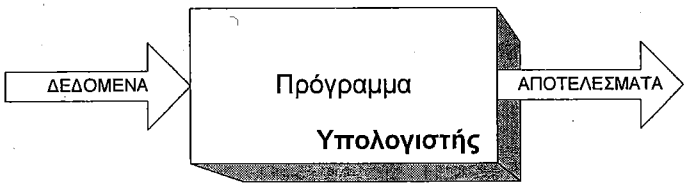
Αρχιτεκτονική του Υπολογιστή

Εισαγωγή

Ο υπολογιστής είναι μια συσκευή που επεξεργάζεται δεδομένα εκτελώντας μια σειρά **εντολών** που ονομάζουμε **πρόγραμμα**.



Σχήμα 3.1.1: Ο υπολογιστής

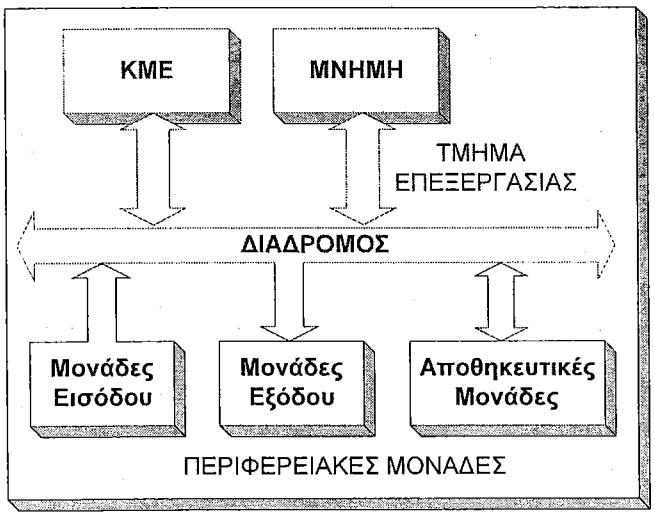
Η Αρχιτεκτονική ενός υπολογιστή

Το τμήμα επεξεργασίας, αποτελείται:

1. από τη **μνήμη (memory)**, στην οποία αποθηκεύονται το πρόγραμμα που θέλουμε να εκτελέσουμε και τα δεδομένα που θα επεξεργαστεί το πρόγραμμα
2. από την **Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ) (Central Processing Unit – CPU)**, που εκτελεί τις εντολές του προγράμματος
3. και από το **διάδρομο (Bus)** που επιτρέπει τη διακίνηση των δεδομένων. Συγκεκριμένα χρησιμεύει για την επικοινωνία της ΚΜΕ και της μνήμης, καθώς και την επικοινωνία τους με τις περιφερειακές μονάδες

Οι περιφερειακές μονάδες διακρίνονται σε τρεις κυρίως κατηγορίες, ανάλογα με τη λειτουργία που επιτελούν.

1. Οι περιφερειακές μονάδες που τροφοδοτούν με δεδομένα έναν υπολογιστή λέγονται **μονάδες εισόδου** και είναι για παράδειγμα το πληκτρολόγιο και το ποντίκι.
2. Οι περιφερειακές μονάδες που παίρνουν δεδομένα από ένα υπολογιστή και τα απεικονίζουν με κάποια μορφή (γράμματα, εικόνα, ήχος) ονομάζονται **μονάδες εξόδου.** Τέτοιες είναι οι οθόνες, οι εκτυπωτές και τα ηχεία.
3. Τέλος, οι περιφερειακές μονάδες στις οποίες υπάρχουν αποθηκευμένα προγράμματα ή δεδομένα αποτελούν τις **αποθηκευτικές μονάδες** του υπολογιστή. Τέτοιες είναι οι σκληροί δίσκοι, οι δισκέτες και οι οπτικοί δίσκοι CD.



Σχήμα 3.1.4: Η δομή ενός υπολογιστή

ΒΙΒΛΙΟΓΡΑΦΙΑ

"Peter Norton-Εισαγωγή στους υπολογιστές", Ελληνική μετάφραση, Εκδόσεις Α.Τζιόλα
"The Architecture of Computer Hardware and System Software - An IT Approach" Irv Englander, Wiley 1996, ISBN 0-471-31037-9.
"Αρχιτεκτονική υπολογιστών - Software - Hardware", Thom Luce, Ελληνική μετάφραση, Εκδόσεις Α.Τζιόλα
"Μικροεπεξεργαστές, Θεωρία και εφαρμογές", Gilmore, Ελληνική μετάφραση, Εκδόσεις Α.Τζιόλα
"Στοιχεία υπολογιστικών συστημάτων", Ε.Παπαθανασίου, Εκδόσεις Μπένου.
"Οργάνωση και Αρχιτεκτονική Υπολογιστών", William Stallings, Εκδόσεις ΤΖΙΩΛΑ (σε Ελληνική μετάφραση).

Πίνακας Περιεχομένων

Κεντρική Μονάδα Επεξεργασίας

Εισαγωγή

Ο υπολογιστής επεξεργάζεται δεδομένα ακολουθώντας βήμα – βήμα, τις εντολές ενός προγράμματος.

Το τμήμα του υπολογιστή, που εκτελεί τις εντολές και συντονίζει όλες τις λειτουργίες, είναι η **κεντρική μονάδα επεξεργασίας (ΚΜΕ)**.

Η ΚΜΕ είναι ένα πολύπλοκο λογικό κύκλωμα, σχεδιασμένο να διαβάζει εντολές από τη μνήμη και να τις εκτελεί.

Σήμερα οι περισσότερες ΚΜΕ κατασκευάζονται στη μορφή ενός ολοκληρωμένου κυκλώματος. Στο ολοκληρωμένο κύκλωμα, που περιέχει την ΚΜΕ, ενσωματώνονται συχνά και άλλα βοηθητικά κυκλώματα, για τα οποία θα μιλήσουμε σε παρακάτω κεφάλαια. Το ολοκληρωμένο αυτό κύκλωμα έχει επικρατήσει να το ονομάζουμε **επεξεργαστή** (processor) ή και **μικροεπεξεργαστή** (microprocessor).

Σχήμα 3.2.1 Ολοκληρωμένος μικροεπεξεργαστής 486 DX

Μόλις τροφοδοτήσουμε την ΚΜΕ με τάση, αυτή θα ξεκινήσει την εκτέλεση του προγράμματος από μια συγκεκριμένη διεύθυνση στη μνήμη, που θεωρείται η αρχή του προγράμματος. Η ΚΜΕ θα διαβάσει την πρώτη εντολή από την μνήμη και στη συνέχεια θα την εκτελέσει. Όμοια θα συνεχίσει με την δεύτερη εντολή, την τρίτη κ.ο.κ. Όπως έχουμε ήδη αναφέρει, η ανάγνωση μιας εντολής από την μνήμη αποτελεί την **φάση ανάκλησης** (fetch cycle) της εντολής. Την φάση αυτή ακολουθεί η **φάση εκτέλεσης** (execution cycle) της εντολής.

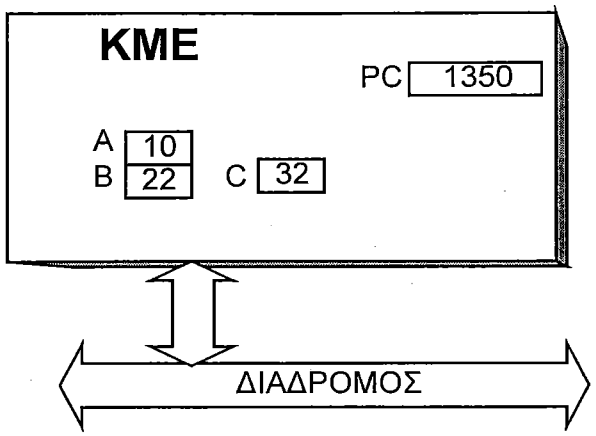
Καταλήγουμε λοιπόν ότι, όταν ένας υπολογιστής δουλεύει, η ΚΜΕ συνεχώς ανακαλεί εντολές από τη μνήμη και τις εκτελεί.

Καταχωρητές

Ας θυμηθούμε το παράδειγμα της απλής αριθμομηχανής (σχ. 3.1.4), που πρόσθετε οποιοδήποτε πλήκτρο πατάγαμε με το ήδη υπάρχον άθροισμα. Η ΚΜΕ λαμβάνει δεδομένα από το πληκτρολόγιο και εκτελεί συνεχώς την πράξη της πρόσθεσης του αριθμού που πατάμε με το ήδη υπάρχον άθροισμα.

Τι γίνονται τα δεδομένα που διαβάζει η κεντρική μονάδα επεξεργασίας; Πού αποθηκεύονται για να επεξεργαστούν; Πού αποθηκεύεται το αποτέλεσμα της πρόσθεσης;

Τα δεδομένα που διαβάζει από το πληκτρολόγιο, η ΚΜΕ τα αποθηκεύει στους **καταχωρητές** της (registers). Οι καταχωρητές είναι θέσεις μνήμης μέσα στην ΚΜΕ, που χρησιμοποιούνται για την προσωρινή αποθήκευση και την επεξεργασία των δεδομένων. Κάθε καταχωρητής έχει ένα συγκεκριμένο όνομα που τον χαρακτηρίζει.



Σχήμα 3.2.2: Οι καταχωρητές της ΚΜΕ

Στο σχήμα 3.2.2 βλέπουμε μια ΚΜΕ με τέσσερις καταχωρητές. Τα ονόματα των καταχωρητών αυτών είναι ***A***, ***B***, ***C*** και ***PC*** αντίστοιχα. Στο σχήμα βλέπουμε επίσης και το περιεχόμενο του κάθε καταχωρητή. Για παράδειγμα το περιεχόμενο του καταχωρητή ***A*** είναι $10_{16}=16_{10}$.

Κάθε φορά που θέλουμε να εκτελέσουμε κάποια αριθμητική ή λογική πράξη μεταξύ δύο αριθμών, θα πρέπει πρώτα να μεταφερθεί κάθε ένας από τους δύο αριθμούς σε ένα καταχωρητή της ΚΜΕ. Ο λόγος που πρέπει να γίνει η μεταφορά αυτή είναι ότι η ΚΜΕ μπορεί να κάνει αριθμητικές ή λογικές πράξεις **μόνο** μεταξύ των δεδομένων που περιέχουν οι καταχωρητές της. Τα δεδομένα που επεξεργάζεται η ΚΜΕ θα πρέπει να αποθηκεύονται μέσα σε κάποιο καταχωρητή της έστω και προσωρινά.

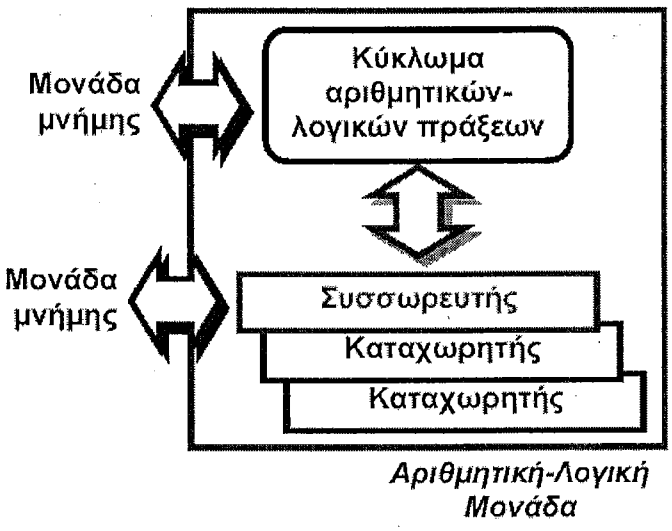
Οι καταχωρητές διακρίνονται σε καταχωρητές **γενικού σκοπού** (general purpose registers - GPR) και καταχωρητές **ειδικού σκοπού** (special purpose registers - SPR).

