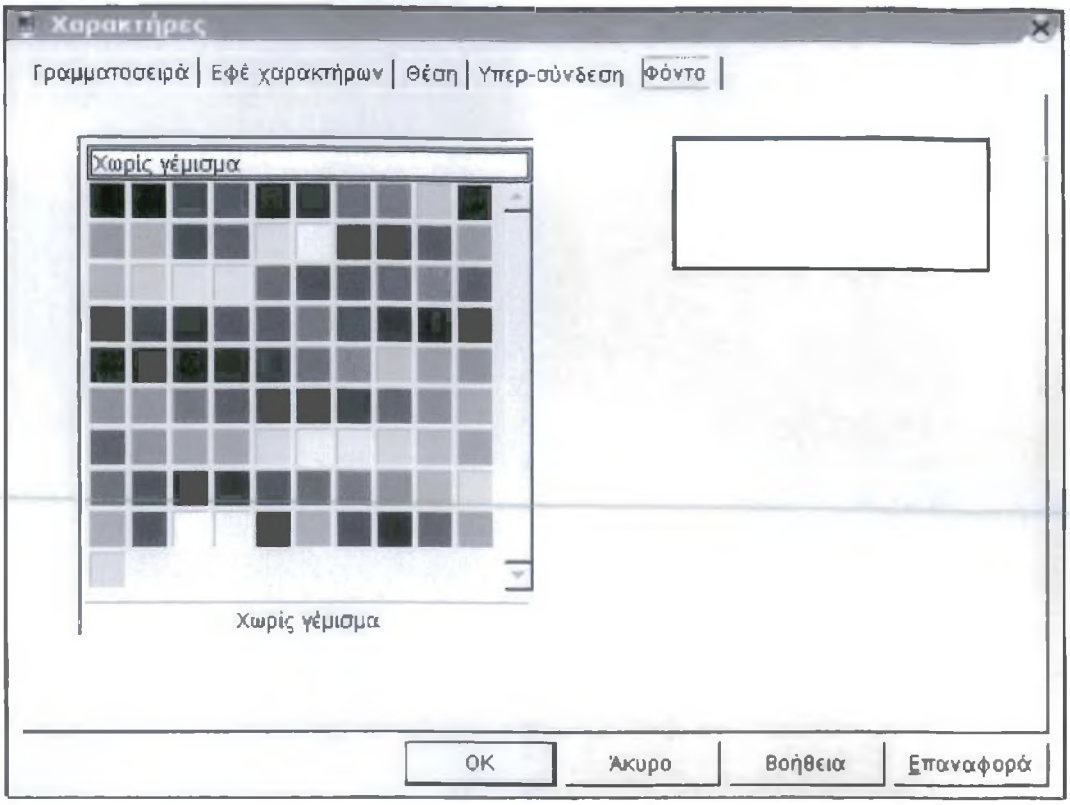
Εδώ ορίζεται επίσης τυχόν **περιστροφή (rotation)** των χαρακτήρων σε διάφορες γωνίες καθώς και η **Απόσταση (spacing)** μεταξύ των χαρακτήρων, ενώ υπάρχει και η δυνατότητα της **Διαγραμμάτωσης ανά ζεύγη (Kerning Pairs)** που μεταβάλλει την απόσταση μεταξύ των γραμμάτων, ανάλογα με τα γράμματα τα ίδια (άλλη απόσταση έχει για παράδειγμα ο χαρακτήρας «ι» από τον προηγούμενο,όταν ακολουθεί τον χαρακτήρα «Γ» και άλλη όταν ακολουθεί τον χαρακτήρα «Α»).

Εικόνα 11: Μορφοποίηση χαρακτήρων – Υπερ-σύνδεση



Η καρτέλα **Υπερ-σύνδεση (Hyperlink)** δίνει τη δυνατότητα να ορίσουμε ένα αρχείο στο διαδίκτυο (Internet) ή στο τοπικό μας δίκτυο το οποίο θα καλείται με το αντίστοιχο πρόγραμμα όταν πατάμε με το ποντίκι πάνω στο επιλεγμένο κείμενο. Ο Υπερ-σύνδεσμος αυτός μπορεί να καλείται σε κάποιο από τα ήδη ορισμένα και δημιουργημένα πλαίσια στο έγγραφό μας, ή σε ένα νέο. Αν και η έννοια του Υπερ-συνδέσμου προέρχεται από το διαδίκτυο, και η χρήση του είναι κατανοητή σε αρχεία διαδικτύου (HTML σελίδες), τίποτα δεν σας απαγορεύει να τη χρησιμοποιήσετε σε ένα κανονικό έγγραφο, καλώντας άλλα έγγραφα, όπως θα δούμε και στο παράδειγμα που ακολουθεί.

Εικόνα 12: Μορφοποίηση χαρακτήρων - Φόντο



Τέλος, από την καρτέλα με όνομα **Φόντο (Background)** μπορούμε να επιλέξουμε αν οι επιλεγμένοι χαρακτήρες θα έχουν κάποιο χρώμα ως φόντο, ή κανένα.

	Το Φόντο (Background) των γραμμάτων μπορεί να τροποποιηθεί εύκολα και από τη Γραμμή Αντικειμένων Κειμένου με το κατάλληλο κουμπί.

Το ανθρώπινο σώμα αποτελείται κυρίως από νερό (H_2O)

- Αρχικά γράφουμε το κείμενό μας κανονικά χωρίς καμία μορφοποίηση.
- Στη συνέχεια επιλέγουμε το χαρακτήρα «2», πατάμε το δεξί κουμπί του ποντικιού, επιλέγουμε *Χαρακτήρας (Character)* και πηγαίνουμε στην καρτέλα *Θέση(Position)*. Εκεί επιλέγουμε το *Δείκτης (Subscript)* και ρυθμίζουμε, αν θέλουμε,το πόσο μικρό θα είναι το γράμμα αναλογικά με τα υπόλοιπα, καθώς και το πόσο θα είναι χαμηλωμένο κάτω από τη βασική γραμμή (οι προεπιλεγμένες ρυθμίσεις είναι και οι συνηθισμένες για έντυπα). Αφού πατήσουμε το κουμπί Ok θα δούμε ότι ο χαρακτήρας 2 φαίνεται πια σαν δείκτης στο χαρακτήρα Η.
- Επιλέγουμε όλο τον τύπο, δηλαδή τους τρεις χαρακτήρες: Η, 2 και Ο. Πατάμε το δεξί κουμπί του ποντικιού, επιλέγουμε *Χαρακτήρας (Character)* και πηγαίνουμε στην καρτέλα *Υπερσύνδεση (Hyperlink)*. Στο πεδίο *URL* γράφουμε την τοποθεσία του αντικειμένου που θέλουμε να συνδέσουμε, κατά τη σύμβαση των δικτυακών συνδέσεων (π.χ. Για ένα αρχείο που βρίσκεται στο σύστημά μας και στον αρχικό κατάλογο με όνομα finalfiles, και ονομάζεται

waterspecs.sxw, η πλήρης ονομασία κατά τη σύμβαση δικτυακών συνδέσεων θα είναι: file:///finalfiles/waterspecs.sxw). Πατώντας το Ok κλείνουμε το διάλογο.

-. Τέλος, με όλο το κείμενο επιλεγμένο, καλούμε πάλι το διάλογο για τη Μορφοποίηση χαρακτήρων και από την καρτέλα *Γραμματοσειρά (Font)* αλλάζουμε τον *Τύπο (Format)* σε *Έντονο (Bold)*.

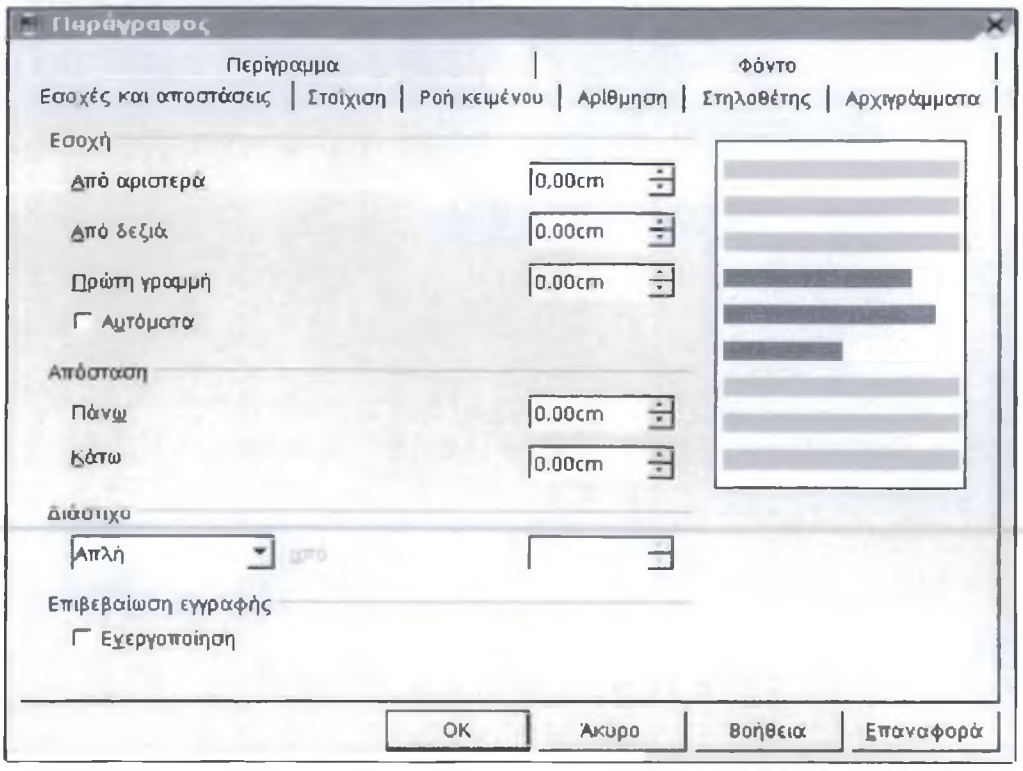
Οποιαδήποτε στιγμή τώρα πατήσουμε με το ποντίκι πάνω στο κείμενο H2O, θα ανοίξει, σε νέο παράθυρο, το αρχείο finalfiles/waterspecs.sxw από το δίσκο
μας.

3) Μορφοποίηση παραγράφων

Η μορφοποίηση του κειμένου μπορεί, και μερικές φορές πρέπει να γίνει, και σε επίπεδο παραγράφων. Όπως αναφέραμε και νωρίτερα, η καλύτερη τακτική, τόσο για την εμφάνιση όσο και για την έννοια του κειμένου, είναι να εφαρμόζεται πρώτα η μορφοποίηση της παραγράφου σε κάθε παράγραφο και στη συνέχεια αν χρειάζεται, η μορφοποίηση χαρακτήρων για να τονίσει κάτι. Οι δυνατότητες μορφοποίησης των παραγράφων είναι και αυτές πάρα πολλές.

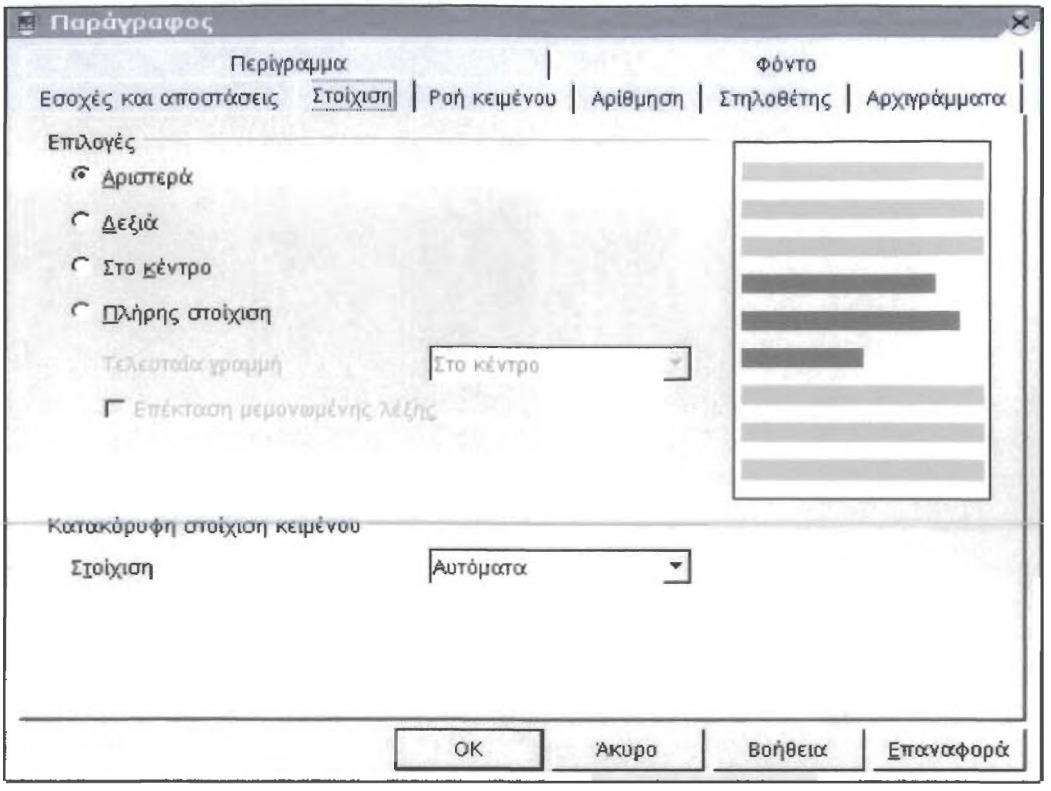
Αφού έχουμε ήδη γράψει το κείμενο που θέλουμε να μορφοποιήσουμε, τοποθετούμε το δείκτη του ποντικιού μέσα στην παράγραφο που θέλουμε να μορφοποιήσουμε και πατάμε το δεξί κουμπί. Από το μενού που εμφανίζεται, επιλέγουμε το **Παράγραφος (Paragraph)** οπότε και εμφανίζεται ένας διάλογος με οκτώ καρτέλες. Τον ίδιο διάλογο μπορούμε να καλέσουμε επιλέγοντας από το μενού **Μορφή (Format)** την επιλογή **Παράγραφος (Paragraph)**.

Εικόνα 13: Μορφοποίηση παραγράφων – Εσοχές & Αποστάσεις



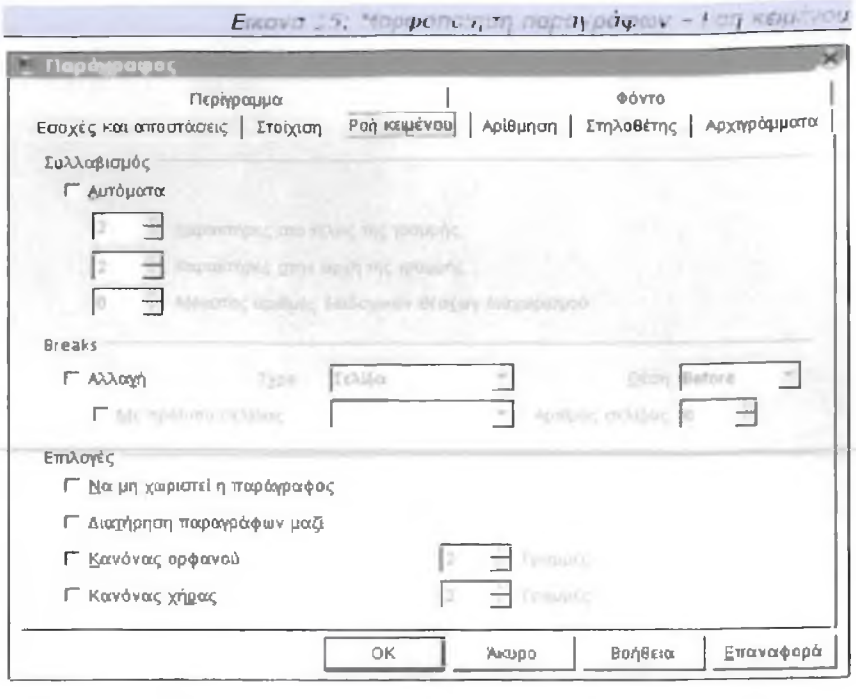
Από την καρτέλα **Εσοχές & Αποστάσεις (Indents & Spacing)** μπορούμε να επιλέξουμε την εσοχή που θα έχει η παράγραφος από το περιθώριο της σελίδας, τόσο **Από αριστερά (From Left)** όσο και **Από δεξιά (From Right)**. Ξεχωριστή εσοχή μπορεί να οριστεί για την **Πρώτη γραμμή (First line)** της παραγράφου. Μπορούμε επίσης να επιλέξουμε αν θέλουμε να υπάρχει ένα κενό, μία **Απόσταση (Spacing)** δηλαδή καθορισμένου μεγέθους, **Πάνω (Before)** ή / και **Κάτω (After)** από την παράγραφο. Εδώ ορίζουμε και τις αποστάσεις των γραμμών μεταξύ τους, το **Διάστιχο (Line Spacing)** δηλαδή, είτε από κάποιες προκαθορισμένες τιμές (απλή, διπλή, ένα και μισό διάστιχο, αναλογικό) ή με επιλογή
συγκεκριμένης τιμής απόστασης ως ελάχιστη ή σταθερή. Τέλος, η επιλογή **Επιβεβαίωση εγγραφής (Register True)** σημαίνει ότι το διάστιχο θα τροποποιηθεί τόσο όσο χρειάζεται, ούτως ώστε οι τελευταίες γραμμές σε κάθε σελίδα ή στήλη να είναι στην ίδια θέση. Αυτή είναι μία πολύ χρήσιμη επιλογή για περιπτώσεις δημιουργίας βιβλίων ή φυλλαδίων.

Εικόνα 14: Μορφοποίηση παραγράφων – Στοιχισι,



Από την καρτέλα **Στοίχιση (Alignment)** επιλέγουμε τη στοίχιση της παραγράφου. **Αριστερή (Left)** στοίχιση σημαίνει ότι το κείμενο από την αριστερή του πλευρά θα έχει κοινή αρχή, ενώ δεξιά θα εκτείνεται όσο περισσότερο γίνεται μέχρι να φτάσει στο άκρο του πλαισίου στο οποίο γράφουμε το κείμενο. **Δεξιά (Right)** στοίχιση σημαίνει το αντίθετο, ότι δηλαδή από δεξιά θα υπάρχει κοινό τέλος των γραμμών και αριστερά θα εκτείνονται όσο χρειάζεται. **Στο κέντρο (Center)** επιλέγουμε αν επιθυμούμε όλες οι γραμμές της παραγράφου να εκτείνονται το ίδιο δεξιά και αριστερά, δηλαδή το κέντρο της κάθε γραμμής να είναι στο κέντρο της σελίδας. Τέλος, **Πλήρη στοίχιση (Justified)** θα διαλέξουμε αν θέλουμε να έχουμε κοινή αρχή και κοινό τέλος στις γραμμές (όπως π.χ. στα βιβλία ή στις εφημερίδες). Ξεχωριστή ρύθμιση μπορεί να γίνει για την τελευταία γραμμή της παραγράφου στην περίπτωση της πλήρους στοίχισης, μια που η πλήρης στοίχιση επιτυγχάνεται μεγαλώνοντας τα κενά ανάμεσα στις λέξεις ή ακόμα και ανάμεσα στα γράμματα αν απαιτείται.

Η Στοίχιση (Alignment) της παραγράφου μπορεί να οριστεί εύκολα και γρήγορα από τη Γραμμή Αντικειμένων Κειμένου, όπου παρέχονται κουμπιά για Αριστερή, Δεξιά, Κεντραρισμένη και Πλήρη Στοίχιση.



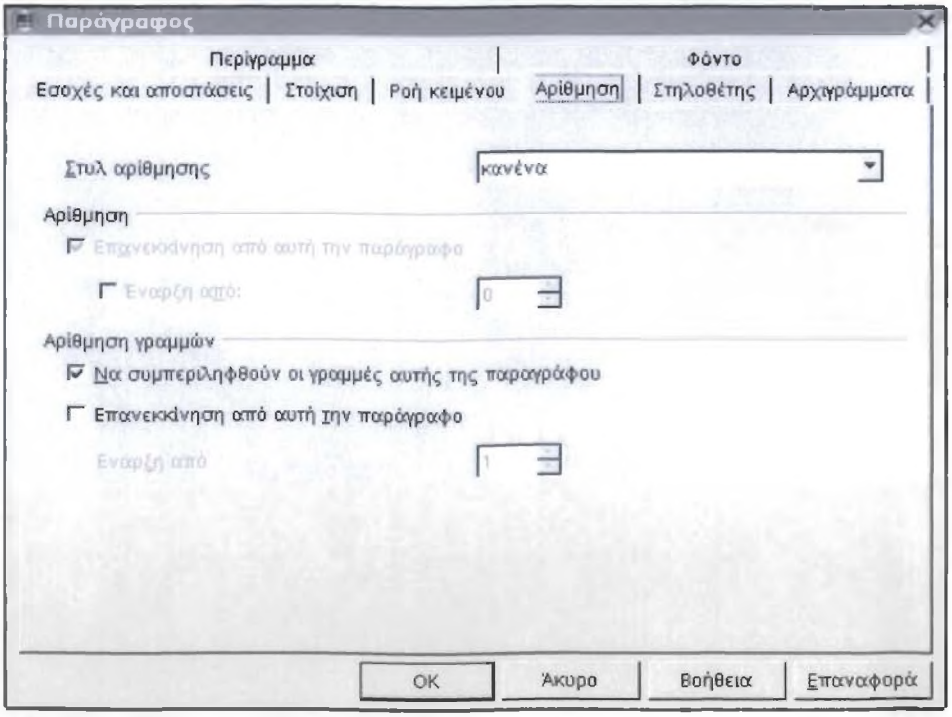
Για καλύτερα αισθητικά αποτελέσματα στην πλήρη και αριστερή στοίχιση, μπορεί να χρησιμοποιηθεί ταυτόχρονα ο συλλαβισμός, από την καρτέλα **Ροή Κειμένου (Text Flow)**. Συγκεκριμένα μπορούμε να ενεργοποιήσουμε τον αυτόματο συλλαβισμό, ορίζοντας ταυτόχρονα τον ελάχιστο αριθμό χαρακτήρων που θα απομείνουν σε κάθε γραμμή μετά το χωρισμό της λέξης. Μπορούμε επίσης και να ορίσουμε το **Μέγιστο αριθμό διαδοχικών θέσεων χωρισμού (Maximumno. of consecutive hyphens)** αποφεύγοντας έτσι την εμφάνιση διαχωριστικών παυλών σε πολλές συνεχόμενες γραμμές στη δεξιά πλευρά του κειμένου, κάτι που
θεωρείται «τυπογραφικά» λάθος.

Αν ορίσουμε 3 Χαρακτήρες στο τέλος της γραμμής (Characters at line end) και 2 Χαρακτήρες στην αρχή της γραμμής (Characters at line begin) τότε η λέξη «μείγμα» θα χωριστεί μεταξύ του γ και του μ, αφήνοντας 4 χαρακτήρες στην πάνω γραμμή και 2 στην κάτω. Αν όμως ορίσουμε 3 χαρακτήρες στην αρχή της γραμμής, η ίδια λέξη «μείγμα» δε θα χωριστεί, γιατί γραμματικά δεν μπορεί να χωριστεί αφήνοντας τουλάχιστον 3 χαρακτήρες μετά την παύλα χωρισμού.

Στην ίδια καρτέλα μπορούμε επίσης να ορίσουμε αν θέλουμε να επιβάλλεται μία **Αλλαγή (Break)**, σελίδας ή στήλης (αν η παράγραφός μας είναι σε σελίδα με πάνω από μία στήλες), **Πριν (Before)** ή **Μετά (After)** τη συγκεκριμένη παράγραφο. Εκτός από την υποχρεωτική αλλαγή, έχουμε τη δυνατότητα να ορίσουμε και αν θέλουμε να τοποθετηθεί όλη η παράγραφος μαζί (δηλαδή να **μη χωριστούν οι γραμμές (don't seperate lines)**) στην ίδια σελίδα, να ορίσουμε να
τοποθετηθεί στην ίδια σελίδα μαζί **με την επόμενη παράγραφο (keep with next paragraph)** καθώς και να ορίσουμε τον ελάχιστο αριθμό γραμμών που θα βρίσκονται σε κάθε σελίδα ή στήλη, αν η παράγραφος τύχει να είναι σε θέση που να χωρίζεται. Ο ελάχιστος αριθμός γραμμών από την αρχή της παραγράφου μέχρι το τέλος της σελίδας ορίζεται ως **Κανόνας Χήρας (Widow Control)** ενώ ο ελάχιστος αριθμός γραμμών στην επόμενη σελίδα ορίζεται ως **Κανόνας Ορφανού (Orphans control)**.

Αυτό τυπογραφικά λέγεται Έλεγχος Χηρών και Ορφανών, και χρησιμοποιείται επειδή είναι αντιαισθητικό να υπάρχει μία μόνο γραμμή από κάποια παράγραφο σε μία σελίδα.

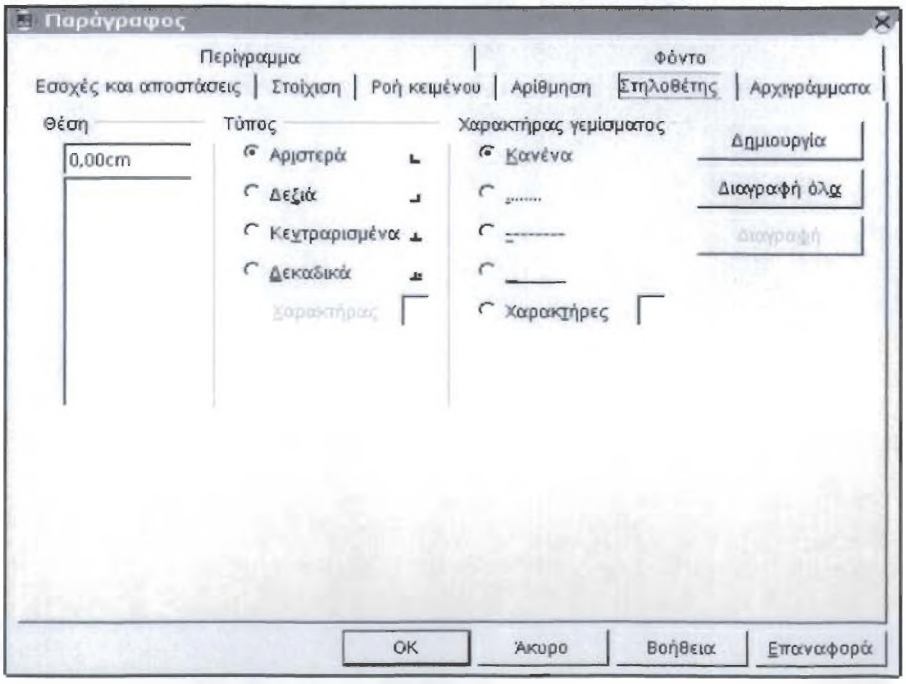
Εικόνα 16: Μορφοποίηση παραγράφων – Αρίθμηση



Ειδική καρτέλα υπάρχει για την **Αρίθμηση (Numbering)** στην οποία επιλέγουμε το **στυλ αρίθμησης (Numbering style)** της παραγράφου. Μπορούμε ακόμα να ορίσουμε, αν χρησιμοποιήσουμε ένα στυλ που έχει χρησιμοποιηθεί και νωρίτερα, να **επανεκκινήσει η αρίθμηση από την τρέχουσα παράγραφο (restart at this paragraph)**, και όχι μόνο από τον αριθμό 1 αλλά από όποιον αριθμό θέλουμε εμείς να ξεκινήσει (**Έναρξη από (start with)**). Η επιλογή **Αρίθμηση Γραμμών (Line Numbering)** ορίζει αν η συγκεκριμένη παράγραφος θα συμπεριληφθεί στην αρίθμηση γραμμών του κειμένου, αν αυτή ζητηθεί, από την επιλογή **Αρίθμηση Γραμμών (Line Numbering)** του μενού **Εργαλεία (Tools)**.