- Οι καταχωρητές γενικού σκοπού χρησιμοποιούνται μόνο για την αποθήκευση και επεξεργασία των δεδομένων της ΚΜΕ.
- Οι καταχωρητές ειδικού σκοπού, εκτός από αποθηκευτικοί χώροι, είναι συνυφασμένοι με μια λειτουργία της ΚΜΕ. Για παράδειγμα, ένας καταχωρητής ειδικού σκοπού είναι ο μετρητής προγράμματος, ***PC***. Ο μετρητής προγράμματος φυλάει τη διεύθυνση της επόμενης εντολής του προγράμματος. Με βάση την τιμή του καταχωρητή γίνεται η ανάκληση μιας εντολής από τη μνήμη.

Accumulators - Συσσωρευτές

Κάθε ΚΜΕ (CPU) έχει τουλάχιστον ένα καταχωρητή, για την αποθήκευση του **αποτελέσματος** κάθε αριθμητικής ή λογικής πράξης.

Οι καταχωρητές αυτοί ονομάζονται Accumulators - Συσσωρευτές.

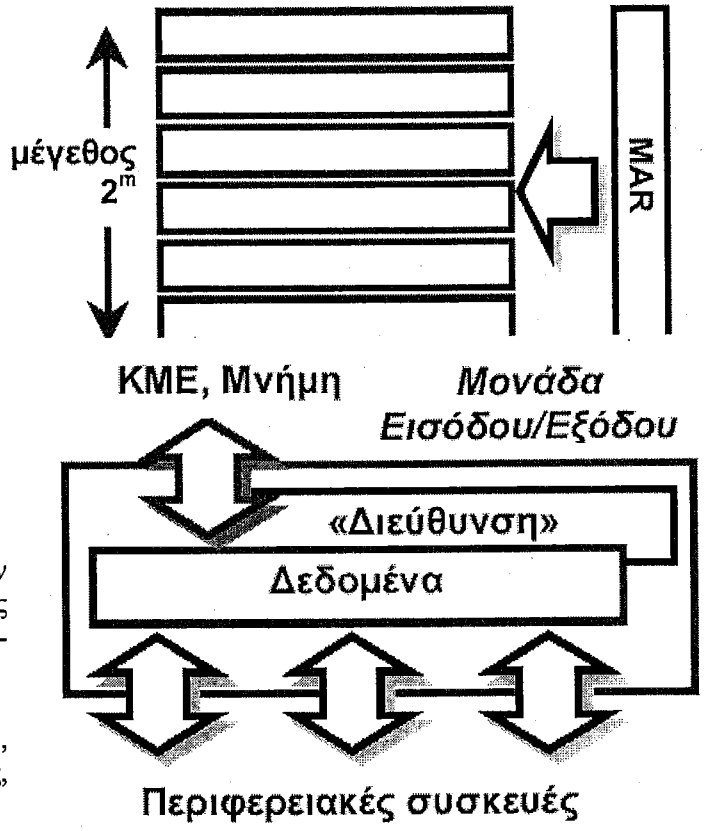


General purpose Registers - Καταχωρητές γενικής χρήσης

Κάθε ΚΜΕ (CPU) έχει αριθμό καταχωρητών γενικού σκοπού χρησιμοποιούνται μόνο για την αποθήκευση και επεξεργασία των δεδομένων.

Data Counter (DC), or Memory Address Register (MAR), or Data Pointer

Κρατά την **διεύθυνση** της λέξης **δεδομένων** που πρόκειται να διαβαστεί από τη μνήμη ή να γραφεί σε αυτή.



Memory Data Register - MDR

Ο καταχωρητής δεδομένων της μνήμης χρησιμοποιείται για την προσωρινή αποθήκευση των **δεδομένων** που διαβάζονται ή γράφονται στη μνήμη.

Σε σύγχρονους υπολογιστές θα βρούμε αυτούς τους δύο καταχωρητές να υπάρχουν **και** στα κυκλώματα ελέγχου των διαφόρων συσκευών (μονάδες εισόδου / εξόδου - interface).

Οι ελεγκτές αυτοί παρέχουν την δυνατότητα της άμεσης προσπέλασης της μνήμης (Direct Memory Addressing - DMA)

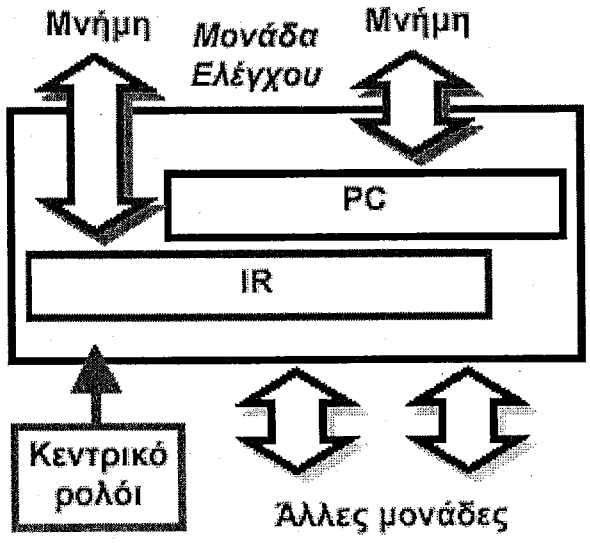
Η μονάδα ελέγχου (Control Unit), διαθέτει δύο (τουλάχιστον) καταχωρητές, τους:

Καταχωρητής Εντολών (Instruction register - IR)

Δέχεται μια από τις εντολές του προγράμματος από τη μνήμη, για να αναγνωριστεί / αποκωδικοποιηθεί, να αναλυθεί σε επιμέρους εργασίες και τέλος να εκτελεσθεί.

Μετρητής προγράμματος - ή μετρητής εντολών (program counter - PC)

Το περιεχόμενό του δίνει κάθε φορά τη διεύθυνση της μνήμης στην οποία υπάρχει **η επόμενη εντολή** του προγράμματος που θα εκτελεσθεί.



Η εντολή αυτή θα μεταφερθεί πρώτα στον καταχωρητή **IR**, θα αποκωδικοποιηθεί και ανάλογα με την εντολή ο PC, θα πάρει την διεύθυνση της επόμενης εντολής.

Δείκτης Στοίβας - Stack Pointer (SP)

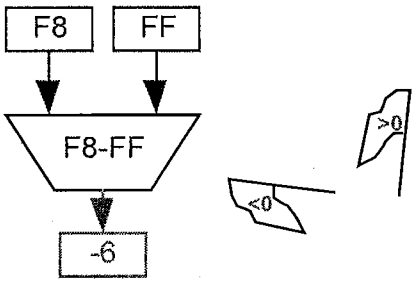
Όταν ο υπολογιστής εκτελεί ένα συγκεκριμένο πρόγραμμα, πολλές φορές αναγκάζεται ή κατόπιν εντολής του προγράμματος ή κατόπιν επίδρασης από κάποιο εξωτερικό συμβάν (**διακοπή - interrupt**), να παρεκλίνει της κανονικής ροής του προγράμματος ώστε να εκτελέσει κάποιο άλλο. Όταν αυτό το δεύτερο πρόγραμμα τελειώσει, ο υπολογιστής θα πρέπει να γνωρίζει που διέκοψε (**interrupt**) ώστε να συνεχίσει. Για το λόγο αυτό ένα μέρος της μνήμης, χρησιμοποιείται για να αποθηκεύει την κατάσταση των καταχωρητών όταν έγινε αυτή η διακοπή. Ο χώρος αυτός λέγεται **στοίβα - stack**.

Ο **stack pointer** είναι ένας ειδικός καταχωρητής που δείχνει πάντα στην κορυφή της στοίβας.

Flag Register

Η ALU περιέχει ειδικές θέσεις μνήμης, στις οποίες κρατά πληροφορίες για το αποτέλεσμα των πράξεων που εκτελεί, όπως για παράδειγμα εάν το αποτέλεσμα της πράξης είναι μεγαλύτερο (θετικό) ή μικρότερο (αρνητικό) από το μηδέν. Οι θέσεις αυτές ονομάζονται **σημαίες (flags)**. Κάθε φορά που εκτελείται μια αριθμητική ή λογική πράξη οι πληροφορίες για το αποτέλεσμα των πράξεων αποθηκεύονται στις αντίστοιχες σημαίες και τότε λέμε ότι οι σημαίες **ενημερώθηκαν**.

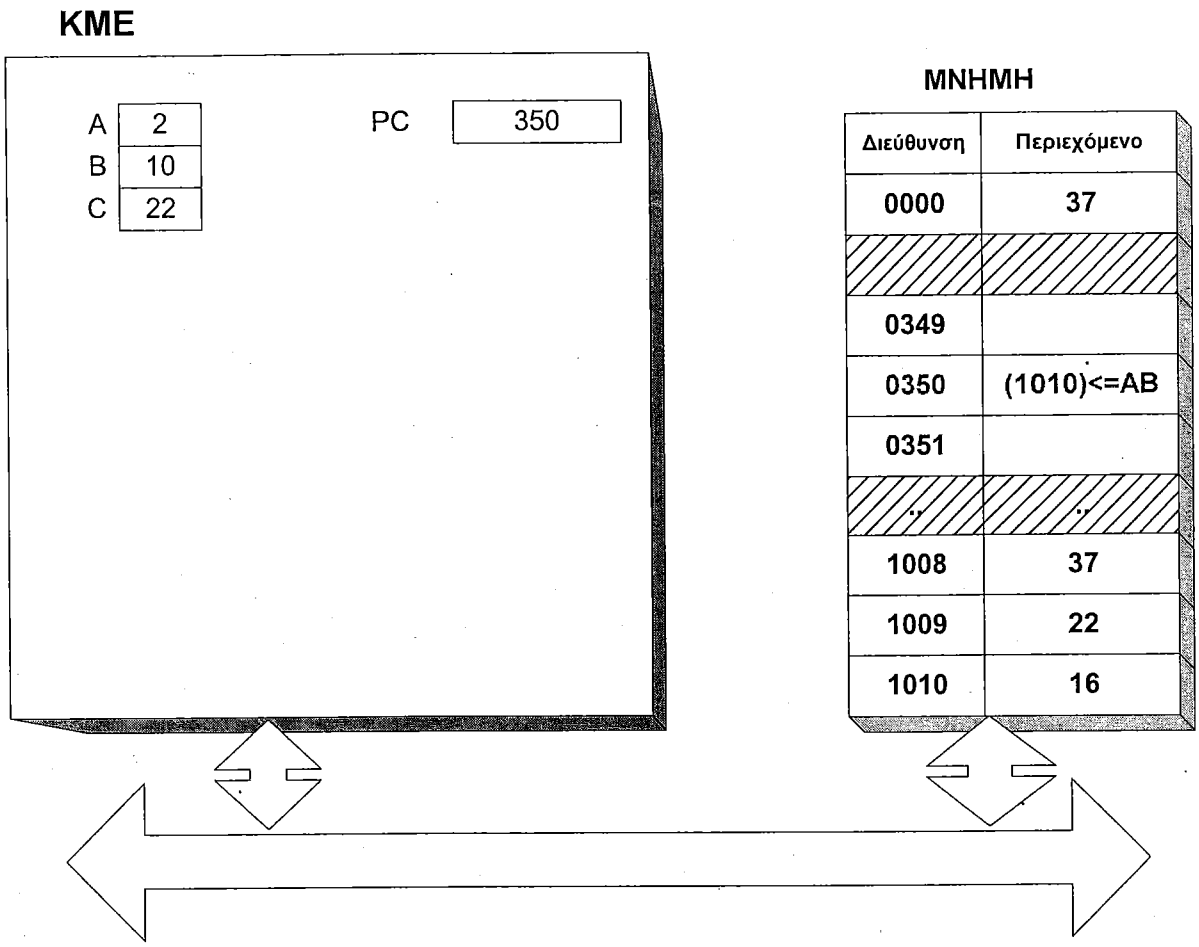
Παράδειγμα:
έστω ότι η ΚΜΕ (CPU) εκτελεί την αφαίρεση 248-254.
Το αποτέλεσμα αυτής της πράξης είναι ο αριθμός –6 ο οποίος είναι μικρότερος του μηδενός.
Η εκτέλεση αυτή της εντολής έχει ως αποτέλεσμα η σημαία του προσήμου να δείχνει ότι έχουμε αρνητικό αποτέλεσμα.