Αν στο κείμενό σας θέλετε να συμπεριλάβετε τμήμα κώδικα ενός προγράμματος, είναι καλό να ενεργοποιήσετε την αρίθμηση γραμμών για αυτό το τμήμα του κειμένου. Έτσι γίνεται εξάλλου στα περισσότερα βιβλία που περιέχουν μεγάλα τμήματα κώδικα, ως δείγμα για παράδειγμα.

Εικόνα 17: Μορφοποίηση παραγράφων – Στηλοθέτες



Οι **Στηλοθέτες (Tabs)** έχουν επίσης τη δική τους καρτέλα, δίνοντας τη δυνατότητα να προσθέσουμε στηλοθέτη σε όποια **Θέση (Position)** θέλουμε, να ορίσουμε τον **Τύπο (Type)** του στηλοθέτη, καθώς και το **Χαρακτήρα Γεμίσματος (Fill character)** αν επιθυμούμε. Οι τύποι των στηλοθετών που μπορούμε να ορίσουμε, είναι οι εξής:

- **Αριστερός (Left)**, που σημαίνει ότι το κείμενο θα ξεκινάει από αυτόν τοστηλοθέτη.

- **Δεξιός (Right)**, που σημαίνει ότι το κείμενο θα τελειώνει σε αυτόν το στηλοθέτη.

- **Κεντρικός (Centered)**, που σημαίνει ότι όσο κείμενο θα υπάρχει αριστερά του στηλοθέτη, άλλο τόσο θα υπάρχει και δεξιά του.

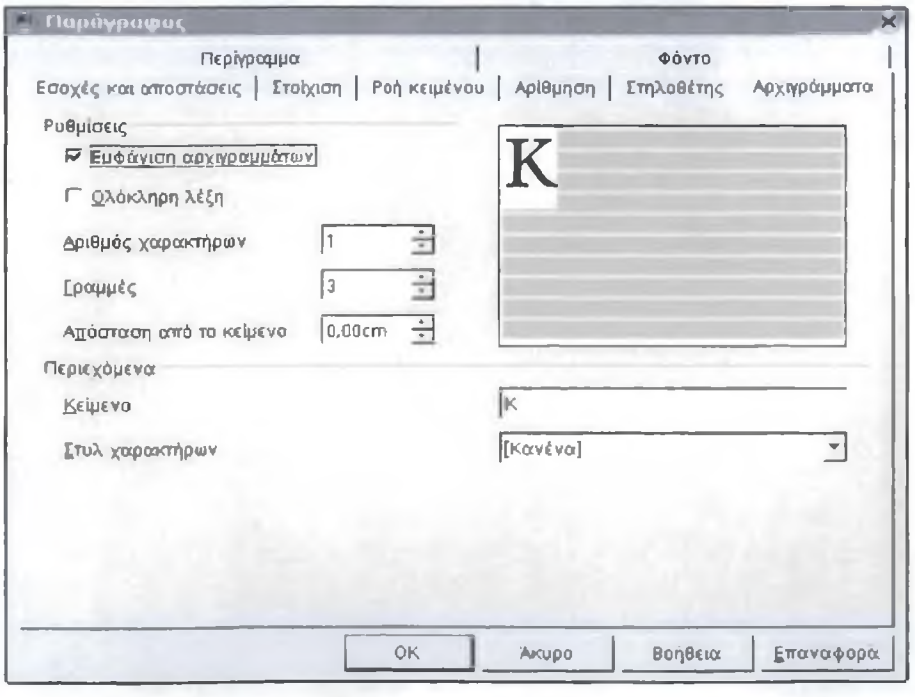
- **Δεκαδικός (Decimal)**, που σημαίνει ότι στη θέση του στηλοθέτη θα βρίσκεται ο χαρακτήρας που θα ορίσουμε στο πεδίο **Χαρακτήρας (Character)**.

Το όνομα Δεκαδικός προκύπτει επειδή συνήθως ορίζουμε ως χαρακτήρα το δεκαδικό σημείο και χρησιμοποιείται σε αριθμητικές εκφράσεις.

Ως χαρακτήρας γεμίσματος ορίζεται ένας χαρακτήρας ο οποίος εκτείνεται από τον ένα στηλοθέτη μέχρι τον επόμενο, και χρησιμοποιείται συνήθως στους πίνακες περιεχομένων.

Οι στηλοθέτες μπορούν να οριστούν και από το χάρακα που υπάρχει στο πάνω τμήμα της οθόνης. Ωστόσο, οι ρυθμίσεις αυτές δεν παρέχουν την ακρίβεια και την ευελιξία που παρέχει η καρτέλα Στηλοθέτες του διαλόγου για τη Μορφοποίηση παραγράφων.

Εικόνα 18: Μορφοποίηση παραγράφων - Αρχιγράμματα



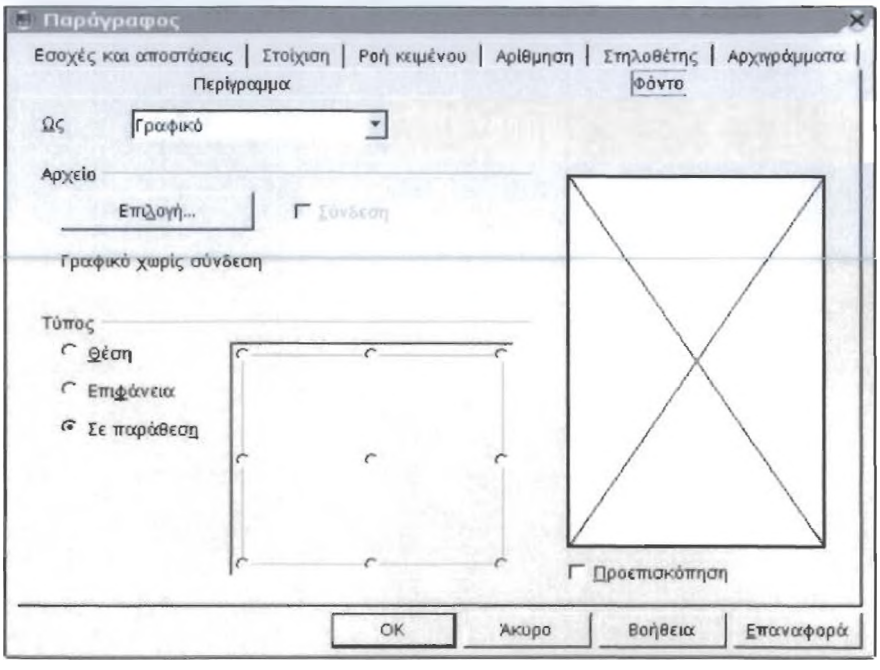
Η καρτέλα **Αρχιγράμματα (Drop Caps)** παρέχει μία λειτουργία που συναντάται στα *συστήματα επιτραπέζιας τυπογραφίας (DTP)*. Παραδοσιακά ως **Αρχίγραμμα (Drop Cap)** ορίζεται το πρώτο γράμμα μίας παραγράφου, το οποίο έχει την κορυφή του στο ίδιο επίπεδο με τα υπόλοιπα κεφαλαία γράμματα της πρώτης γραμμής, έχει αρκετά μεγαλύτερο μέγεθος και «εκτείνεται» μέσα στην παράγραφο.

Αυτή η παραδοσιακή έννοια, στα σύγχρονα προγράμματα έχει επεκταθεί και έχει γίνει πολύ πιο ευέλικτη. Συγκεκριμένα στο OpenOffice.org μπορεί κάποιος να ορίσει τον **Αριθμό Χαρακτήρων (Number of Characters)** που θα εμφανίζονται ως μεγαλύτεροι, ακόμα και **Ολόκληρη λέξη (Whole world)**,

καθώς και πόσες **Γραμμές (Lines)** κειμένου θα καταλαμβάνουν. Η **Απόσταση από το κείμενο (Space to text)** είναι επίσης παραμετροποιήσιμη ενώ μπορεί να οριστεί και το **Στυλ χαρακτήρα (Character style)** που θα έχουν τα αρχιγράμματα.

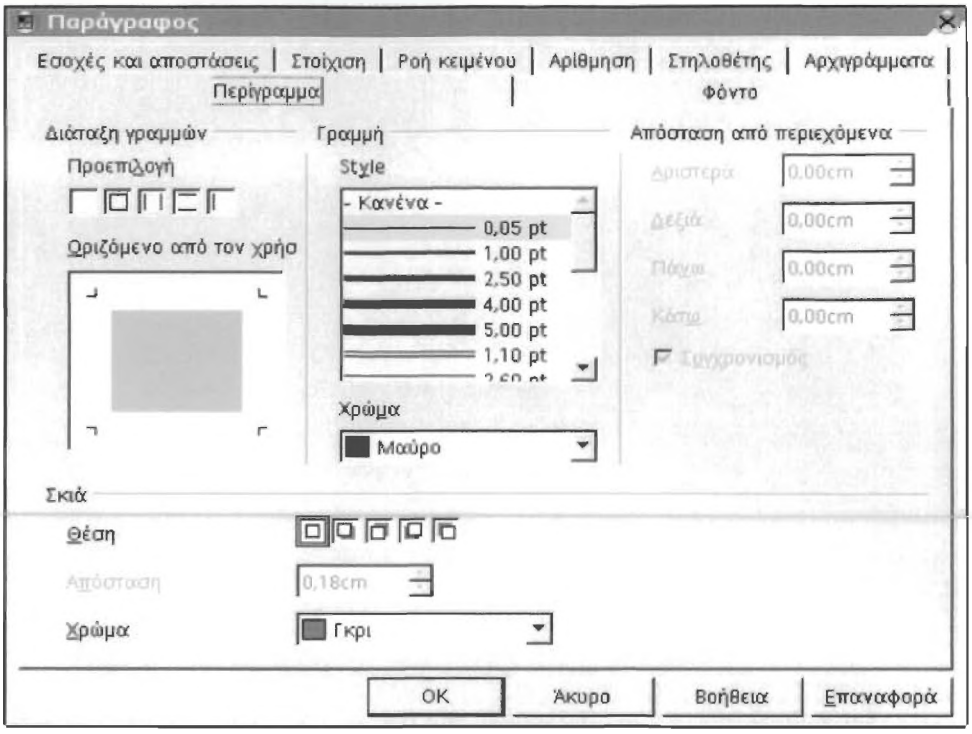
Θα μπορούσε εύκολα κάποιος να ορίσει ένα στυλ χαρακτήρα με μία διακοσμητική γραμματοσειρά, και να αναθέτει αυτό το στυλ στα αρχιγράμματα που δημιουργεί. Κάτι τέτοιο εξάλλου γίνεται σε πολλά λογοτεχνικά βιβλία.

Εικόνα 19: Μορφοποίηση παραγράφων - Φόντο



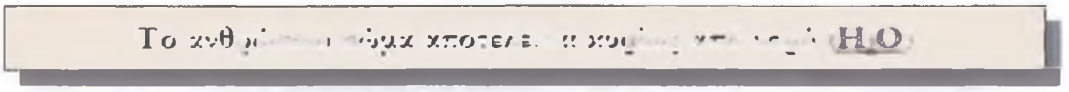
Η καρτέλα **Φόντο (Background)** μας επιτρέπει να ορίσουμε το φόντο της παραγράφου, είτε σαν **χρώμα (color)** ή σαν κάποιο **γραφικό (graphic)** της επιλογής μας. Αν επιλέξουμε το χρώμα, τότε η καρτέλα έχει την ίδια λειτουργικότητα με την αντίστοιχη του διαλόγου για την Μορφοποίηση χαρακτήρων. Αν επιλέξουμε γραφικό, μπορούμε να ορίσουμε αν θέλουμε να τοποθετηθεί είτε σε σταθερή **Θέση (Position)** ή πολλές φορές σε **παράθεση (Tile)**.

Εικόνα 20: Μορφοποίηση παραγράφων - Περίγραμμα



Τέλος, την καρτέλα **Περίγραμμα (Border)** θα τη συναντήσουμε σε πάρα πολλά σημεία της σουίτας. Από αυτή μπορούμε να επιλέξουμε γραμμές, του πάχους και του χρώματος που θέλουμε, οι οποίες θα τοποθετηθούν πάνω, κάτω, δεξιά ή και αριστερά από την παράγραφο. Η **απόσταση από τα περιεχόμενα (spacing to contents)** είναι παραμετροποιήσιμη. Από την ίδια καρτέλα, ορίζουμε και τη **σκιά (shadow style)** που θέλουμε να έχει η παράγραφός μας. Η **θέση (position)**, η **απόσταση (distance)** και το **χρώμα (color)** της σκιάς αλλάζουν κατά τη βούληση του χρήστη.

Ας δούμε πως μπορούμε να χρησιμοποιήσουμε το περίγραμμα της παραγράφου για να δημιουργήσουμε την εικόνα ενός πλαισίου γύρω από το κείμενο του παραδείγματος με το νερό. Εκτός από το περίγραμμα, θα προσθέσουμε και χρώμα φόντου καθώς και μία σκιά.



- Επιλέγουμε το κείμενο που έχουμε γράψει, και από την καρτέλα *Στοίχιση(Alignment)* επιλέγουμε την *Κεντραρισμένη (Centered)*.
- Πηγαίνουμε στην καρτέλα *Φόντο (Background)*. Διαλέγουμε ένα *χρώμα (color)*που θεωρούμε ότι ταιριάζει ως φόντο, για παράδειγμα το «Πορτοκαλί 4».
- Στη συνέχεια, πηγαίνουμε στην καρτέλα *Περίγραμμα (Border)*. Πατάμε το δεύτερο εικονάκι από αριστερά στο πεδίο *διάταξη γραμμών (Line arrangement)* και βλέπουμε ότι στις τέσσερις πλευρές της παραγράφου (πάνω, κάτω, δεξιά και αριστερά) προστίθενται ισάριθμες γραμμές του προκαθορισμένου πάχους(0,05 cm).

- Στην περιοχή *Απόσταση από περιεχόμενα (Spacing to contents)* ενεργοποιούμε το *Συγχρονισμό (Synchronize)* και μετατρέπουμε ένα από τα πεδία σε 0,15 cm.

Αυτό θα αφήσει 0,15 εκατοστά από το κείμενο μέχρι το περίγραμμα σε κάθε πλευρά. Η δεξιά και η αριστερή πλευρά του περιγράμματος, θα βρίσκονται (πάντα) στα όρια της παραγράφου, έτσι αυτή η ρύθμιση, για τις δύο αυτές γραμμές, σημαίνει την *ελάχιστη* απόσταση.

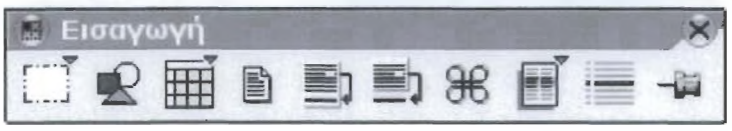
- Επιλέγουμε το δεύτερο από αριστερά εικονάκι στο πεδίο *Θέση (Position)* της περιοχής *Σκιά (Shadow Style)*. Αφήνουμε το *μέγεθος (distance)* και το *χρώμα(color)* στις προεπιλεγμένες τιμές.
- Πατάμε το Ok και βλέπουμε ότι η παράγραφός μας έχει πια φόντο, και ένα περίγραμμα με σκιά.

4) Γραφικά, Πίνακες και Πλαίσια

Το OpenOffice.org μας δίνει τη δυνατότητα να προσθέσουμε στο έγγραφό μας γραφικά, πίνακες και πλαίσια, είτε για λόγους εμφάνισης, ή για λόγους καλύτερης δόμησης και κατανόησης της πληροφορίας. Για να εισάγουμε έναν πίνακα, μία εικόνα ή ένα πλαίσιο, μπορούμε να χρησιμοποιήσουμε το μενού **Εισαγωγή(Insert)**, και συγκεκριμένα τις εντολές **Πίνακας (Table)**, **Πλαίσιο (Frame)** και **Γραφικό (Graphic)**. Εναλλακτικά, μπορούμε να πατήσουμε το κατάλληλο κουμπί στη **Γραμμή Εργαλείων (Main Toolbar)** στην αριστερή πλευρά της οθόνης.

Κρατώντας το κουμπί αυτό πατημένο για λίγο, εμφανίζεται μία επί πλέον γραμμή εργαλείων, η οποία είναι *κινούμενη (floating)*, μπορεί δηλαδή να μετακινηθεί οπουδήποτε θέλουμε, και περιέχει κουμπιά για εισαγωγή των στοιχείων αυτών, μεταξύ άλλων.

Εικόνα 21: Γραμμή εργαλείων εισαγωγής αντικειμένων



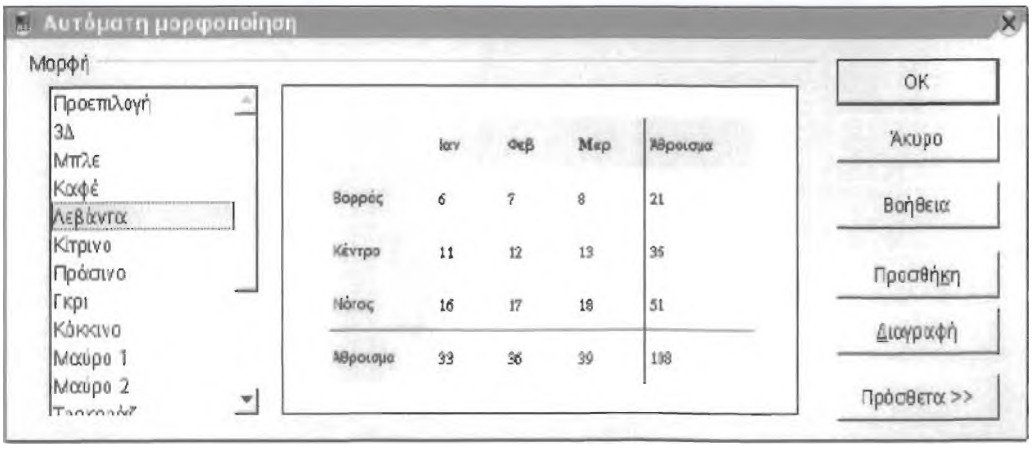
Αν επιλέξουμε να εισάγουμε ένα γραφικό (το δεύτερο από αριστερά εικονίδιο στη γραμμή εργαλείων), εμφανίζεται ένας τυπικός διάλογος επιλογής αρχείων. Ο ίδιος διάλογος εμφανίζεται αν ακολουθήσουμε τα μενού, και συγκεκριμένα από την επιλογή **Γραφικό (Graphic)**, επιλέξουμε την υποεπιλογή **Από Αρχείο (From file...)**. Στο διάλογο αυτό έχουμε τη δυνατότητα **προεπισκόπησης (preview)** του αρχείου που θα επιλέξουμε πριν το εισάγουμε στο έγγραφο, καθώς και τη δυνατότητα να το **Συνδέσουμε (Link)** ή όχι. Αν δεν επιλέξουμε τη σύνδεση, το γραφικό θα ενσωματωθεί μέσα στο έγγραφό μας. Αντίθετα, αν επιλέξουμε τη σύνδεση, το γραφικό δεν θα υπάρχει μέσα στο έγγραφο, αλλά σε αυτό θα υπάρχουν μόνο οι «οδηγίες» για το που θα βρεθεί αυτό το γραφικό. Αν λοιπόν επιθυμούμε να στείλουμε ένα αρχείο σε κάποιο συνεργάτη και έχουμε συνδεδεμένα γραφικά σε αυτό, πρέπει να στείλουμε μαζί και τα γραφικά. Προφανώς, το αρχείο που έχει συνδεδεμένα γραφικά, θα έχει πολύ μικρότερο μέγεθος από ότι αν είχε (το ίδιο αρχείο) ενσωματωμένα γραφικά.

	Ας υποθέσουμε ότι γράφουμε ένα βιβλίο το οποίο θα τυπωθεί και θα διανεμηθεί μέσω διαδικτύου σε ηλεκτρονική μορφή, και ότι το έντυπο βιβλίο θέλουμε να έχει ασπρόμαυρες εικόνες, ενώ το ηλεκτρονικό έγχρωμες. Αν συνδέσουμε τα γραφικά, μπορούμε να χρησιμοποιήσουμε το ίδιο ακριβώς αρχείο, απλά μεταφέροντας τις εικόνες που θέλουμε κάθε φορά στο κατάλληλο σημείο.

Στην περίπτωση εισαγωγής πίνακα, εμφανίζεται ένας διάλογος για να ορίσουμε τον αριθμό στηλών και γραμμών και το αν θα έχει ή όχι ο πίνακας επικεφαλίδα, η οποία μπορεί να επαναλαμβάνεται σε κάθε νέα σελίδα στην οποία πρέπει να επεκταθεί ο πίνακας. Μπορούμε επίσης να ορίσουμε αν θα επιτρέπεται να επεκτείνεται ο πίνακας σε νέα σελίδα ή αν θα πρέπει να βρίσκεται όλος μαζί σε μία σελίδα, καθώς και αν θα έχει περίγραμμα ή όχι. Όλες αυτές οι ρυθμίσεις μπορούν να αλλάξουν στη συνέχεια.

Η μορφοποίηση του πίνακα μπορεί να γίνει, μετά τη δημιουργία του, σχετικά αυτόματα. Από το μενού **Μορφή (Format)** μπορούμε να επιλέξουμε την **Αυτόματη Μορφοποίηση (Autoformat)**, για να εμφανιστεί ένας διάλογος με περισσότερες από 15 προκαθορισμένες μορφοποιήσεις, από τις οποίες μπορούμε να διαλέξουμε μία για να εφαρμοστεί στον πίνακα που δημιουργήσαμε.

Εικόνα 22. Διάλογος Αυτομορφοποίησης πίνακα



Για την εισαγωγή νέου πλαισίου δεν εμφανίζεται κανένας διάλογος, αλλά ο δείκτης του ποντικιού μετατρέπεται σε σταυρό, και πρέπει να «δημιουργήσουμε» το πλαίσιο τοποθετώντας το δείκτη – σταυρό σε ένα σημείο μέσα στη σελίδα, το οποίο θα είναι το ένα γωνιακό σημείο (π.χ. πάνω αριστερά), και κρατώντας πατημένο το αριστερό κουμπί του ποντικιού να το σύρουμε μέχρι το σημείο στο οποίο θέλουμε να είναι το άλλο γωνιακό σημείο (δηλαδή το κάτω δεξιά για το παράδειγμά μας).

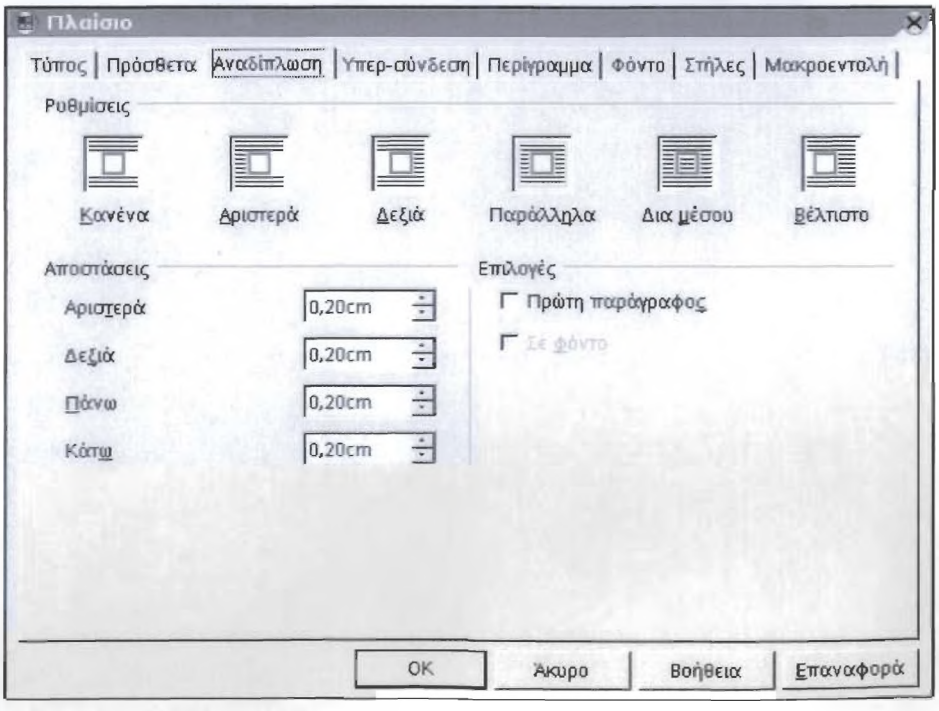
5) Κοινές ρυθμίσεις

Όταν τοποθετήσουμε το αντικείμενο που θέλουμε μέσα στο έγγραφό μας, επιλέγοντάς το και πατώντας το δεξί κουμπί του ποντικιού, εμφανίζεται το μενού περιεχομένου. Αυτό, μεταξύ άλλων, περιέχει και μία εγγραφή η οποία κατά περίπτωση ονομάζεται **Γραφικό (Gpaphics)**, **Πίνακας (Table)** ή **Πλαίσιο (Frame)**. Αυτή η εγγραφή ανοίγει το διάλογο «ελέγχου» του αντικειμένου, ο οποίος αποτελείται από διάφορες καρτέλες. Δύο από αυτές, είναι κοινές για όλα τα αντικείμενα, συγκεκριμένα οι καρτέλες **Φόντο (Background)** και **Περίγραμμα (Borders)**. Οι καρτέλες αυτές παρέχουν την ίδια λειτουργικότητα με αυτές του διαλόγου για τη Μορφοποίηση παραγράφων. Ωστόσο, ειδικά για τον πίνακα, μπορούν να ρυθμιστούν επιπλέον και τα περιγράμματα που θα βρίσκονται ανάμεσα στα γραμμές και τις στήλες, κάτι που δεν έχει νόημα για τα πλαίσια και τα γραφικά.

6) Αναδίπλωση κειμένου

Για τις εικόνες και τα πλαίσια, μπορούμε να ορίσουμε την αναδίπλωση του κειμένου γύρω τους, καθώς και τη θέση που θα πάρουν στο έγγραφό μας. Από το διάλογο ελέγχου του αντικειμένου, επιλέγουμε την καρτέλα **Αναδίπλωση (Wrap)** η οποία μας παρέχει 6 διαφορετικές ρυθμίσεις.

Εικόνα 23: Μορφοποίηση εικόνας ή πλαισίου – Αναδίπλωση



- **Κανένα (None)** : Το πλαίσιο ή η εικόνα, θα είναι σε μία ξεχωριστή γραμμή μέσα στο έγγραφό μας, και το ύψος της θα είναι τόσο όσο χρειάζεται για να χωρέσει το αντικείμενο. Το κείμενο θα σταματάει στην προηγούμενη γραμμή και θα ξαναρχίζει στην επόμενη.

- **Αριστερά (Before)** : Το πλαίσιο ή η εικόνα δεν θα είναι σε ξεχωριστή γραμμή, αλλά θα καταλαμβάνει όσες κανονικές γραμμές κειμένου χρειάζεται (ανάλογα με το ύψος του). Το κείμενο θα συνεχίζει κανονικά και σε αυτές τις γραμμές, ωστόσο θα σταματάει μόλις «συναντήσει» το αντικείμενο και θα συνεχίζει από την επόμενη γραμμή. Το αντικείμενο δηλαδή θεωρείται ότι ορίζει το τέλος της γραμμής.

- **Δεξιά (After)** : Αυτή η ρύθμιση είναι ίδια με την **Αριστερά** με τη διαφορά ότι το κείμενο δεν βρίσκεται αριστερά του αντικειμένου, αλλά δεξιά. Το αντικείμενο δηλαδή, θεωρείται ότι ορίζει την αρχή της γραμμής.

- **Παράλληλα (Parallel)** : Με αυτή τη ρύθμιση, το κείμενο απλά παρακάμπτει το αντικείμενο που έχουμε εισάγει. Το κείμενο τοποθετείται τόσο πριν όσο και μετά από το αντικείμενο, αρκεί να υπάρχει ο σχετικός χώρος.