Η ALU εκτελεί την αφαίρεση 248-254

Αρχιτεκτονική της ΚΜΕ

Η ΚΜΕ χωρίζεται, σε διαφορετικά τμήματα. Κάθε τμήμα είναι υπεύθυνο για μια σειρά λειτουργιών. Θα προσπαθήσουμε να δούμε τα διαφορετικά τμήματα της ΚΜΕ περιγράφοντας την εκτέλεση μιας εντολής.



Σχήμα 3.2.3: Η κατάσταση της ΚΜΕ

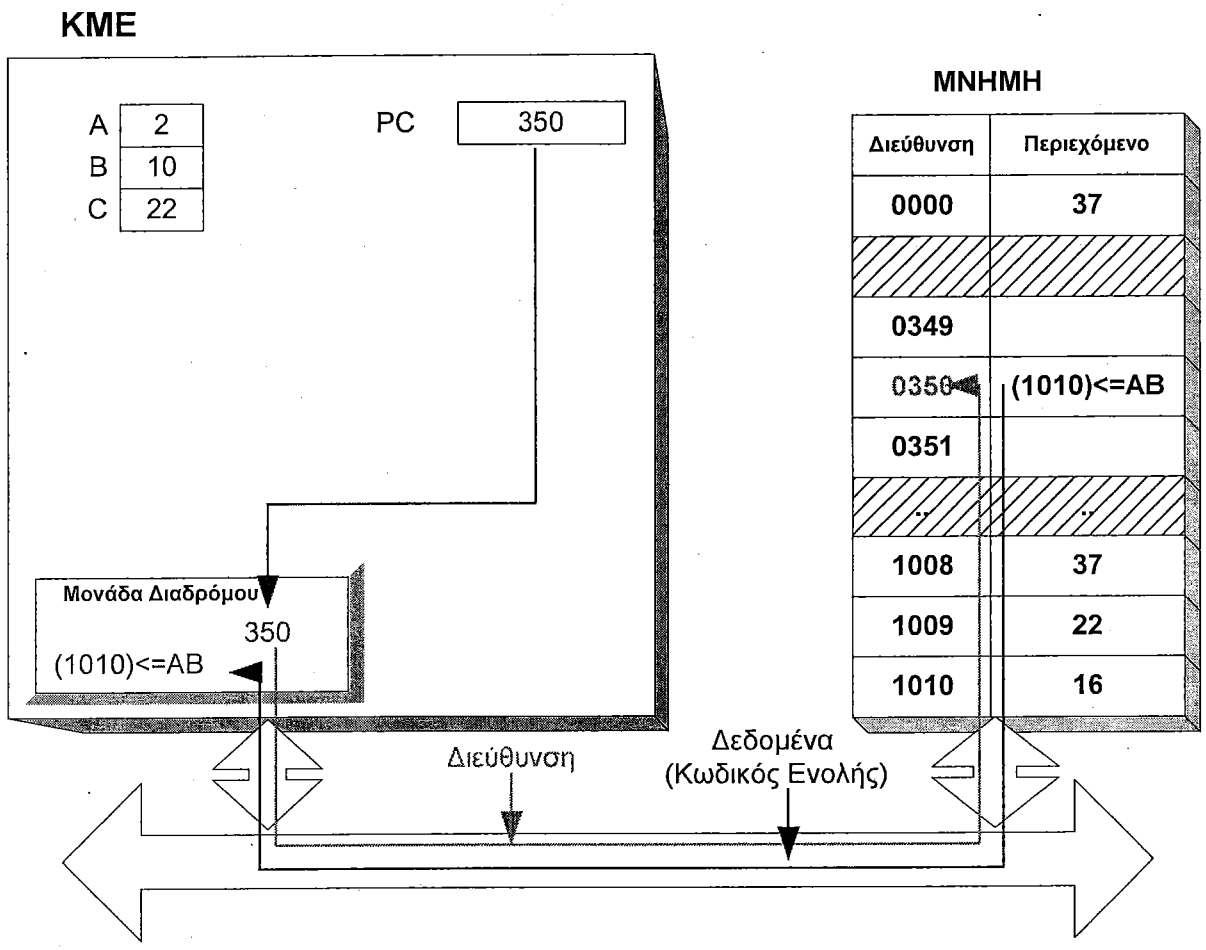
Στο παραπάνω σχήμα βλέπουμε ότι στους καταχωρητές ***A*** και ***B*** της ΚΜΕ έχουν αποθηκευτεί οι τιμές «2» και «10» αντίστοιχα. Η εντολή που ακολουθεί στο πρόγραμμα, είναι του πολλαπλασιασμού των καταχωρητών ***A*** και ***B*** και της αποθήκευσης του αποτελέσματος στη θέση μνήμης με διεύθυνση 1010. Την εντολή αυτή, την παριστάνουμε στο σχήμα ως (1010)←***A•B***, όπου (1010) είναι η θέση της μνήμης με διεύθυνση 1010, ενώ ***A*** και ***B*** οι αντίστοιχοι καταχωρητές της ΚΜΕ. Με το σύμβολο ← δηλώνουμε ότι το αποτέλεσμα της παράστασης ***A•B*** θα αποθηκευτεί στην θέση μνήμης με διεύθυνση 1010. Παρατηρούμε επιπλέον στο σχήμα ότι ο μετρητής προγράμματος, ***PC***, έχει την τιμή της διεύθυνσης που είναι αποθηκευμένη η εντολή που θέλουμε να εκτελέσουμε, δηλαδή αυτή του πολλαπλασιασμού.

Μονάδα Διαδρόμου (Bus Unit)

Όπως κάθε εντολή, έτσι και η εντολή του πολλαπλασιασμού πρέπει πρώτα να ανακληθεί (fetch) από τη μνήμη. Η κεντρική μονάδα επεξεργασίας θα πρέπει να διαβάσει από τη μνήμη το περιεχόμενο της διεύθυνσης 350, όπου βρίσκεται η εντολή του πολλαπλασιασμού.

Για το λόγο αυτό, η ΚΜΕ πρέπει να διαθέτει μια μονάδα, για να επικοινωνεί με τη μνήμη μέσω του διαδρόμου. Η μονάδα αυτή εμφανίζει με μορφή κατάλληλων ηλεκτρικών σημάτων, την διεύθυνση 350, που είναι αποθηκευμένη η εντολή του πολλαπλασιασμού και στη συνέχεια διαβάζει από το διάδρομο το περιεχόμενο της διεύθυνσης αυτής.

Η μονάδα αυτή ονομάζεται **μονάδα διαδρόμου.** Η μονάδα διαδρόμου παράγει όλα τα απαραίτητα ηλεκτρικά σήματα ώστε η ΚΜΕ να επικοινωνεί μέσω του διαδρόμου, με τη μνήμη ή τις περιφερειακές μονάδες.



Σχήμα 3.2.4: Μονάδα Διαδρόμου

Όπως βλέπουμε στο σχήμα 3.2.4, για να διαβάζει η μονάδα του διαδρόμου την επόμενη εντολή του προγράμματος από τη μνήμη, θα πρέπει πρώτα να γνωρίζει τη διεύθυνση της εντολής αυτής. Η διεύθυνση, όπως έχουμε ήδη αναφέρει, βρίσκεται στον μετρητή προγράμματος ***PC***. Αφού διαβάσει το περιεχόμενο του μετρητή προγράμματος, η μονάδα διαδρόμου εμφανίζει σε δυαδική παράσταση, με κατάλληλα σήματα, την διεύθυνση 350 πάνω στο διάδρομο. Η μνήμη απαντά με το περιεχόμενο της διεύθυνσης αυτής, που είναι η εντολή (1010)←***A•B***.

Όπως κάθε δεδομένο σε έναν υπολογιστή έτσι και κάθε εντολή παριστάνεται από ένα κατάλληλο δυαδικό κώδικα. Έτσι κάθε εντολή αντιστοιχεί σε ένα δυαδικό αριθμό, τον οποίο ονομάζουμε **κωδικό** ή **κώδικα** της εντολής. Η εντολή που είναι αποθηκευμένη στη μνήμη και που τη διαβάζουμε στη φάση ανάκλησης είναι ο αριθμός αυτός. Για παράδειγμα, όταν είμαστε στην φάση ανάκλησης της εντολής του πολλαπλασιασμού από τη μνήμη, μεταφέρεται μέσω του διαδρόμου ο κώδικας της εντολής αυτής από τη διεύθυνση 350 στην ΚΜΕ.

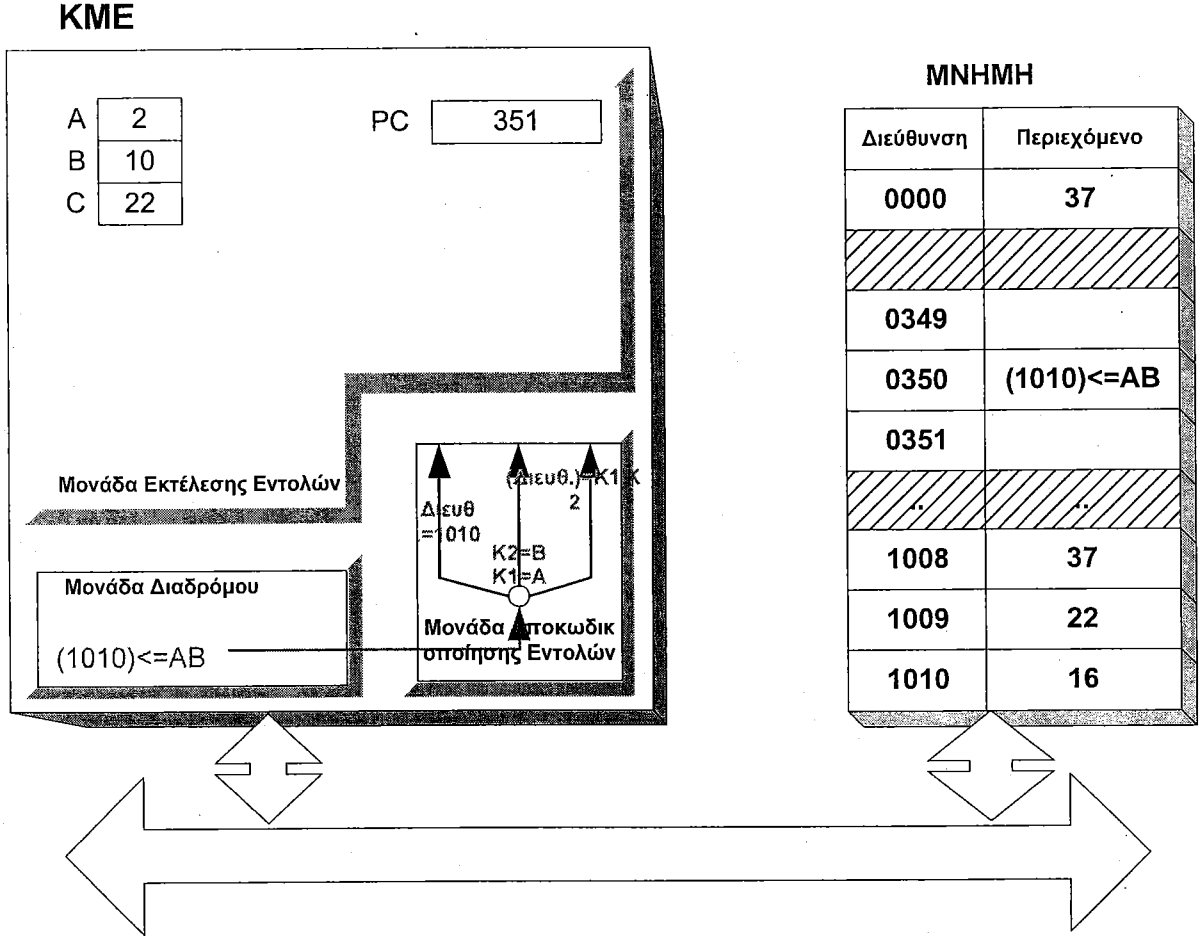
Παράλληλα η ΚΜΕ αυξάνει την τιμή του μετρητή προγράμματος ***PC***, ώστε αυτός να δείχνει την επόμενη προς εκτέλεση εντολή που βρίσκεται στη θέση 351.