- **Διά μέσου (Through)** : Όταν ενεργοποιηθεί αυτή η ρύθμιση, το κείμενο δεν επηρεάζεται καθόλου από την ύπαρξη του αντικειμένου, αλλά η ροή του συνεχίζεται κανονικά σα να μην υπάρχει τίποτα εκεί. Περνάει δηλαδή διά μέσου του αντικειμένου.

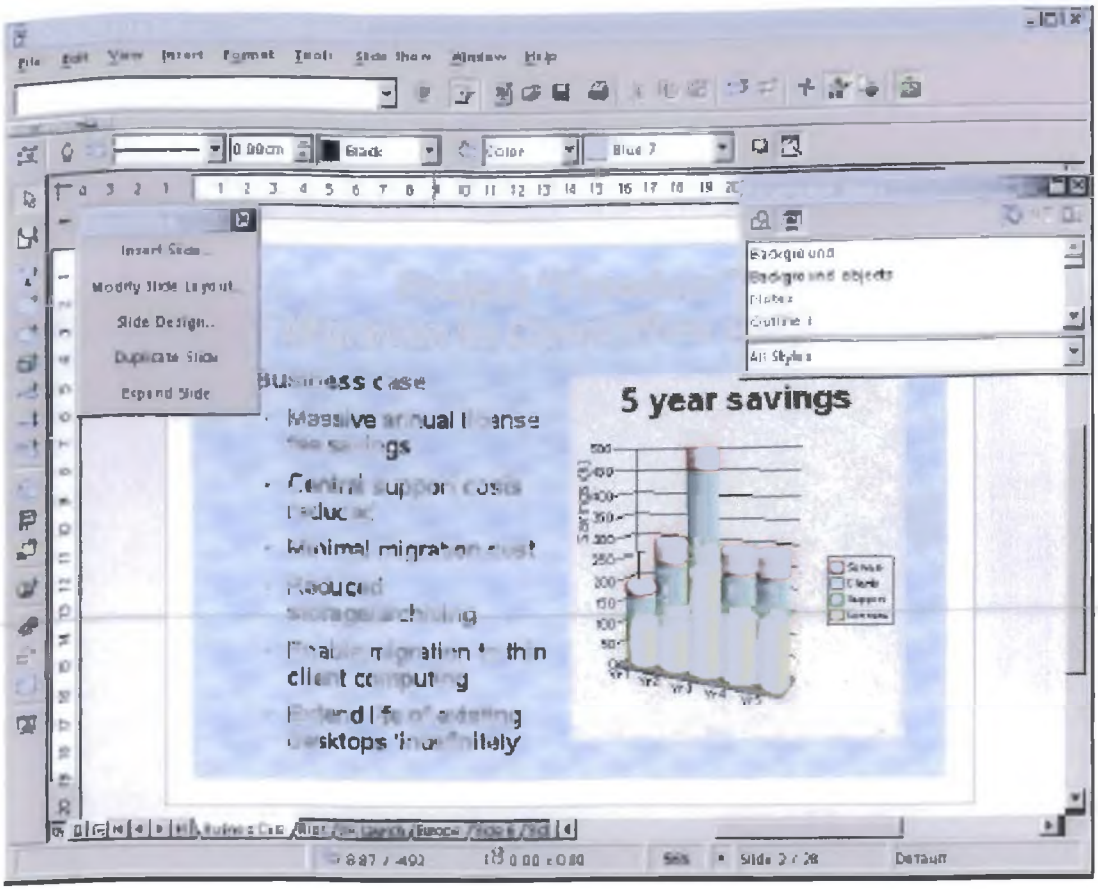
- **Βέλτιστο (Optimal)** : Με αυτή τη ρύθμιση το κείμενο βρίσκεται δεξιά από το αντικείμενο, αριστερά από το αντικείμενο, ή μόνο πάνω και κάτω από το αντικείμενο, ανάλογα με τη θέση και το μέγεθος του αντικειμένου. Αν η απόσταση μεταξύ του αντικειμένου και του περιθωρίου του εγγράφου είναι μικρότερη από 2 εκατοστά, δεν τοποθετείται κείμενο εκεί. Στα αντικείμενα τα οποία έχουν λιγότερο από 1,5 εκατοστό πλάτος και απέχουν τουλάχιστον 2 εκατοστά από την κάθε άκρη του εγγράφου, εφαρμόζεται η *Παράλληλη(Parallel)* ρύθμιση.

Εκτός από την ροή του κειμένου, μπορούμε να ορίσουμε και την απόσταση που θα έχει το εισηγμένο αντικείμενο από το κείμενο. Οι αποστάσεις πάνω, κάτω, δεξιά και αριστερά από το αντικείμενο μπορούν να ρυθμιστούν ξεχωριστά.

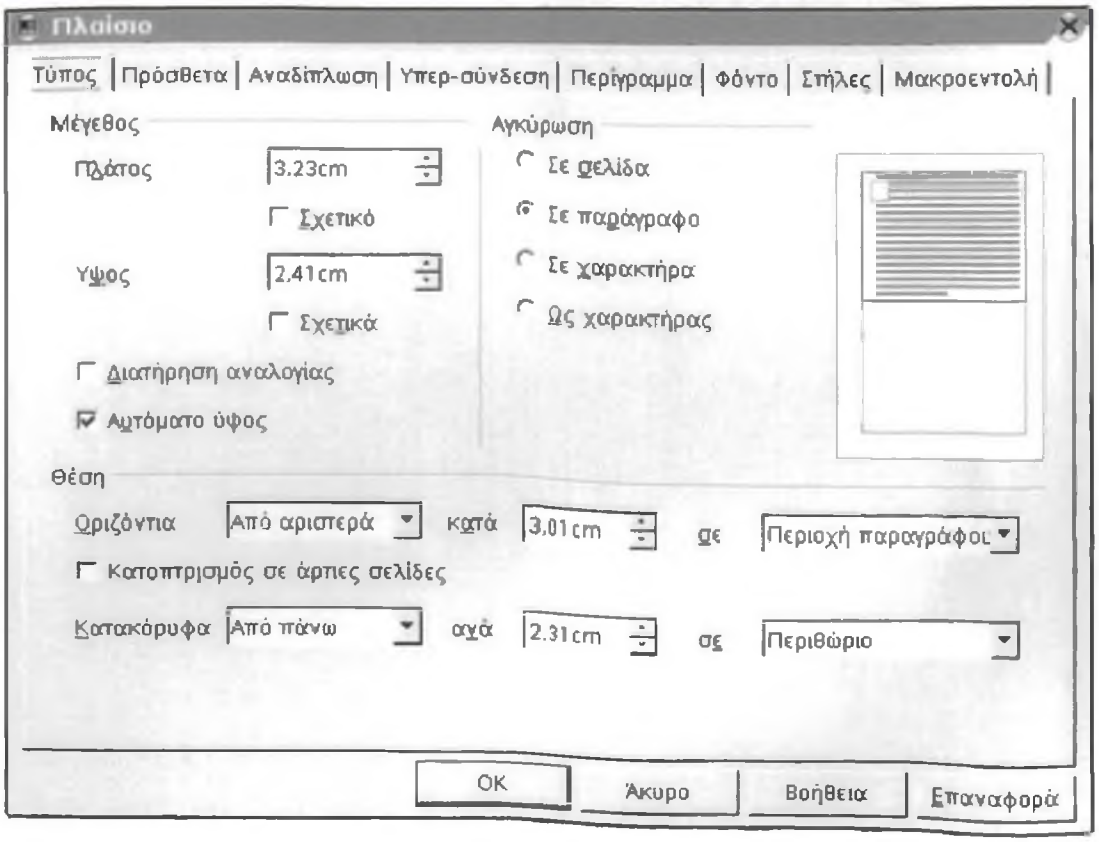
Αν έχει ενεργοποιηθεί η επιλογή **Πρώτη Παράγραφος (First Paragraph)** τότε ξεκινάει μία νέα παράγραφος κάτω από το αντικείμενο, ενώ η επιλογή **Σε φόντο(In Background)** μπορεί να επιλεγεί μόνο αν έχουμε επιλέξει τη ρύθμιση ροής **Διά μέσου (Through)** και καθορίζει αν το αντικείμενο που εισάγαμε θα είναι πίσω (κάτω) από το κείμενο ή μπροστά (πάνω) από αυτό.

7) Θέση αντικειμένου

Σχεδόν ίδιες είναι και οι ρυθμίσεις της θέσης του αντικειμένου που έχουμε εισάγει, με τις οποίες μπορούμε να ορίσουμε τη σχετική θέση του αντικειμένου αυτού, καθώς και το μέγεθός του. Αυτές οι ρυθμίσεις έχουν έννοια και παρέχονται μόνο για πλαίσια και εικόνες.



Εικόνα 24: Μορφοποίηση εικόνας ή πλαισίου – Τύπος



Από το διάλογο ελέγχου, επιλέγουμε την καρτέλα **Τύπος (Type)**. Αυτή η καρτέλα χωρίζεται σε τρία τμήματα, που ονομάζονται **Μέγεθος (Size)**, **Αγκύρωση(Anchor)** και **Θέση (Position)**. Από το **Μέγεθος (Size)** μπορούμε να ορίσουμε το **Πλάτος (Width)** και το **Ύψος (Height)** του αντικειμένου, ορίζοντας ταυτόχρονα αν θα επιβάλλεται η **Διατήρηση Αναλογίας (Keep ratio)** σε τυχόν αλλαγές μεγέθους. Από το τμήμα που λέγεται **Αγκύρωση (Anchor)** μπορούμε να ορίσουμε αν η θέση του αντικειμένου θα είναι σχετική με **την παράγραφο (To paragraph)**, **τη σελίδα (To page)**, **το χαρακτήρα (To character)** ή αν θα ενσωματωθεί στο κείμενο ως ένας ακόμα **Χαρακτήρας (As character)**. Τέλος, από τη **Θέση (Position)** ορίζουμε την οριζόντια και κατακόρυφη απόσταση που
θα έχει το αντικείμενο που εισάγαμε, από το σημείο αγκύρωσης, ενώ αν το αντικείμενο βρίσκεται σε σημείο που επαναλαμβάνεται σε όλες τις σελίδες (π.χ. κεφαλίδα), μπορούμε να ορίσουμε αν θα καθρεφτίζεται η θέση του στις αντικρυστές σελίδες ή αν θα είναι σταθερή.

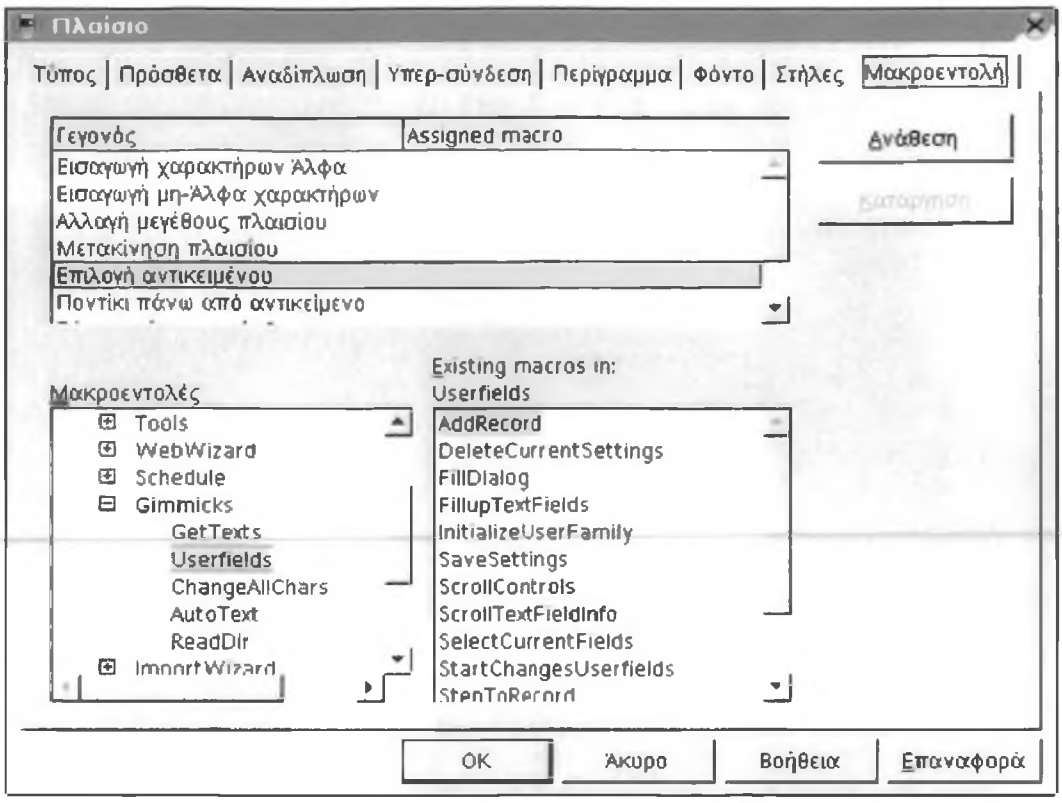
8) Υπερ-σύνδεση (Hyperlink)

Για τα πλαίσια και τις εικόνες που έχουμε εισάγει στο έγγραφό μας, μπορούμε να ορίσουμε έναν υπερ-σύνδεσμο (hyperlink) που θα ανοίγει όταν πατάμε με το ποντίκι πάνω στο σχετικό αντικείμενο. Ο **Υπερ-σύνδεσμος (Hyperlink)** ορίζεται από την ομώνυμη καρτέλα του διαλόγου ελέγχου του αντικειμένου, και έχει τις ίδιες ρυθμίσεις με τον υπερ-σύνδεσμο όπως τον είδαμε στη Μορφοποίηση χαρακτήρων.

9) Αντιστοίχηση Μακροεντολών

Τόσο στα πλαίσια όσο και στις εικόνες, υπάρχουν ορισμένα **Γεγονότα (Events)**,όπως π.χ. η **Επιλογή του αντικειμένου (Click object)** στα οποία μπορούμε να αντιστοιχίσουμε **Μακροεντολές (Macros)** που θα εκτελούνται όποτε συμβαίνει το αντίστοιχο γεγονός.