Μονάδα αποκωδικοποίησης εντολών (Instruction unit)

Στη φάση της ανάκλησης ο κώδικας της εντολής οδηγείται από την μονάδα διαδρόμου στην **μονάδα αποκωδικοποίησης εντολών**.

Η μονάδα αυτή αναγνωρίζει ότι πρόκειται για εντολή πολλαπλασιασμού. Η εντολή (1010)←***A•B*** περιέχει επιπλέον πληροφορίες για το ποιοι καταχωρητές θα πάρουν μέρος στον πολλαπλασιασμό καθώς και για το πού θα αποθηκευτεί το αποτέλεσμα. Οι επιπλέον αυτές πληροφορίες ονομάζονται **ορίσματα** της εντολής. Η μονάδα αποκωδικοποίησης διαχωρίζει τις πληροφορίες που περιέχονται στην εντολή, δηλαδή το είδος της εντολής (πολλαπλασιασμός δύο καταχωρητών και αποθήκευση του αποτελέσματος στη μνήμη), και τα ορίσματα δηλαδή τους καταχωρητές που παίρνουν μέρος στον

πολλαπλασιασμό (*A*, *B*) αλλά και το πού θα αποθηκευτεί το αποτέλεσμα, και με κατάλληλα σήματα πληροφορεί την **μονάδα εκτέλεσης** για την εντολή που πρέπει να εκτελεσθεί.



Σχήμα 3.2.5:Μονάδα Αποκωδικοποίησης Εντολών

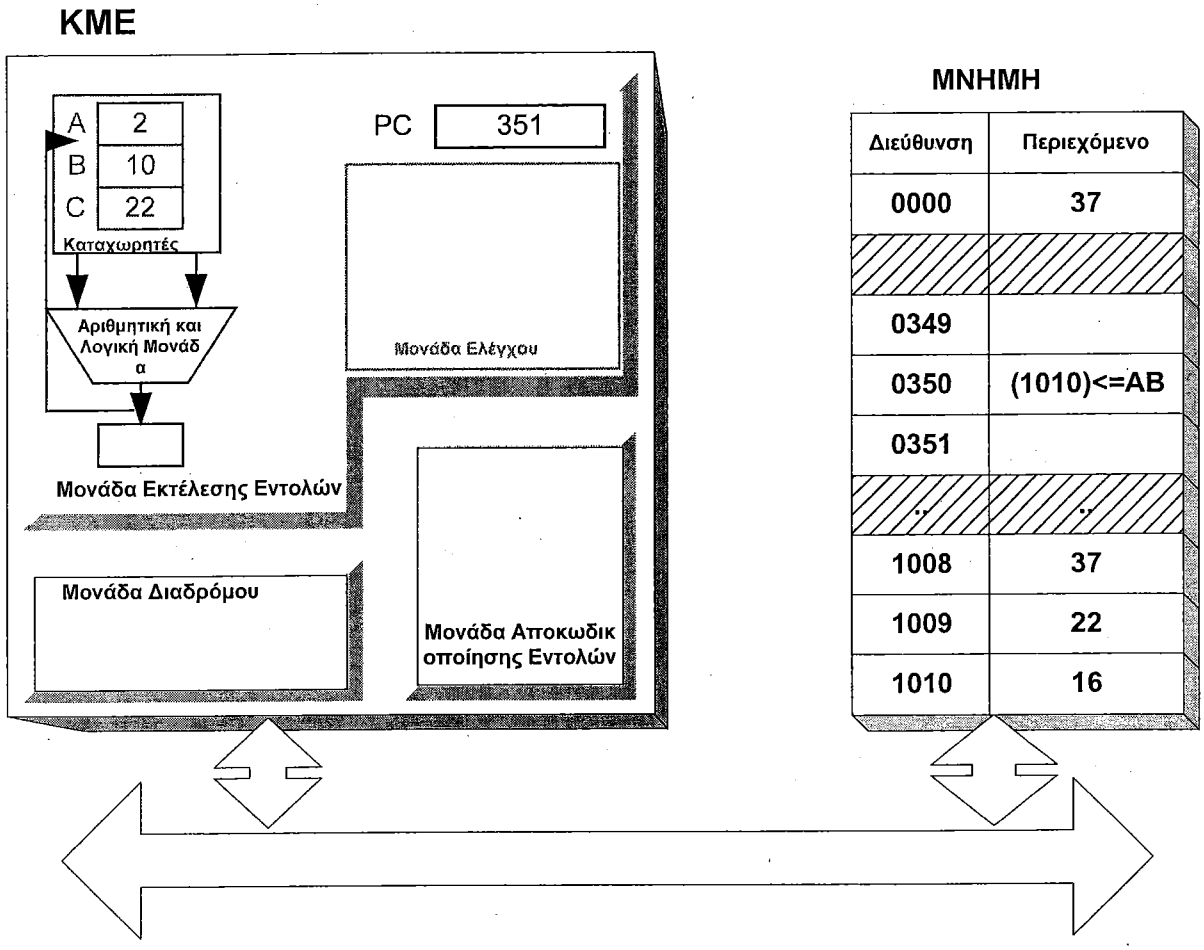
Στο σχήμα βλέπουμε ότι η μονάδα αποκωδικοποίησης των εντολών χωρίζει την εντολή στις επιμέρους πληροφορίες που περιέχει η εντολή που ανακλήθηκε από τη μνήμη. Οι πληροφορίες αυτές είναι:
Ποια πράξη θα εκτελεστεί. Εδώ έχουμε την αριθμητική πράξη του πολλαπλασιασμού μεταξύ δύο καταχωρητών και της αποθήκευσης του αποτελέσματος σε κάποια διεύθυνση της μνήμης. ((Διευθ.)←***Κατ1•Κατ2***)

Ποιοι καταχωρητές συμμετέχουν: Έδω έχουμε τους καταχωρητές ***A***, ***B*** (Κατ1=***A***, Κατ2=***B***). Και τέλος Πού θα πάει το αποτέλεσμα: Εδώ δίνεται η διεύθυνση της θέσης μνήμης όπου θα αποθηκευτεί το αποτέλεσμα. Η διεύθυνση είναι η 1010_{16}. (Διευθ.=1010)

Οι πληροφορίες αυτές περνούν, όπως θα δούμε στη συνέχεια στη μονάδα εκτέλεσης εντολών.

Μονάδα Εκτέλεσης (Execution Unit)

Η μονάδα εκτέλεσης εντολών αποτελείται από τρεις υπομονάδες: τους καταχωρητές, την αριθμητική και λογική μονάδα και την μονάδα ελέγχου.



Σχήμα 3.2.6: Η Μονάδα Εκτέλεσης Εντολών

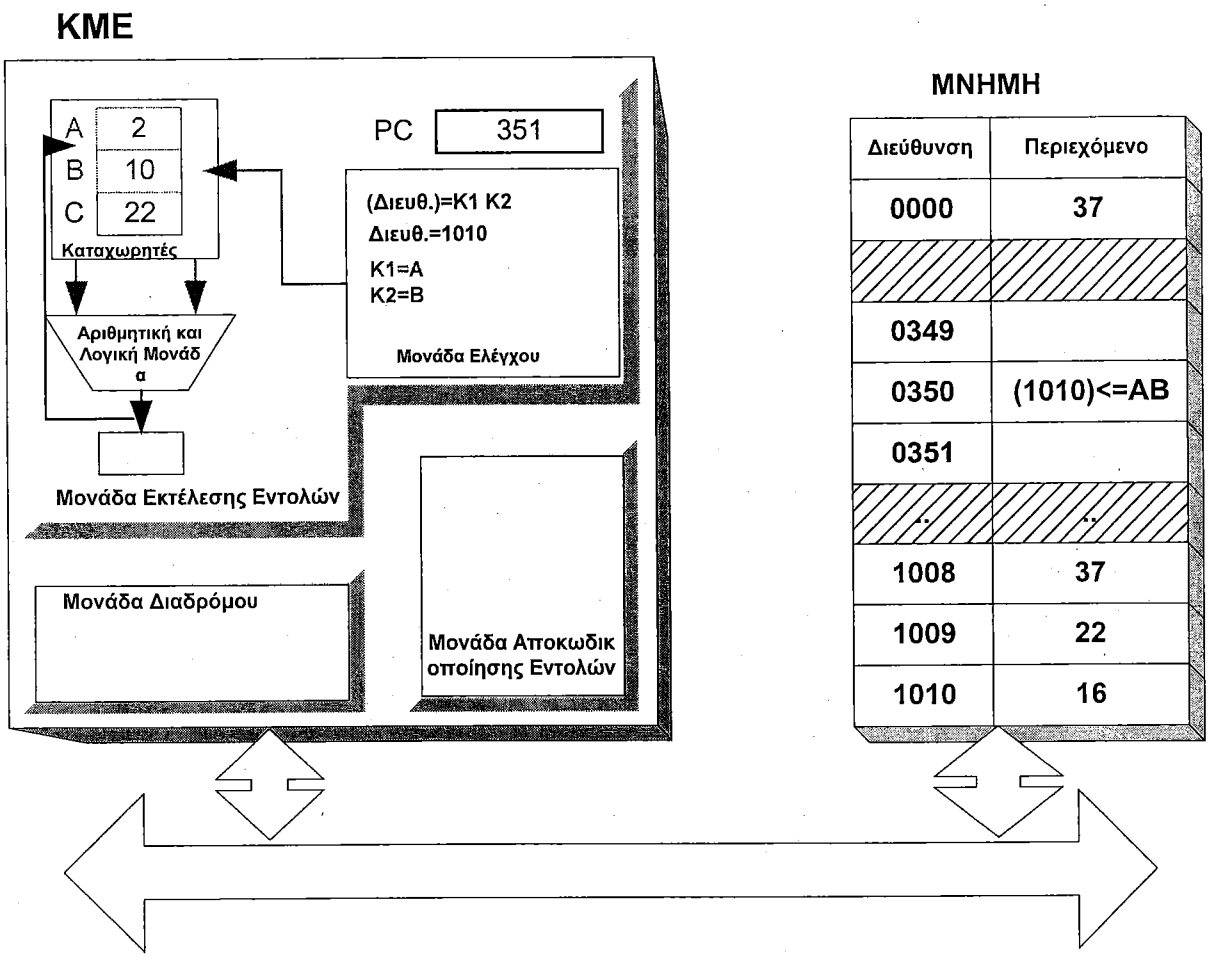
Αριθμητική και Λογική Μονάδα (Arithmetic and Logic Unit - ALU)

Η αριθμητική και λογική μονάδα είναι το σύνολο των κυκλωμάτων της ΚΜΕ που εκτελούν αριθμητικές και λογικές πράξεις μεταξύ των καταχωρητών της ΚΜΕ.

Ανάλογα με τον τύπο της ΚΜΕ, η αριθμητική και λογική μονάδα (ALU) μπορεί να εκτελεί πράξεις πρόσθεσης, αφαίρεσης, πολλαπλασιασμού και διαίρεσης ακεραίων αριθμών καθώς και τις λογικές πράξεις **Η** (OR), **ΚΑΙ** (AND), **ΟΧΙ** (NOT).