Εικόνα 25: Μορφοποίηση εικόνας ή πλαισίου – Μακροεντολές



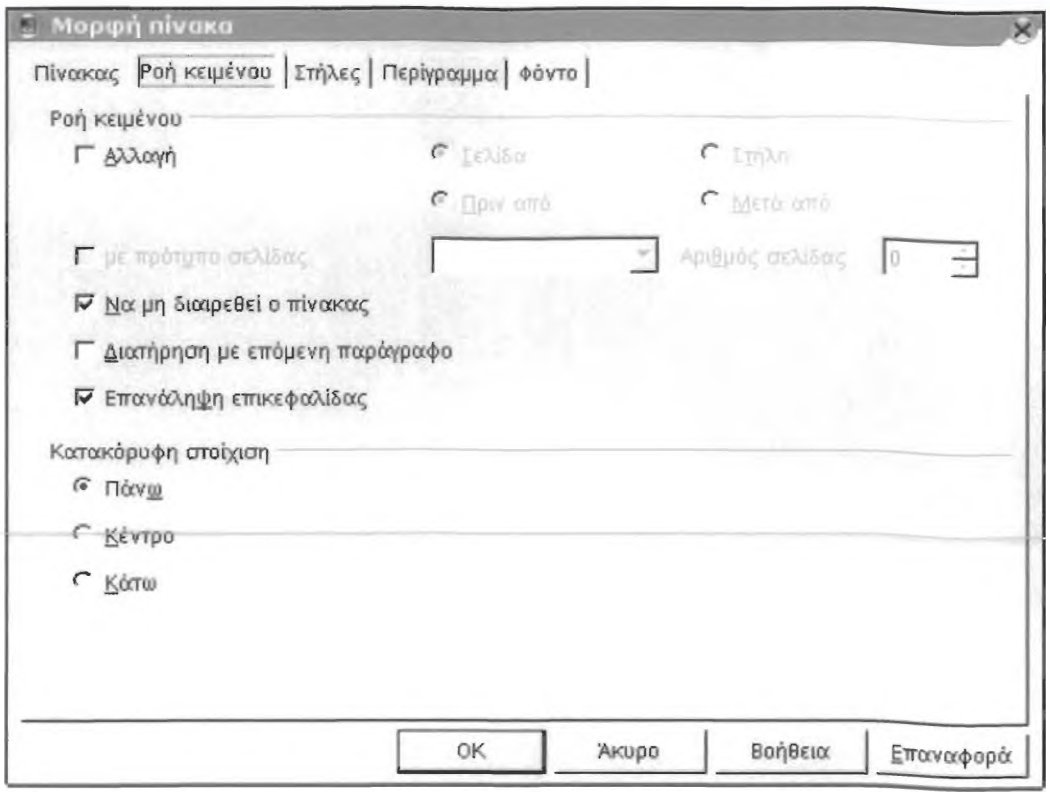
10) Ειδικές ρυθμίσεις για τους πίνακες

Από το διάλογο ελέγχου του πίνακα, έχουμε τρεις επί πλέον καρτέλες, την καρτέλα με όνομα **Ροή Κειμένου (Text Flow)**, την καρτέλα που ονομάζεται **Στήλες(Columns)** και την καρτέλα **Πίνακα (Table)**. Αυτές οι καρτέλες δίνουν τις εξής δυνατότητες:

10.1) Ροή Κειμένου

Στην καρτέλα **Ροή κειμένου (Text Flow)** μπορούμε να ορίσουμε αν θέλουμε να υπάρχει μία **Αλλαγή (Break) Σελίδας (Page)** ή **Στήλης (Column)**, **πρίν(before)** ή **μετά (after)** από τον πίνακα. Μπορούμε επίσης να ορίσουμε αν θέλουμε **Να μη διαιρεθεί ο πίνακας (Don't split table)**, να διατηρηθεί **με την επόμενη παράγραφο (keep with next paragraph)** ή τέλος να έχουμε **Επανάληψη Επικεφαλίδας (Repeat heading)** κάθε φορά που ο πίνακας εκτείνεται σε μία νέα σελίδα.

Εικόνα 23: Μορφοποίηση πίνακα - Ροή κειμένου

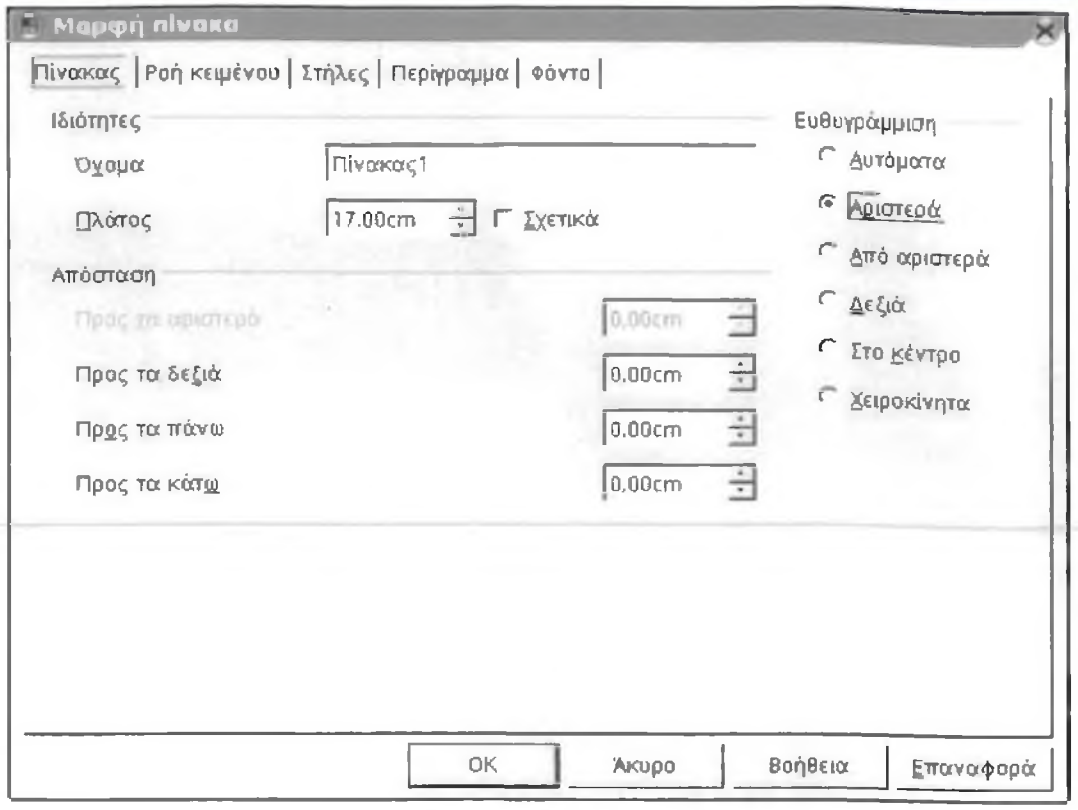


Από την ίδια καρτέλα, ορίζουμε και την **Κατακόρυφη στοίχιση (Vertical Alignment)** των περιεχομένων των κελιών του πίνακα, αλλά αυτή η ρύθμιση ισχύει για κάθε κελί ξεχωριστά.

10.2) Πίνακας

Στην καρτέλα **Πίνακας (Table)** ορίζουμε τις **Ιδιότητες (Properties)** του πίνακα. Βασικές ιδιότητες είναι το **όνομα (Name)** του πίνακα, το **πλάτος** του (το οποίο μπορεί να ορίζεται ως απόλυτη αριθμητική τιμή, ή **Σχετικά (Relative)** ως ποσοστό) καθώς και η **Ευθυγράμμισή (Alignment)** του.

Εικόνα 27. Μορφοποίηση πίνακα – Πίνακας



Αναφερόμαστε στη στοίχιση του πίνακα εντός της σελίδας, και όχι των περιεχομένων εντός του πίνακα. Οι διαφορετικές ρυθμίσεις έχουν τις εξής έννοιες:

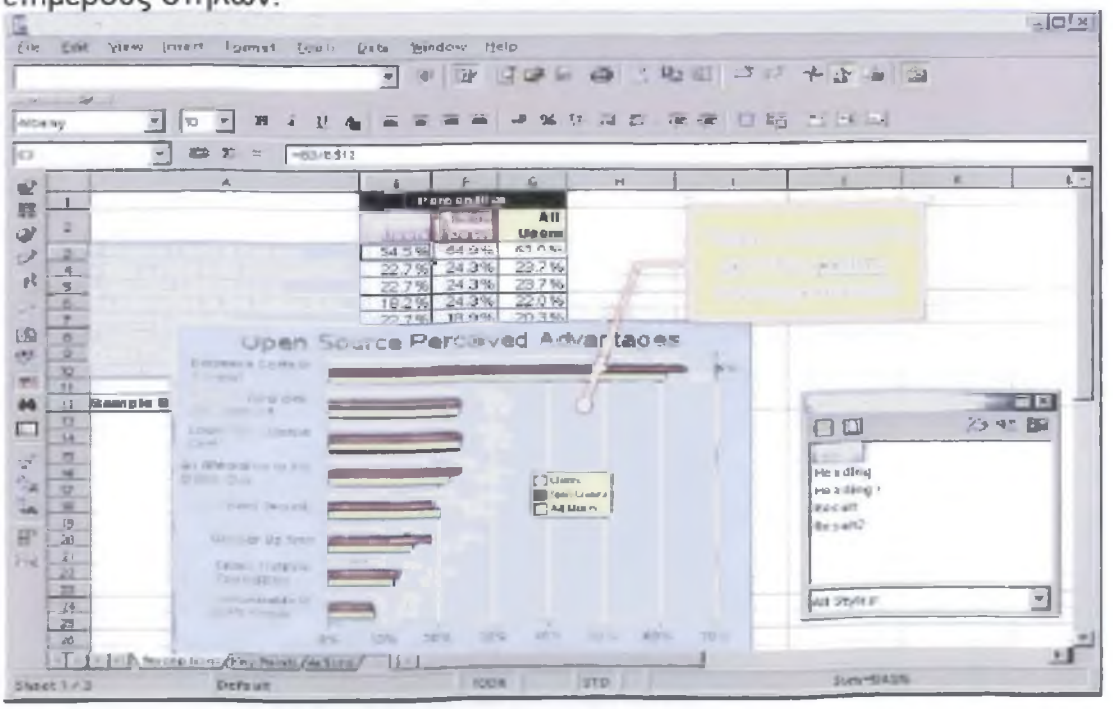
- **Αυτόματα (Automatic)** : Η αριστερότερη στήλη του πίνακα ξεκινάει στο αριστερό περιθώριο της σελίδας και η δεξιότερη στήλη του πίνακα τελειώνει στο δεξί περιθώριο της σελίδας.
- **Αριστερά (Left)** : Η αριστερότερη στήλη του πίνακα ξεκινάει στο αριστερό περιθώριο της σελίδας. Ο πίνακας εκτείνεται τόσο όσο δηλώνεται στο πεδίο **Πλάτος (width)** του τμήματος **Ιδιότητες (Properties)**.
- **Από Αριστερά (From left)** : Η αριστερότερη στήλη του πίνακα, ξεκινάει τόσο όση δηλώνεται στο πεδίο **Προς τα Αριστερά (Left)** του τμήματος **Απόσταση (Spacing)** από το αριστερό περιθώριο της σελίδας. Το πλάτος του πίνακα ορίζεται από το πεδίο **Πλάτος (width)** του τμήματος **Ιδιότητες(Properties)**.
- **Δεξιά (Right)** : Η δεξιότερη στήλη του πίνακα θα τελειώνει στο δεξί περιθώριο της σελίδας, ενώ ο πίνακας θα έχει τόσο πλάτος όσο δηλώνεται στο πεδίο **Πλάτος (width)** του τμήματος **Ιδιότητες (Properties)**. Αν το πεδίο **Προς τα Δεξιά (Right)** του τμήματος **Απόσταση (Spacing)** δεν είναι 0, τότε η δεξιότερη στήλη δε θα βρίσκεται στο περιθώριο, αλλά όσο λέει αυτό το πεδίο προς το μέσο της σελίδας.
- **Στο κέντρο (Centered)** : Ο πίνακας θα απέχει τόσο από δεξιά, όσο και από αριστερά στη σελίδα. Θα είναι δηλαδή στο κέντρο της. Το πόσο θα απέχει από δεξιά και αριστερά ορίζεται από το πεδίο **Προς τα Αριστερά (Left)** του τμήματος **Απόσταση (Spacing)** ή υπολογίζεται από την τιμή του πεδίου **Πλάτος (Width)** του τμήματος **Ιδιότητες (Properties)**. Το τμήμα **Απόσταση**

(Spacing), εκτός από τις τιμές **Προς τα Αριστερά (Left)** και **Προς τα Δεξιά (Right)** που συναντήσαμε πιο πάνω, έχει επίσης τις τιμές **Προς τα Πάνω (Above)** και **Προς τα Κάτω (Below)** που ορίζουν την απόσταση μεταξύ του πίνακα και της τελευταίας γραμμής της προηγούμενης παραγράφου ή της πρώτης γραμμής της επόμενης παραγράφου αντίστοιχα.