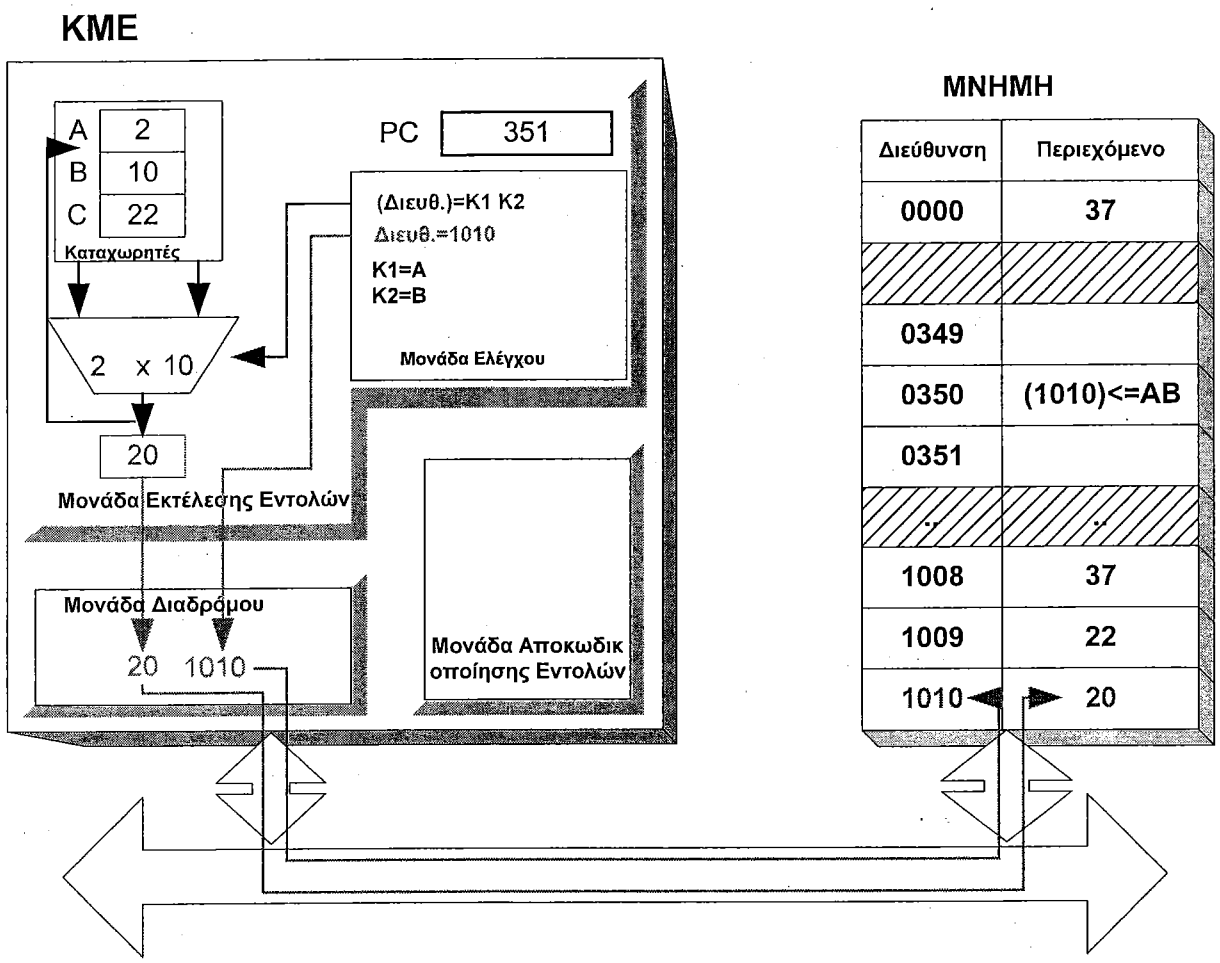
Μονάδα Ελέγχου (Control Unit)

Ο έλεγχος της λειτουργίας της αριθμητικής και λογικής μονάδας γίνεται από την μονάδα ελέγχου. Η μονάδα ελέγχου παραλαμβάνει τις επιμέρους πληροφορίες της εντολής από τη μονάδα αποκωδικοποίησης και ακολουθεί μια σειρά βημάτων για την εκτέλεση της.
Για παράδειγμα στην περίπτωση του πολλαπλασιασμού των καταχωρητών Α και Β η μονάδα ελέγχου θα κάνει τα ακόλουθα βήματα:



Σχήμα 3.2.9: Η εκτέλεση της εντολής

Αρχικά η μονάδα ελέγχου θα επιλέξει τους δύο καταχωρητές, Α και Β, που σύμφωνα με την εντολή θα πρέπει να οδηγηθούν στην είσοδο της αριθμητικής και λογικής μονάδας.



Σχήμα 3.2.10: Η ολοκλήρωση της εκτέλεσης της εντολής

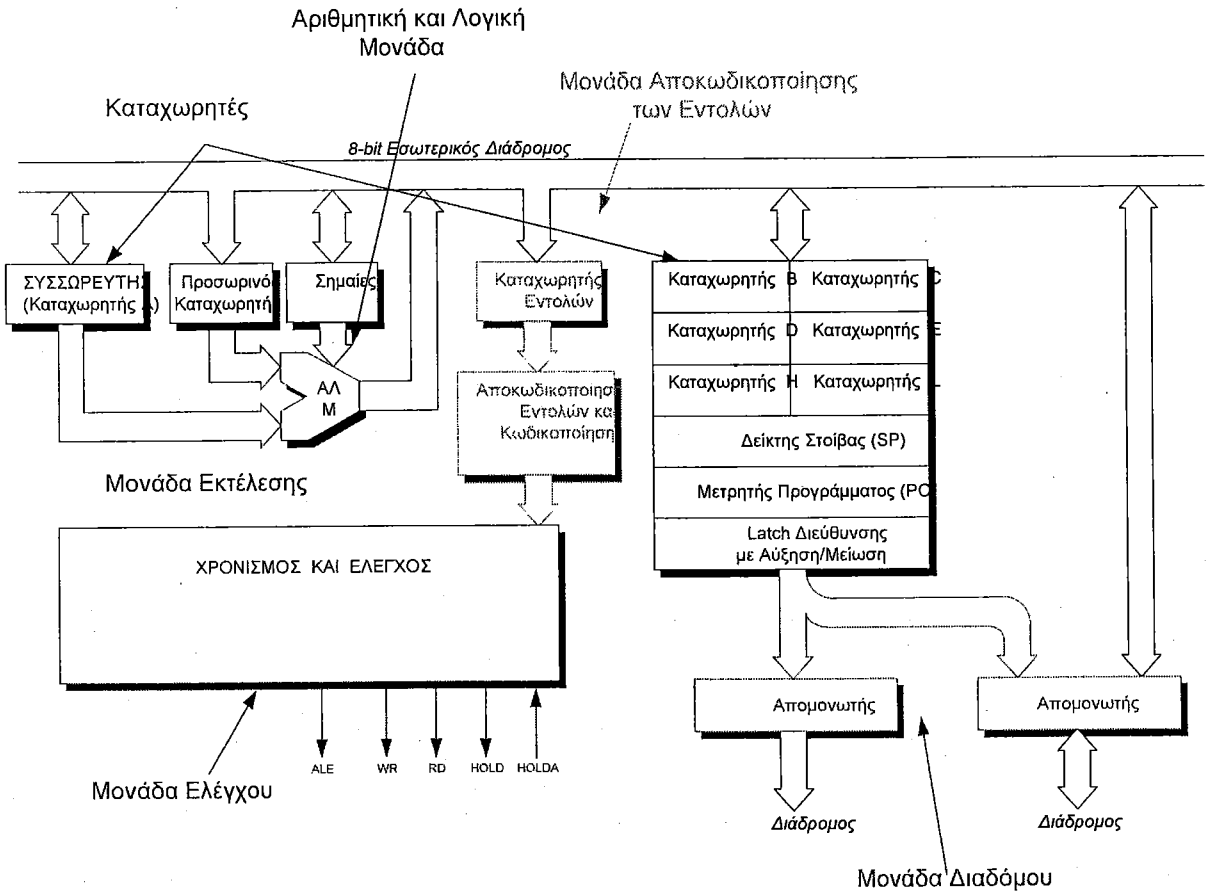
Θα ενεργοποιήσει την πράξη του πολλαπλασιασμού
Τέλος θα αποθηκεύσει το αποτέλεσμα σε κάποιο προσωρινό καταχωρητή. Στη συνέχεια θα το δώσει στη μονάδα διαδρόμου να το γράψει στη θέση μνήμης 1010.
Η μονάδα διαδρόμου γράφει το αποτέλεσμα στη μνήμη και έτσι ολοκληρώνεται η εκτέλεση της εντολής.

Στη συνέχεια η ΚΜΕ προχωράει στην εκτέλεση της επόμενης εντολής με την ίδια πάλι διαδικασία (ανάκληση της εντολής από τη θέση 351 και εκτέλεση της).

Η ΚΜΕ 8085

Μία από τις πρώτες ΚΜΕ που κυκλοφόρησε στην αγορά και παρουσίασε σημαντική επιτυχία, ήταν η ΚΜΕ 8085 της Intel.

Στο σχήμα 3.2.11, βλέπουμε την εσωτερική αρχιτεκτονική της ΚΜΕ 8085.



Η ΚΜΕ 8085 διαθέτει μια πολύ απλή μονάδα διαδρόμου, που αποτελείται από μερικά απλά ψηφιακά κυκλώματα (απομονωτές, latch). Ολόκληρο το κύκλωμα που είναι απαραίτητο για τη σωστή λειτουργία του διαδρόμου, είναι ενσωματωμένο στη μονάδα ελέγχου της ΚΜΕ. Ο διαχωρισμός, της μονάδας του διαδρόμου από τη μονάδα ελέγχου, καθιερώθηκε σε πιο σύγχρονους επεξεργαστές.
Η μονάδα αποκωδικοποίησης των εντολών της ΚΜΕ 8085, αποτελείται από τον καταχωρητή εντολών και την λογική αποκωδικοποίησης της εντολής.

Η ΚΜΕ 8085 διαθέτει 7 καταχωρητές γενικού σκοπού, (***A,B,C,D,E,H,L***) και αρκετούς καταχωρητές ειδικού σκοπού (***PC, SP, προσωρινός καταχωρητής, καταχωρητής εντολών*** κ.ο.κ). Διαθέτει μια αριθμητική και λογική μονάδα και μια πολύπλοκη μονάδα ελέγχου.
Τελειώνοντας θα θέλαμε να τονίσουμε, ότι ο διαχωρισμός μιας ΚΜΕ σε επιμέρους μονάδες, δεν είναι μοναδικός. Παλιότερες ΚΜΕ δεν περιέχουν όλες τις μονάδες στις οποίες αναφερθήκαμε σε αυτό το , ενώ οι σύγχρονες ΚΜΕ περιέχουν περισσότερες και πιο πολύπλοκες μονάδες.

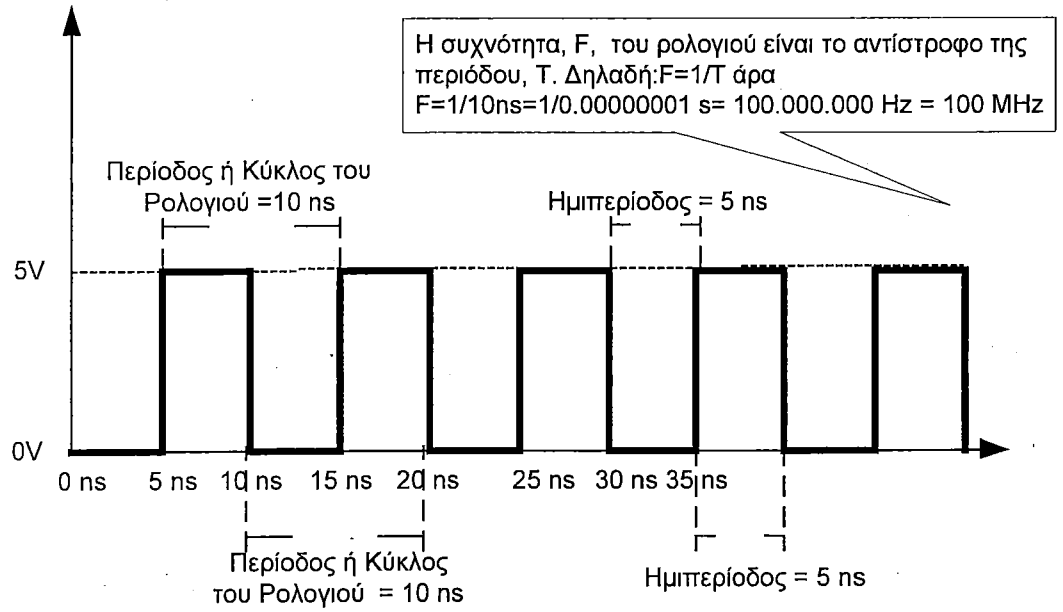
Χαρακτηριστικά της ΚΜΕ

Το ρολόι (clock)

Για να εκτελέσει η ΚΜΕ μια εντολή, εκτελεί όπως είδαμε ένα αριθμό από διαδοχικές λειτουργίες. Για παράδειγμα στο προηγούμενο , είδαμε ότι η ανάκληση και η εκτέλεση της εντολής του πολλαπλασιασμού (1010)←***A•B***, έγινε σε συγκεκριμένο αριθμό βημάτων στις διάφορετικές μονάδες της ΚΜΕ.

Κάθε μία από αυτές τις λειτουργίες διαρκεί ένα μικρό χρονικό διάστημα. Για το συγχρονισμό των λειτουργιών αυτών, είναι απαραίτητο κάποιο **ρολόι**. Σε κάθε «κτύπο» του ρολογιού η ΚΜΕ εκτελεί μία στοιχειώδη λειτουργία.

Το κύκλωμα παραγωγής του ρολογιού είναι συνήθως εξωτερικό, και παράγει μια κυματομορφή όπως αυτή του παρακάτω σχήματος.



Σχήμα 3.3.1: Η κυματομορφή του ρολογιού συχνότητας 100 MHz

Παρατηρούμε ότι το σήμα του ρολογιού, εναλλάσσεται μεταξύ της στάθμης των 5 V και αυτής των 0V. Λόγω του σχήματος του το σήμα αυτό ονομάζεται **τετραγωνική παλμοσειρά.** Το χρονικό διάστημα μεταξύ δύο διαδοχικών εναλλαγών του σήματος, για παράδειγμα από τα 0V στα 5V και πάλι στα 0V είναι ίσο με μια **ημιπερίοδο** του ρολογιού. Δύο διαδοχικές ημιπερίοδοι αποτελούν μια ολόκληρη **περίοδο** ή όπως αλλιώς λέμε ένα **κύκλο** του ρολογιού. Το αντίστροφο της περιόδου είναι η **συχνότητα.** Η συχνότητα μας πληροφορεί τον αριθμό των κύκλων του ρολογιού στη διάρκεια του ενός δευτερολέπτου και για τους μικροεπεξεργαστές συνήθως είναι της τάξης των εκατοντάδων MHz. Το ένα MHz είναι ίσο με ένα εκατομμύριο κύκλους το δευτερόλεπτο.

Κάθε ΚΜΕ είναι σχεδιασμένη να λειτουργεί μέχρι κάποια μέγιστη συχνότητα. Συνήθως η συχνότητα του ρολογιού, επιλέγεται να είναι ίση με τη μέγιστη επιτρεπόμενη συχνότητα λειτουργίας της ΚΜΕ και αυτό γιατί όπως θα δούμε όσο μεγαλύτερη είναι η συχνότητα του ρολογιού τόσο πιο γρήγορα εκτελεί τις εντολές η ΚΜΕ. Η συχνότητα του ρολογιού ονομάζεται συχνά και συχνότητα λειτουργίας της ΚΜΕ.

Ο ρυθμός με τον οποίο εκτελούνται οι εντολές είναι συνάρτηση της συχνότητας λειτουργίας της ΚΜΕ. Είναι όμως λάθος να πιστεύουμε ότι η ΚΜΕ εκτελεί εντολές με τη συχνότητα λειτουργίας του ρολογιού της. Εάν το ρολόι της ΚΜΕ είναι 100 MHz (100.000.000 κύκλους το δευτερόλεπτο) αυτό δε σημαίνει απαραίτητα ότι εκτελούνται και 100.000.000 εντολές το δευτερόλεπτο.

Όπως έχουμε αναφέρει ανάλογα με το πώς έχει κατασκευαστεί μια ΚΜΕ μια εντολή για να εκτελεστεί χρειάζεται πολλές διαδοχικές λειτουργίες κάθε μια από τις οποίες διαρκεί ένα κύκλο του ρολογιού.

Ο κατασκευαστής συνήθως δίνει τον αριθμό των κύκλων του ρολογιού που απαιτούνται για την ανάκληση και εκτέλεση κάθε εντολής. Έτσι εάν μια εντολή πρόσθεσης χρειάζεται 5 κύκλους ρολογιού για να ανακληθεί και να εκτελεστεί και η συχνότητα λειτουργίας της ΚΜΕ είναι 100 MHz, δηλαδή κάθε κύκλος διαρκεί 10 ns, τότε η εντολή της πρόσθεσης χρειάζεται συνολικά 5*10 ns=50 ns για να εκτελεστεί. Συνεπώς μπορούνται να εκτελούνται 20.000.000 εκατομμύρια προσθέσεις το δευτερόλεπτο.

Ένα μέτρο της ταχύτητας της ΚΜΕ είναι ο μέσος όρος των εντολών που μπορεί να εκτελέσει σε ένα δευτερόλεπτο. Με ειδικά προγράμματα (benchmarks) μπορούν οι κατασκευαστές να μετρήσουν πόσες εντολές εκτελεί κατά μέσο όρο η ΚΜΕ. Ο μέσος όρος των εντολών που εκτελεί μια ΚΜΕ το

δευτερόλεπτο μετριέται σε MIPS (Million Instructions Per Second), δηλαδή σε εκατομμύρια εντολές το δευτερόλεπτο. Στην περίπτωση που για παράδειγμα έχουμε μια ΚΜΕ με ρολόι 100 MHz το δευτερόλεπτο, και η κάθε εντολή διαρκεί 5 κύκλους ρολογιού, τότε η ταχύτητα της είναι 100.000.000/5=20.000.000 εντολές= 20 MIPS.

Το εύρος σε bits της ΚΜΕ

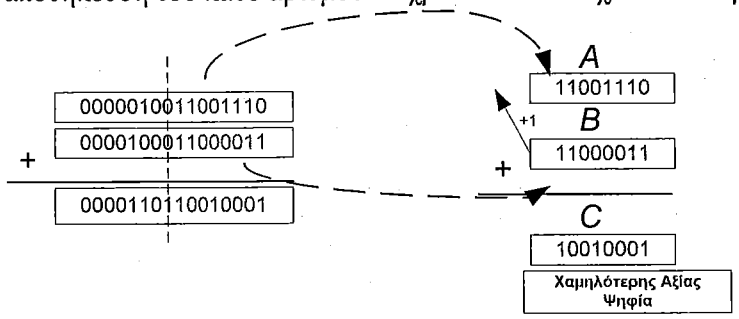
Η ΚΜΕ μπορεί να κάνει αριθμητικές ή λογικές πράξεις μεταξύ των καταχωρητών της. Συνεπώς κάθε φορά τα δεδομένα πρέπει πρώτα να εισάγονται από τη μνήμη ή από τις περιφερειακές μονάδες σε κάποιο καταχωρητή της ΚΜΕ και μετά γίνεται η επεξεργασία τους.
Ένα σημαντικό χαρακτηριστικό της ΚΜΕ είναι το **εύρος** σε bits των καταχωρητών και της Αριθμητικής και Λογικής Μονάδας (ΑΛΜ), που διαθέτει. Όσο πιο «μεγάλος» είναι ένας καταχωρητής τόσα «περισσότερα bits του δεδομένου χωράει».

Για παράδειγμα έστω ότι θέλουμε να εκτελέσουμε την πρόσθεση 1230+ 2243 χρησιμοποιώντας μια ΚΜΕ με τρεις καταχωρητές, τους ***A***, ***B***, ***C***. Για να εκτελέσει η ΚΜΕ την πρόσθεση θα πρέπει πρώτα να μεταφερθούν οι αριθμοί σε δύο από τους καταχωρητές της. Ο αριθμός 1230_{10} είναι ίσος με $4CE_{16}$= 10011001110_2 και χρειαζόμαστε τουλάχιστον 11 bits για να τον αποθηκεύσουμε. Εάν η ΚΜΕ μας διαθέτει καταχωρητές και ΑΛΜ των 16 bits τότε η αποθήκευση του δεδομένου θα γίνει μόνο σε ένα καταχωρητή, για παράδειγμα στον καταχωρητή ***A***. Όμοια η αποθήκευση του 2243_{10}=$8C3_{16}$= 100011000011_2 θα γίνει σε ένα άλλο καταχωρητή, για παράδειγμα στον καταχωρητή ***B***.

```
  0000010011001110   A    1230(10)
+ 0000100011000011   B  + 2243(10)
-------------------------------------
  0000110110010001   C    3473(10)
```

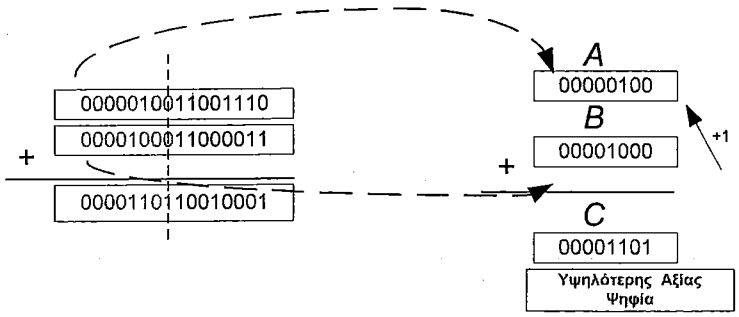
Σχήμα 3.3.2: Πρόσθεση με 16 bit καταχωρητές

Η πρόσθεση θα γίνει με μια μόνο εντολή, την ***C←A+B*** στην ΑΛΜ ενώ το αποτέλεσμα χωράει να αποθηκευτεί σε ένα καταχωρητή εύρος 16 bit. Η μεταφορά του αποτελέσματος από την ΚΜΕ στη μνήμη ή σε κάποια περιφερειακή μονάδα χρειάζεται επίσης μόνο μια εντολή.
Εάν η ΚΜΕ όμως που χρησιμοποιούμε διαθέτει καταχωρητές και ΑΛΜ των 8 bits τότε η αποθήκευση του κάθε αριθμού θα χρειαστεί τουλάχιστον δύο βήματα.



Σχήμα 3.3.3: Πρόσθεση με 8-bit καταχωρητές. Βήμα 1°

Στο σχήμα φαίνεται ο τρόπος αποθήκευσης των προσθετέων και του αποτελέσματος μέσα στους καταχωρητές της ΚΜΕ. Έτσι στον καταχωρητή ***A*** βάζουμε τα 8 bit με τη μικρότερη αξία του αριθμού 1230_{10}=10011001110_2. Στον καταχωρητή ***B*** αποθηκεύονται με τον ίδιο ακριβώς τρόπο τα 8 bit με τη μικρότερη αξία του αριθμού 2243_{10}= 100011000011_2. Η πρόσθεση γίνεται και το αποτέλεσμα αποθηκεύεται στον καταχωρητή ***C***. Εάν έχει προκύψει κρατούμενο κατά την πρόσθεση των δύο καταχωρητών, τότε η σημαία του κρατουμένου της ΑΛΜ τίθεται στην τιμή 1. Το αποτέλεσμα αποθηκεύεται στη μνήμη και είμαστε έτοιμοι για το δεύτερο βήμα.
Στο δεύτερο βήμα, στους καταχωρητές ***A*** και ***B*** της ΚΜΕ θα φορτωθούν, τα υπόλοιπα υψηλότερης αξίας bit των αριθμών 1230_{10}=10011001110_2 και 2243_{10}= 100011000011_2 όπως ακριβώς φαίνεται στο σχήμα.