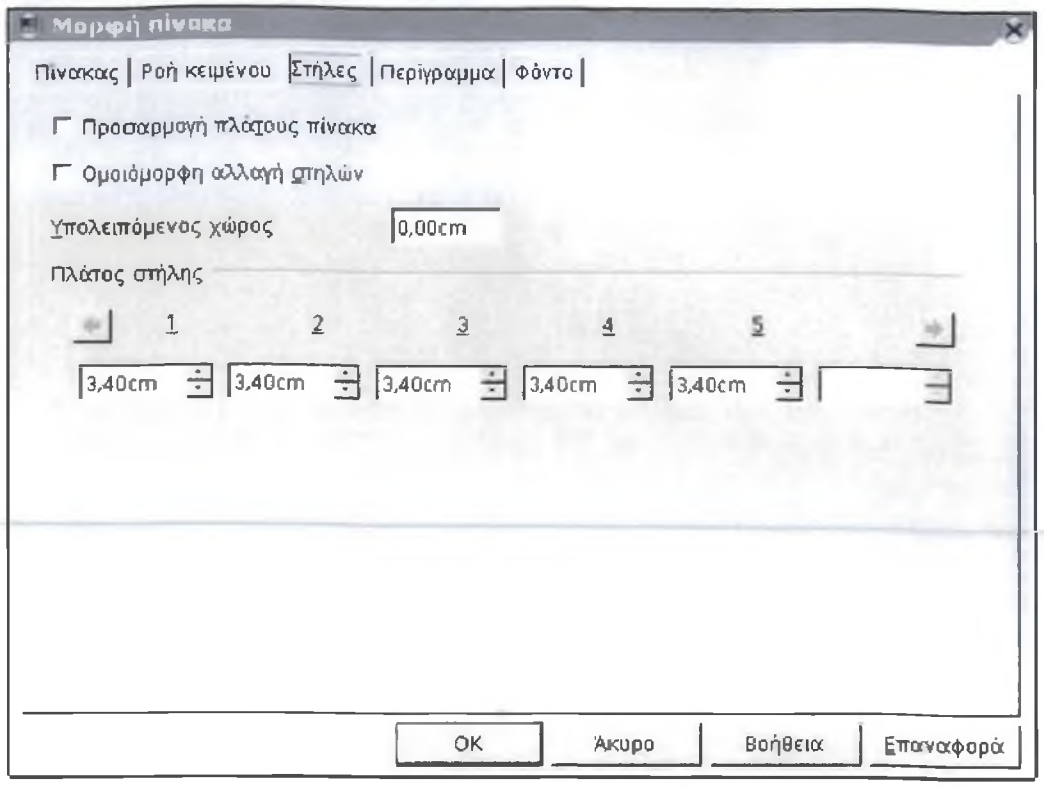
Όλες οι προαναφερθείσες τιμές αποστάσεων δίνονται με νούμερα ή **Σχετικά (Relative)** αν το αντίστοιχο πεδίο είναι επιλεγμένο.

10.3) Στήλες

Τελευταία καρτέλα του διαλόγου ελέγχου ενός πίνακα, είναι αυτή που ονομάζεται **Στήλες (Columns)** και μας επιτρέπει να ορίζουμε ξεχωριστά για κάθε στήλη του πίνακα το πλάτος της. Αν στην καρτέλα **Πίνακας** δεν έχουμε επιλέξει τις **Σχετικές (Relative)** τιμές και οποιαδήποτε άλλη στοίχιση εκτός από την **Αυτόματη (Automatic)** τότε έχουμε τη δυνατότητα να επιλέξουμε και **Ομοιόμορφο Πλάτος Στηλών (Adjust Columns Propostionally)** ή την αυτόματη **Προσαρμογή Πλάτους Πίνακα (Adapt Table Width)** ούτως ώστε να μην επηρεάζεται το συνολικό του πλάτος από τυχόν αλλαγές πλάτους των επιμέρους στηλών.



Εικόνα 28: Μορφοποίηση πίνακα – Στήλες



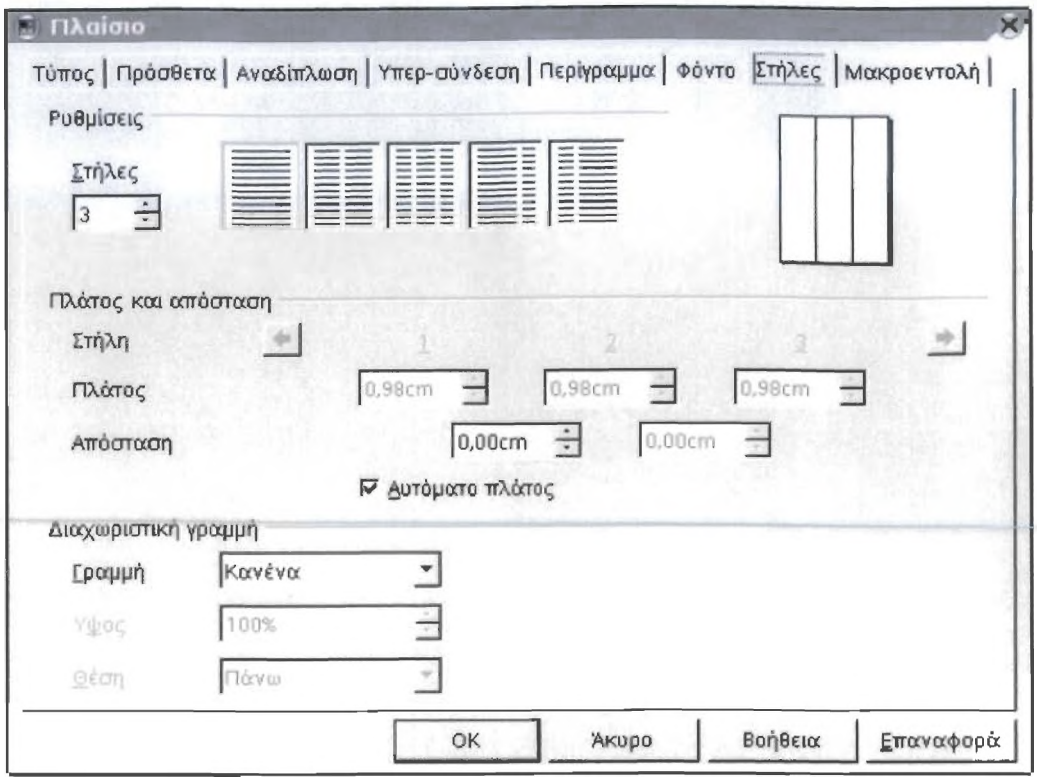
11) Ειδικές ρυθμίσεις για τα πλαίσια

Από το διάλογο ελέγχου του πλαισίου, έχουμε επί πλέον τη δυνατότητα να ορίσουμε:

11.1) Στήλες (Columns)

Τα πλαίσια, μπορούμε να τα φανταστούμε ως «σελίδες» των διαστάσεων που εμείςθέλουμε και ορίζουμε. Ως σελίδες, μπορεί να αποτελούνται από μία ή περισσότερες στήλες. Στην καρτέλα αυτή λοιπόν ορίζουμε τον αριθμό των στηλών που θα έχει το πλαίσιο, το **Πλάτος (Width)** της κάθε μίας από αυτές (ή το ίδιο πλάτος για όλες αν επιλέξουμε το **Αυτόματο Πλάτος (AutoWidth)**) και τυχόν **Απόσταση (Spacing)** που θέλουμε να έχουν μεταξύ τους. Αν θέλουμε να έχουμε μία
Διαχωριστική γραμμή (Seperator Line) που θα εμφανίζεται μεταξύ των στηλών, πρέπει να ορίσουμε το **Ύψος (Height)** που θα έχει αναλογικά με τη στήλη, και αν αυτό είναι οτιδήποτε άλλο από 100%, σε ποιο σημείο της στήλης θα εμφανίζεται (π.χ. **Πάνω (Top)**)

Εικόνα 29: Μορφοποίηση πλαισίου - Στήλες

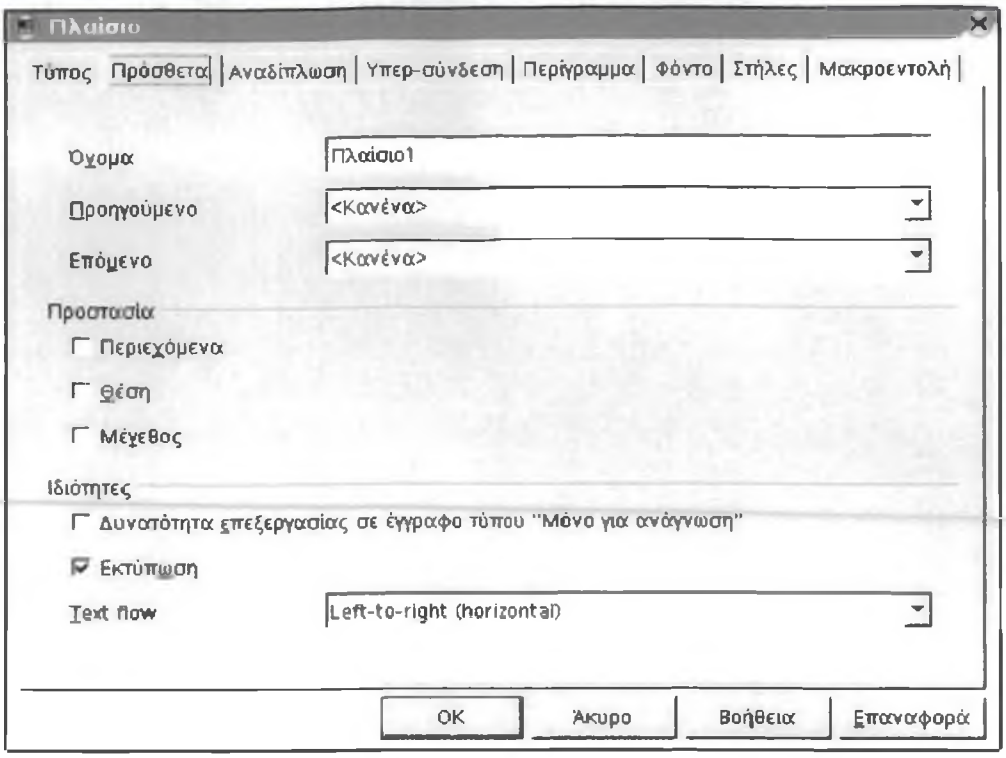


11.2) Πρόσθετα (Options)

Από την καρτέλα αυτή ορίζουμε το **Όνομα (Name)** του πλαισίου, το **Προηγούμενο (Previous)** και **Επόμενο (Next)** πλαίσιο αν υπάρχουν (για λόγους σύνδεσης μεταξύ τους). Εδώ επίσης μπορούμε να ορίσουμε, τα **Περιεχόμενα(Contents)**, το **Μέγεθος (Size)** και τη **Θέση (Position)** του πλαισίου αν θέλουμε να είναι προστατευμένα (να μην μπορούν να τροποποιηθούν δηλαδή).

Τέλος, μπορούμε να ορίσουμε αν το πλαίσιο θα εμφανίζεται σε τυχόν **Εκτύπωση(Print)** καθώς και αν θα δίνει τη **Δυνατότητα Επεξεργασίας σε έγγραφο τύπου «Μόνο για Ανάγνωση» (Editable in read-only document)**.

Εικόνα 30: Μορφοποίηση πλαισίου – Πρόσθετα



12) Ειδικές ρυθμίσεις για τα γραφικά

Από το διάλογο ελέγχου ενός γραφικού, έχουμε επί πλέον τη δυνατότητα να ορίσουμε:

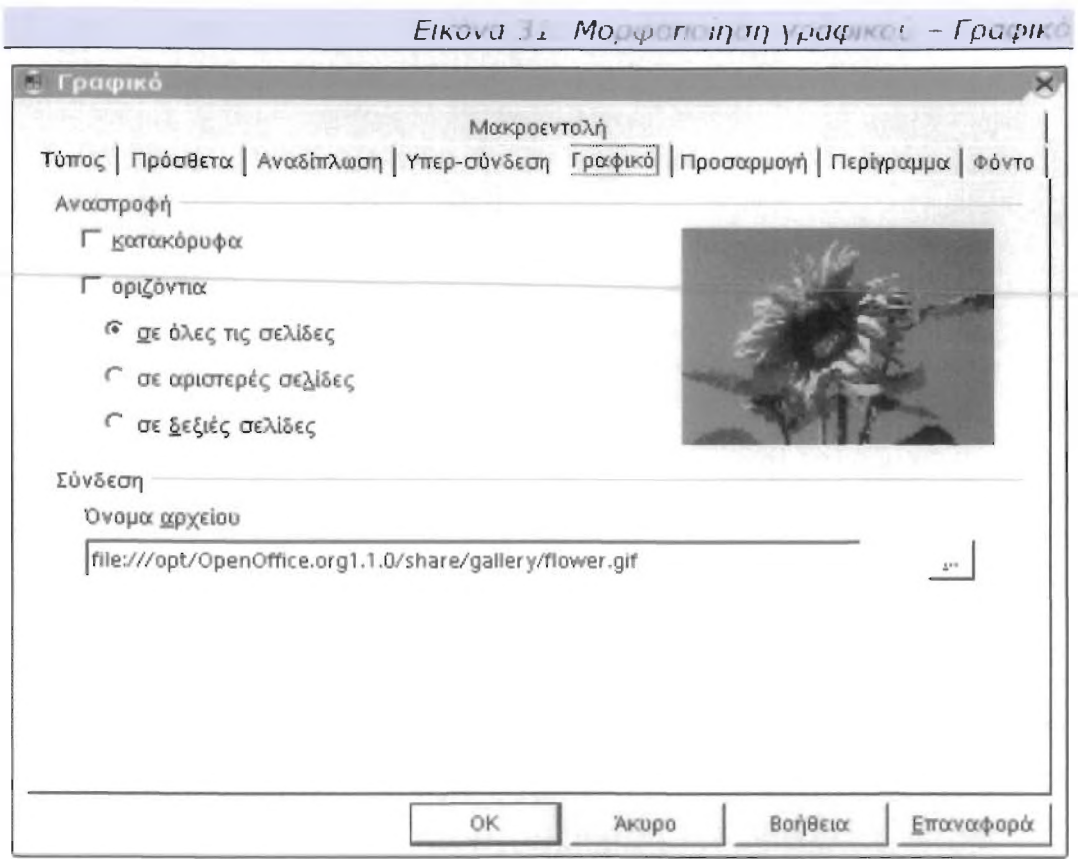
12.1)Πρόσθετα (Options)

Από την καρτέλα αυτή ορίζουμε το **Όνομα (Name)** του γραφικού, τον **Προηγούμενο (Previous)** και **Επόμενο (Next)** σύνδεσμο αν υπάρχουν (για λόγους σύνδεσης μεταξύ τους) καθώς και ένα **Εναλλακτικό (Alternative)** κείμενο για να εμφανίζεται όταν δεν μπορεί για οποιοδήποτε λόγο να εμφανιστεί το γραφικό.

Εδώ επίσης ορίζουμε αν θέλουμε, να είναι προστατευμένα (να μην μπορούν να τροποποιηθούν) τα **Περιεχόμενα (Contents)**, το **Μέγεθος (Size)** και η **Θέση(Position)** του γραφικού. Τέλος, μπορούμε να ορίσουμε αν το γραφικό θα εμφανίζεται σε τυχόν **Εκτύπωση (Print)** του εγγράφου ή όχι.

12.2)Γραφικό (Graphics)

Η καρτέλα **Γραφικό (Graphics)** μας επιτρέπει να ορίσουμε αν θέλουμε το γραφικό που έχουμε εισάγει να εμφανίζεται **Αντεστραμμένο (Flip)**, τόσο **Οριζόντια(Horizontally)** όσο και **Κατακόρυφα (Vertically)**. Την οριζόντια αντιστροφή μπορούμε να την ενεργοποιήσουμε αν θέλουμε μόνο για τις ζυγές ή μονές σελίδες, ή για όλες.



13) Γραμμές αντικειμένων

Μόλις επιλέξουμε ένα γραφικό, ενεργοποιούνται οι γραμμές γραφικών αντικειμένων και αντικειμένων πλαισίων. Από αυτές μπορούμε να κάνουμε κι άλλες εργασίες οι οποίες μπορεί να υπάρχουν ή και όχι στους διαλόγους ελέγχου. Αναλυτικά:

13.1) Γραμμή γραφικών αντικειμένων

Η γραμμή γραφικών αντικειμένων μας δίνει τη δυνατότητα να τροποποιήσουμε την εμφάνιση του αντικειμένου που έχουμε εισάγει. Συγκεκριμένα, μπορούμε να τροποποιήσουμε το ποσοστό κάθε ενός από τα βασικά χρώματα (Κόκκινο, Πράσινο, Μπλε) που θα έχει το αντικείμενό μας, τη **Φωτεινότητα (Brightness)** και το **Κοντράστ (Contrast)** της εικόνας, καθώς και τη **διαφάνεια** του αντικειμένου.

Μπορούμε επίσης να αναστρέψουμε την εικόνα, τόσο οριζόντια όσο και κατακόρυφα.

Εικόνα 32: Τμήμα γραμμής γραφικών αντικειμένων

Αν θέλουμε να έχουμε μία εικόνα που θα εμφανίζεται ως Υδατογράφημα (Watermark) πίσω από το κείμενό μας σε μία σελίδα, μπορούμε να την τοποθετήσουμε, να ορίσουμε τη διαφάνειά της σε ένα ποσοστό που να μην «μπερδεύεται» με το κείμενο μπροστά (π.χ. 80%) και να ορίσουμε να βρίσκεται Σε φόντο (από την καρτέλα Αναδίπλωση).

Εκτός από αυτές τις τροποποιήσεις, μπορούμε να εφαρμόσουμε και ειδικά εφέ, μία λειτουργία που θα αναλύσουμε στο τμήμα του Draw.

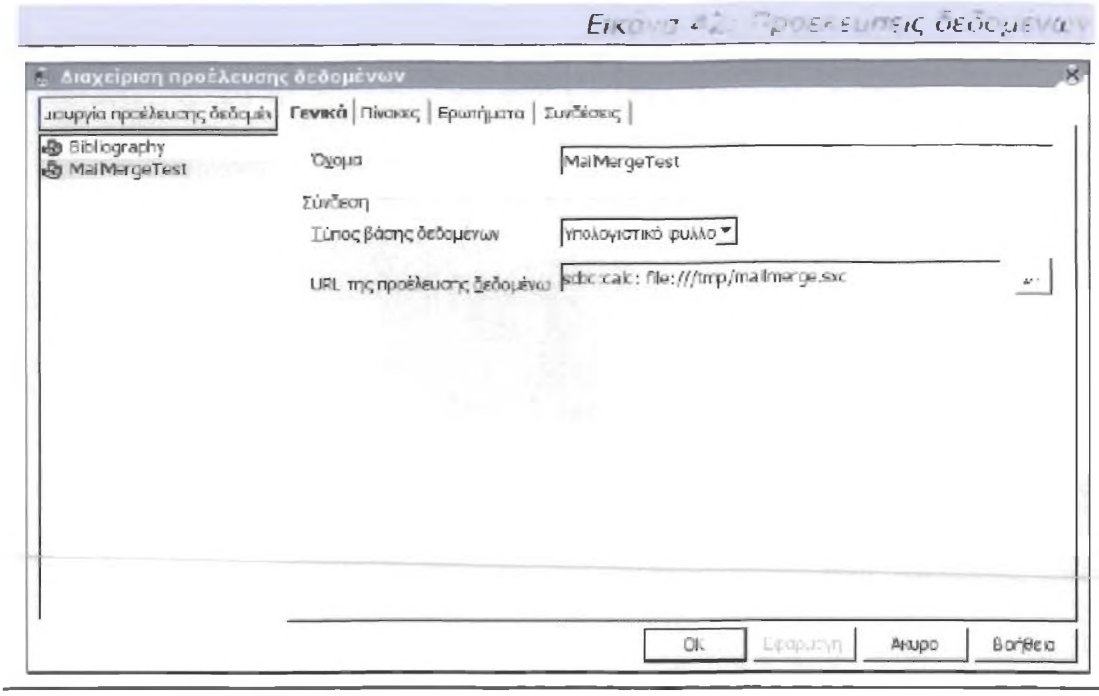
13.2) Γραμμή αντικειμένων πλαισίων

Η γραμμή αντικειμένων πλαισίων μας δίνει τη δυνατότητα να τροποποιήσουμε τα χαρακτηριστικά του πλαισίου (ή ενός νοητού πλαισίου που περιβάλλει την εικόνα). Συγκεκριμένα, μπορούμε να ορίσουμε γρήγορα την **αναδίπλωση**, τη **στοίχιση**, το **χρώμα** και το **είδος του περιγράμματος**, την **αγκύρωση** και τη **θέση** του αντικειμένου.

Εικόνα 33: Τμήμα γραμμής αντικειμένων πλαισίου

6.8.3 Δημιουργία της προέλευσης δεδομένων

Για να δημιουργήσετε μία **Εγκύκλιο Επιστολή (Mail Merge)** στο OpenOffice.org, πρέπει να μετατρέψετε το φύλλο υπολογισμού σας (spreadsheet)σε μία προέλευση δεδομένων (data source), έτσι ώστε να είναι δυνατό να το βλέπετε μέσα στον περιηγητή προελεύσεων δεδομένων.
- Όταν έχετε ολοκληρώσει το φύλλο εργασίας σας με τα απαιτούμενα δεδομένα, πατήστε στο μενού **Αρχείο (File)** επιλογή **Αποθήκευση (Save)** για να το αποθηκεύσετε σε μορφή .sxc, στον κατάλογο που επιθυμείτε. Τώρα μπορείτε να κλείσετε το φύλλο υπολογισμού.
Στο μενού **Εργαλεία (Tools)**, επιλέξτε **Προελεύσεις Δεδομένων (Datasources)**.



Στα αριστερά, βλέπετε τις προελεύσεις δεδομένων που έχουν ήδη δημιουργηθεί και καταχωρηθεί από το **Διαχειριστή Προελεύσεων Δεδομένων**. Χρειάζεται απλά να περιηγηθείτε μέχρι την τοποθεσία όπου αποθηκεύσατε το αρχείο με τη δική σας προέλευση δεδομένων, και να την επιλέξετε.

- Πατήστε στο κουμπί **Δημιουργία Προέλευσης Δεδομένων (New Data Source)**.
- Στα δεξιά, στο πεδίο **Όνομα (Name)** μπορείτε να γράψετε ένα όνομα για αυτή την προέλευση.

- Στο πεδίο **Σύνδεση (Connection)** επιλέξτε τον **Τύπο βάσης δεδομένων (Database Type)** που θα χρησιμοποιήσετε ως προέλευση δεδομένων, στη συγκεκριμένη περίπτωση, από την αναπτυσσόμενη λίστα, επιλέξτε **Υπολογιστικό φύλλο (Spreadsheet)**.
- Στην καρτέλα **Πίνακες (Tables)**, επιβεβαιώστε ότι έχει επιλεγεί το όνομα του φύλλου υπολογισμού.
- Πατήστε το Ok και το φύλλο σας θα μπορεί να χρησιμοποιηθεί για μία **εγκύκλιο επιστολή (Mail Merge)**.

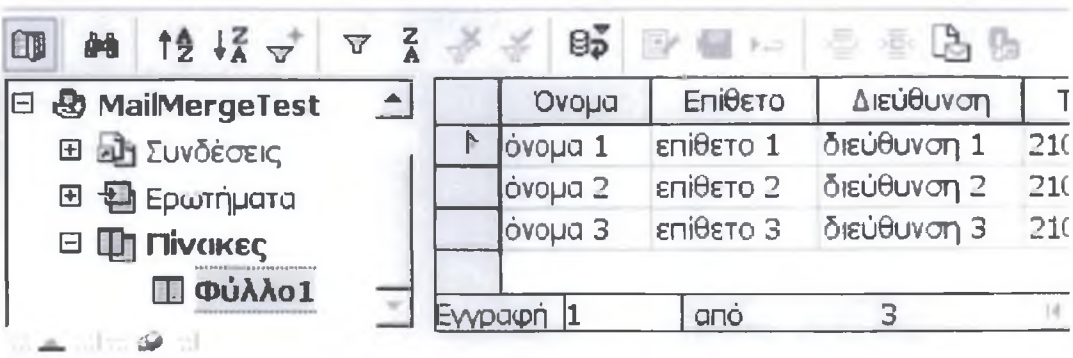
Α) Δημιουργία της Πρότυπης Επιστολής

Ανοίξτε ένα νέο έγγραφο κειμένου, χρησιμοποιώντας για παράδειγμα το μενού **Αρχείο (File)**, επιλογή **Δημιουργία (New)**, **Έγγραφο κειμένου (Text Document)**.

- Γράψτε τα περιεχόμενα της επιστολή σας. Δεν χρειάζεται να εισάγετε καθόλου πεδία διευθύνσεων για την ώρα. Μπορείτε να χρησιμοποιήσετε ένα αστερίσκο (*), σαν σύμβολο κράτησης θέσης, για να βοηθηθείτε όταν έχετε ολοκληρώσει την επιστολή σας.
- Αφού έχετε ολοκληρώσει την επιστολή σας, αποθηκεύστε τη αλλά μην κλείσετε το αρχείο.

- Από το μενού **Προβολή (View)** επιλέξτε **Προελεύσεις Δεδομένων (Data Sources)** ή πατήστε το πλήκτρο F4.

Εικόνα 43: Προβολή προελεύσεων δεδομένων



Οι καταχωρημένες προελεύσεις δεδομένων σας θα απεικονίζονται στον περιηγητή και απλά πρέπει να επιλέξετε την προέλευση με την οποία επιθυμείτε να εργαστείτε.

- Πατήστε στο μικρό σταυρό, δίπλα στο όνομα της προέλευσης δεδομένων, για να απεικονίσετε τη δομή των περιεχομένων της.
- Πατώντας στο όνομα ενός πίνακα, θα δείτε να απεικονίζονται τα πεδία του και οι εγγραφές του στο δεξί μέρος του *Περιηγητή*.

-Προσθέσετε τα πεδία στο έγγραφό σας. Πατήστε, και κρατήστε πατημένο το πεδίο Όνομα 1 (ή όπως αλλιώς ονομάζεται το πεδίο σας), του πίνακά σας. Κρατώντας πατημένο του κουμπί του ποντικιού, σύρετέ το μέχρι το σύμβολο κράτησης θέσης (αστερίσκο) στο έγγραφό σας και απελευθερώστε το. Το πεδίο θα απεικονιστεί στην περιοχή του εγγράφου σας.

-Επαναλάβετε αυτή την ενέργεια για το πεδίο Επίθετο 1. Αλλάξτε τη θέση γραμμής για να εισάγετε τα υπόλοιπα πεδία της διεύθυνσης.

Όταν έχετε εισάγει όλα τα πεδία σας, είστε έτοιμοι για τη συγχώνευση.

Β) Συγχώνευση

Επιλέξτε τις εγγραφές που χρειάζεστε. Αν επιθυμείτε να επιλέξετε όλες τις εγγραφές, πατήστε στο τετράγωνο στην πάνω αριστερή γωνία του πίνακα. Εναλλακτικά,αν θέλετε να επιλέξετε όλα τα περιεχόμενα ενός πεδίου, μπορείτε να πατήσετε στην επικεφαλίδα του πεδίου. Τέλος, αν επιθυμείτε να επιλέξετε όλα τα πεδία μίας εγγραφής, πατήστε στο γκρι παραλληλόγραμμο στα δεξιά της. Ένα μικρό βέλος θα εμφανιστεί και η εγγραφή θα επισημανθεί. Για να επιλέξετε εγγραφές που δεν είναι συνεχόμενες, πατήστε το πλήκτρο **CTRL** ενώ κάνετε τις επιλογές σας.

Τώρα μπορείτε να κάνετε τη συγχώνευση. Στη γραμμή εργαλείων του *Περιηγητή*, το προτελευταίο εικονίδιο εμφανίζει κάτι επιστολές. Πατήστε σε αυτό για να προσπελάσετε τα εργαλεία ταξινόμησης.

Εικόνα 44: Γραμμή εργαλείων περιηγητή δεδομένων

Ταξινομήστε τις εγγραφές, όπως τις έχετε επιλέξει, στην προηγούμενη διαδικασία, στη συνέχεια επιλέξτε τη συσκευή στην οποία θα παραχθεί η συγχώνευση. Αν επιλέξετε **Αρχείο (File)**, θα πρέπει να διαλέξετε τη διαδρομή του αρχείου πατώντας στο κουμπί με τις τρεις τελείες στα δεξιά του πεδίου **Διαδρομή (Path)**,έτσι ώστε να μπορείτε να περιηγηθείτε στο σύστημα αρχείων μέχρι τον κατάλογο στον οποίο επιθυμείτε να δημιουργηθούν τα αρχεία. Για κάθε μία από τις επιλεγμένες εγγραφές, θα δημιουργηθεί ένα ξεχωριστό αρχείο. Μπορείτε να ονομάσετε αυτό το αρχείο βασιζόμενοι σε οποιοδήποτε όνομα πεδίου της προέλευσης δεδομένων.

Εικόνα 45: Δημιουργία εγκυκλιων επιστολών