Σχήμα 3.3.4: Πρόσθεση με 8-bit καταχωρητές. Βήμα 2°

Η πρόσθεση τώρα γίνεται πιο πολύπλοκη μια και πρέπει να συμπεριλάβουμε και το κρατούμενο που έχει προκύψει από την προηγούμενη πρόσθεση. Συνεπώς στην περίπτωση όπου έχει προκύψει κρατούμενο, όπως συμβαίνει στο παράδειγμα μας θα πρέπει να προσθέσει επιπλέον άλλη μια μονάδα στο άθροισμα που προκύπτει από την πρόσθεση των καταχωρητών ***A*** και ***B*** που περιέχουν τα **υψηλότερης αξίας δεδομένα**. Το αποτέλεσμα αυτής της πρόσθεσης αποθηκεύεται στον καταχωρητή ***C***.

Παρατηρούμε ότι για την αποθήκευση του συνολικού αποτελέσματος της πρόσθεσης χρειαζόμαστε δύο καταχωρητές και η μεταφορά του στη μνήμη ή σε κάποια μονάδα εξόδου χρειάζεται δύο εντολές. Συμπεραίνουμε ότι το μήκος σε bits των καταχωρητών και της ΑΛΜ της ΚΜΕ επηρεάζει σημαντικά την επίδοση της ΚΜΕ. Όσο πιο μεγάλοι είναι οι καταχωρητές της ΚΜΕ τόσο πιο γρήγορα κατά κανόνα γίνεται η εκτέλεση των προγραμμάτων. Βέβαια το μήκος των καταχωρητών δεν μπορεί να γίνει απεριόριστα μεγάλο αφού γίνεται εξαιρετικά πολύπλοκη η κατασκευή της ΚΜΕ.
Συνήθως οι καταχωρητές έχουν ίδιο μήκος με το μήκος σε bits των υπόλοιπων εσωτερικών μονάδων της ΚΜΕ, όπως η ΑΛΜ το οποίο λεμε και μήκος της ΚΜΕ . Η παράμετρος αυτή είναι αρκετά σημαντική για την επίδοση του συστήματος. Έτσι μια 16 bit ΚΜΕ σε σύγκριση με μια 8 bit ΚΜΕ που έχουν παρόμοιες εσωτερικές μονάδες, την ίδια συχνότητα ρολογιού και το ίδιο ρεπερτόριο εντολών είναι τουλάχιστον δύο φορές πιο γρήγορη.
Στον πίνακα 3.3.2 βλέπουμε τον μεγαλύτερο φυσικό αριθμό που μπορούμε να αποθηκεύσουμε σε ένα καταχωρητή ανάλογα με το εύρος του. Γενικά σε ένα καταχωρητή των n bit μπορούμε να αποθηκεύσουμε όλους τους αριθμούς από 0 έως 2^{n-1}.

Εύρος σε bits	Μέγιστη ακέραια τιμή που μπορούμε να παραστήσουμε
8	2^8-1= 255
16	2^{16}-1= 65.535
32	2^{32}-1= 4.294.967.295
64	2^{64}-1= 18.446.744.073.709.551.615

Πίνακας 3.3.2 Μέγιστη ακέραια τιμή που μπορούμε να παραστήσουμε σε συνάρτηση του αριθμού των bits

Στον πίνακα 3.3.3, παρουσιάζεται το εύρος σε bits μερικών από τις γνωστότερες ΚΜΕ. Θα πρέπει να διευκρινίσουμε εδώ ότι μια 8 bit ΚΜΕ μπορεί να περιέχει κάποιους 16 bit καταχωρητές καθώς επίσης και 16 bit εσωτερικές μονάδες. Παρ' όλα αυτά, δεν λέμε ότι το εύρος αυτής είναι 16 bit, παρά μόνο εάν όλες οι εσωτερικές μονάδες και κυρίως οι καταχωρητές και η αριθμητική και λογική μονάδα, της ΚΜΕ δεν είναι 16 bit.
Για παράδειγμα η ΚΜΕ, Z80, διαθέτει μετρητή προγράμματος των 16 bit και μπορεί να διαχειρίζεται διευθύνσεις μνήμης των 16 bit. Επιπλέον η αριθμητική και λογική μονάδα υποστηρίζει προσθέσεις και αφαιρέσεις μεταξύ 16 bit καταχωρητών η ίδια η ΑΛΜ είναι των 8 bit (η πράξη των 16 bit γίνεται σε δύο βήματα). Για αυτό η ΚΜΕ Z80 λέμε ότι έχει εύρος 8 bit.

ΚΜΕ	Εύρος της ΚΜΕ σε bit
Z80	8 bit
8085	8 bit
8086	16 bit
8088	16 bit
80386	32 bit
80486	32 bit

Pentium	32 bit

Πίνακας 3.3.3 Εύρος γνωστών ΚΜΕ σε bit

Ρεπερτόριο εντολών

Στη συνέχεια, θα εξετάσουμε ένα άλλο σημαντικό χαρακτηριστικό της ΚΜΕ, το ρεπερτόριο των εντολών που διαθέτει, δηλαδή τις εντολές που μπορεί να εκτελέσει.
Η κάθε ΚΜΕ (8086, 68000, Z80 κ.τ.λ.) υποστηρίζει και ένα διαφορετικό ρεπερτόριο εντολών.

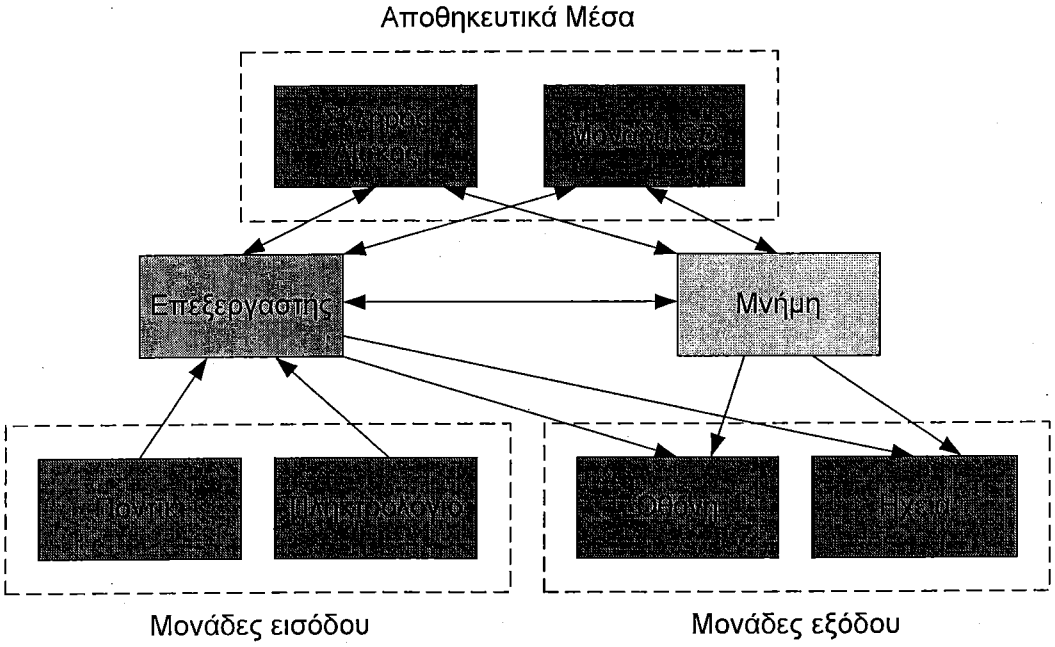
Το ρεπερτόριο εντολών μιας ΚΜΕ είναι καθοριστικός παράγοντας για να μπορέσουμε να αποφανθούμε εάν η ΚΜΕ αυτή είναι κατάλληλη για μια συγκεκριμένη χρήση. Για παράδειγμα στην περίπτωση, όπου έχουμε μια εφαρμογή ψηφιακής επεξεργασίας της ανθρώπινης φωνής, η ΚΜΕ θα πρέπει να διαθέτει εντολές όπου εκτελούν γρήγορα μαθηματικές πράξεις τόσο με ακεραίους όσο και με δεκαδικούς αριθμούς. Σε αντίθεση, η χρήση μιας ΚΜΕ σε μια εφαρμογή όπως η μεταγωγή κλήσεων σε ένα ψηφιακό τηλεφωνικό κέντρο, απαιτεί η ΚΜΕ να διαθέτει «έξυπνες» εντολές για τη γρήγορη μεταφορά των δεδομένων προς τη μνήμη και τις περιφερειακές μονάδες.

Οι κατασκευαστές συνήθως αναφέρονται στις πιθανές εφαρμογές για τις οποίες είναι κατάλληλα τα προϊόντα τους και υπάρχουν ολόκληρες οικογένειες επεξεργαστών για κάθε κατηγορία εφαρμογών.
Ένα άλλο σημαντικό χαρακτηριστικό, του ρεπερτορίου των εντολών μιας ΚΜΕ είναι η **συμβατότητα** του με παλαιότερες ΚΜΕ. Από πολύ νωρίς οι κατασκευαστές των ΚΜΕ, συμπέραναν ότι οι καινούργιες ΚΜΕ θα έπρεπε να μπορούν να εκτελούν προγράμματα παλαιότερων ΚΜΕ. Με άλλα λόγια το ρεπερτόριο των εντολών μιας καινούργιας ΚΜΕ μαζί με τις νέες εντολές, θα έπρεπε να περιέχει όλες τις εντολές της προηγούμενης ΚΜΕ, της ίδιας οικογένειας. Με αυτό το τρόπο, τα προγράμματα που έχουν γραφτεί για μια παλαιότερη ΚΜΕ δεν χρειάζεται να ξαναγραφτούν από την αρχή, για την καινούργια. Το χαρακτηριστικό αυτό έπαιξε σημαντικό ρόλο στην γρήγορη εξέλιξη των προσωπικών υπολογιστών και αποτέλεσε σημαντικό κίνητρο για την καθιέρωση τους.

Διάδρομοι

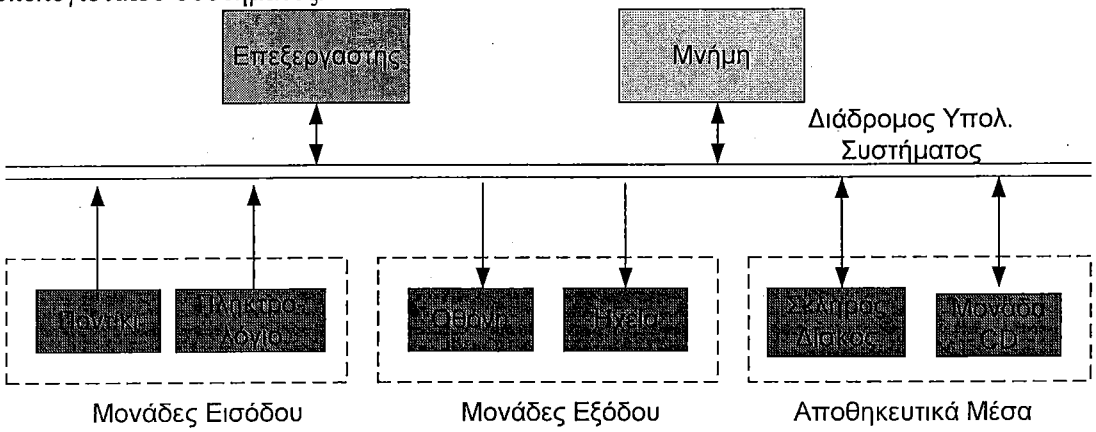
Βασικές έννοιες

Ένα υπολογιστικό σύστημα, όπως έχει αναφερθεί, αποτελείται βασικά από τον επεξεργαστή, την μνήμη, και τις περιφερειακές συσκευές. Οι συσκευές αυτές πρέπει να επικοινωνούν και να ανταλλάσσουν μεταξύ τους δεδομένα.
Ένας απλοϊκός τρόπος σύνδεσης είναι αυτός που απεικονίζεται στο σχήμα 3.4.1. Κάθε συσκευή συνδέεται με όλες τις άλλες συσκευές με τις οποίες πρέπει να επικοινωνεί. Αυτός ο τρόπος σύνδεσης αν και άμεσος έχει το μειονέκτημα ότι οδηγεί σε πολύπλοκη διασύνδεση. Επίσης δεν μπορούν εύκολα να συνδεθούν καινούργιες συσκευές που τυχόν θα θέλαμε να προσθέσουμε στο σύστημά μας.



Σχήμα 3.4.1 Απλοϊκή σύνδεση μονάδων

Ένας πιο ευέλικτος και αποτελεσματικός τρόπος σύνδεσης των συσκευών ενός υπολογιστικού συστήματος είναι η παράλληλη σύνδεση των συσκευών μέσω μίας κοινής λεωφόρου διακίνησης δεδομένων, όπως φαίνεται στο σχήμα 3.4.2. Η λεωφόρος αυτή των δεδομένων ονομάζεται διάδρομος του υπολογιστικού συστήματος.



Σχήμα 3.4.2. Αρχιτεκτονική υπολογιστικού συστήματος

Η συνδεσμολογία αυτή, που βασίζεται σε ένα κοινό διάδρομο, έχει το πλεονέκτημα ότι η σύνδεση τον συσκευών είναι αρκετά απλή και προσφέρει τη δυνατότητα της εύκολης προσθήκης νέων περιφερειακών μονάδων.

Το μειονέκτημα αυτής της συνδεσμολογίας είναι ότι μόνο δύο μονάδες μπορούν να επικοινωνούν ταυτόχρονα μεταξύ τους. Αν προσπαθήσουν την ίδια στιγμή και άλλες δύο μονάδες να ανταλλάξουν δεδομένα τότε λέμε ότι έχουμε σύγκρουση δεδομένων. Είναι σαν να προσπαθούμε να περάσουμε την ίδια στιγμή από τον ίδιο αγωγό δύο διαφορετικά σήματα. Για να μην συμβεί σύγκρουση, η διακίνηση των δεδομένων στο διάδρομο, που αποτελεί το κοινό μέσο επικοινωνίας των μονάδων, ελέγχεται από τον επεξεργαστή.

Σε συστήματα απλής αρχιτεκτονικής, υπάρχει η δυνατότητα να επικοινωνεί μόνο ο επεξεργαστής με την μνήμη ή ο επεξεργαστής με μία περιφερειακή μονάδα. Δεν επιτρέπεται όμως να επικοινωνούν δύο περιφερειακές μονάδες μεταξύ τους ή με την μνήμη. Για παράδειγμα για την απεικόνιση στην οθόνη ενός χαρακτήρα από το πάτημα ενός πλήκτρου, πρέπει να περάσει πρώτα ο χαρακτήρας (ο κωδικός του) από το πληκτρολόγιο στον επεξεργαστή και στην συνέχεια από τον επεξεργαστή στην οθόνη πάντα μέσω του διαδρόμου. Αυτό δεν συμβαίνει σε σύγχρονα συστήματα όπου παρέχεται η

δυνατότητα απευθείας προσπέλασης των περιφερειακών προς την μνήμη (Direct Memory Addressing – DMA).

Όπως αναφέρθηκε και παραπάνω το κοινό μέσο επικοινωνίας των συσκευών με τον επεξεργαστή ονομάζεται **διάδρομος του υπολογιστικού συστήματος**. Φυσικά ο διάδρομος αυτός αποτελείται από ένα σύνολο παράλληλων γραμμών σύνδεσης. Το πλήθος τους ονομάζεται **εύρος διαδρόμου**. Ανάλογα με το είδος της πληροφορίας που έχουν οι γραμμές αυτές μπορούμε να διακρίνουμε τις παρακάτω ομάδες συνδέσεων που αποτελούν τα μέρη ενός διαδρόμου.

Διάδρομος δεδομένων (Data bus) ονομάζεται το σύνολο των γραμμών από τις οποίες μεταφέρονται δεδομένα. Τα δεδομένα μπορεί να μεταφέρονται από τον επεξεργαστή, για παράδειγμα προς μία περιφερειακή μονάδα ή αντίστροφα. Από κάθε γραμμή μπορεί να μεταφέρεται ένα bit κάθε φορά. Ένας ASCII χαρακτήρας περιλαμβάνει 8 bit. Συνεπώς για να μεταφέρονται ASCII χαρακτήρες, ένας χαρακτήρας κάθε φορά, απαιτείται διάδρομος δεδομένων με εύρος 8 bit. Αν ο διάδρομος δεδομένων είχε εύρος 16 bit τότε θα μπορούσαν να μεταφέρονται δύο (2) χαρακτήρες ASCII κάθε φορά.

Διάδρομος διευθύνσεων (Address bus) ονομάζεται το σύνολο των γραμμών από τις οποίες μεταφέρεται η πληροφορία που χαρακτηρίζει και προσδιορίζει τη συσκευή με την οποία θέλει να επικοινωνήσει ο επεξεργαστής. Επειδή τη διεύθυνση τη δίνει ο επεξεργαστής οι γραμμές κατευθύνονται από τον επεξεργαστή προς το διάδρομο διευθύνσεων. Στη συνέχεια οι γραμμές των διευθύνσεων μέσα από το διάδρομο κατευθύνονται προς τις περιφερειακές μονάδες και τη μνήμη.

Διάδρομος ελέγχου ονομάζεται το σύνολο των γραμμών που μεταφέρουν σήματα συγχρονισμού και εντολές του επεξεργαστή και των περιφερειακών συσκευών για την ασφαλή και γρήγορη μεταφορά των δεδομένων. Επίσης τα σήματα ανάγνωσης ή εγγραφής περιλαμβάνονται στο διάδρομο αυτό.

Όπως έχει αναφερθεί παραπάνω **εύρος διαδρόμου** ονομάζεται το πλήθος των γραμμών από τις οποίες αποτελείται. Το εύρος του διαδρόμου δεδομένων είναι σημαντικό για την ταχύτητα μεταφοράς των δεδομένων. Όσο περισσότερες είναι οι γραμμές του διαδρόμου δεδομένων τόσο περισσότερα δυαδικά ψηφία μπορούν να μεταδοθούν παράλληλα. Το εύρος του διαδρόμου δεδομένων στα υπολογιστικά συστήματα κυμαίνεται από 8 έως και 64 bit. Για παράδειγμα ένας διάδρομος δεδομένων με εύρος 16 bit μπορεί να μεταφέρει μόνο δύο byte ταυτόχρονα, ενώ ένας διάδρομος δεδομένων με εύρος 64 bit μπορεί να μεταφέρει ταυτόχρονο 8 bytes.

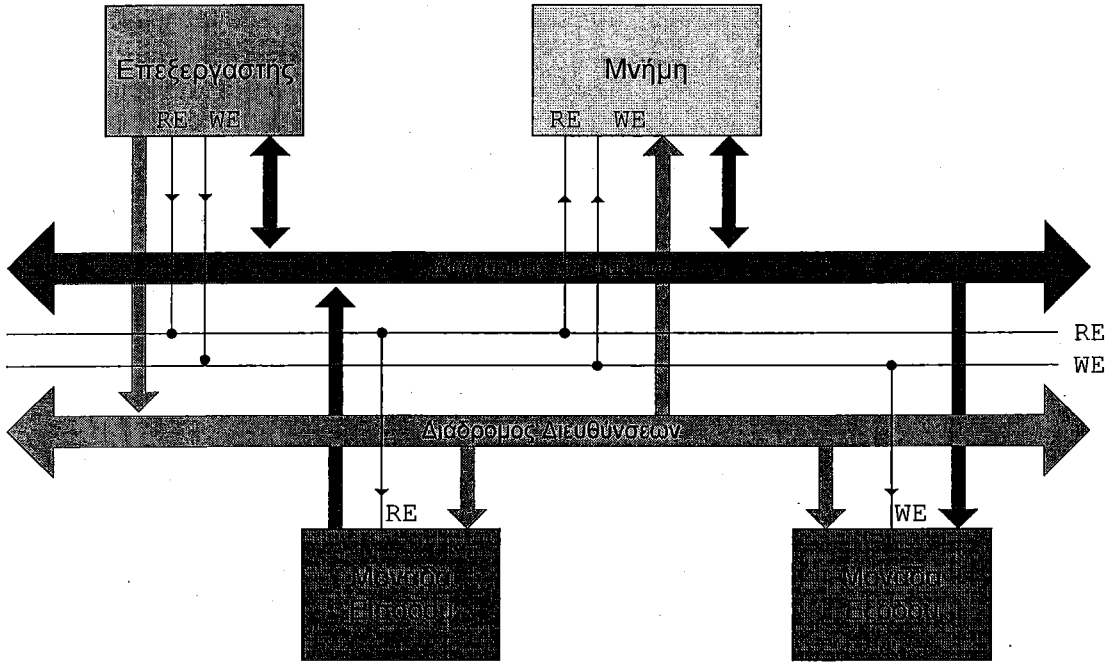
Τέλος ένα χαρακτηριστικό μέγεθος ενός διαδρόμου είναι η **ταχύτητα** του. **Ταχύτητα διαδρόμου** ονομάζεται το πλήθος των διαφορετικών δεδομένων που μπορούν να μεταφερθούν σε ένα δευτερόλεπτο. Για παράδειγμα σε διάδρομο με εύρος διαδρόμου δεδομένων 32 και ταχύτητα 10 εκατομμύρια δεδομένα το δευτερόλεπτα μπορούν να μεταφερθούν 10x32 = 320 εκατομμύρια δυαδικά ψηφία (bit) ή 40 εκατομμύρια bytes το δευτερόλεπτο.

Σύνδεση Μονάδων με το διάδρομο

Ένα υπολογιστικό σύστημα αποτελείται από τον επεξεργαστή, την μνήμη, τις μονάδες εισόδου, εξόδου και τις άλλες περιφερειακές μονάδες. Όπως έχει αναφερθεί αυτές οι μονάδες συνδέονται μεταξύ τους με τους διαδρόμους διευθύνσεων, δεδομένων και ελέγχου.
Ένας επεξεργαστής διαθέτει τα παρακάτω σήματα με τα οποία συνδέεται με το υπόλοιπο σύστημα:
Σήματα διευθύνσεων από τα οποία δίνει την διεύθυνση της μονάδας με την οποία θέλει να επικοινωνήσει.
Σήματα δεδομένων με τα οποία διαβάζει και γράφει δεδομένα στις υπόλοιπες μονάδες.
Σήμα RE (Read Enable) που ενεργοποιείται, όταν ο επεξεργαστής θέλει να διαβάσει δεδομένα.
Σήμα WE (Write Enable) που ενεργοποιείται, όταν ο επεξεργαστής θέλει να γράψει δεδομένα.
Η μνήμη ενός υπολογιστικού συστήματος πρέπει να διαθέτει ανάλογα σήματα με αυτά του επεξεργαστή. Σήματα διευθύνσεων, δεδομένων καθώς και τα σήματα RE και WE.

Επίσης οι μονάδες εισόδου πρέπει να διαθέτουν σήματα δεδομένων και διευθύνσεων. Μια μονάδα εισόδου μπορεί μόνο να δώσει δεδομένα στο διάδρομο και όχι να πάρει δεδομένα από αυτόν. Έτσι σαν σήμα ελέγχου αρκεί να έχει μόνο το σήμα RE (Read Enable). Αντίστοιχα οι μονάδες εξόδου πρέπει να διαθέτουν σήματα για διευθύνσεις και δεδομένα καθώς και το σήμα WE. Οι μονάδες

εξόδου δεν διαθέτουν σήμα RE, διότι σε αυτές γράφουμε δεδομένα και ποτέ δεν διαβάζουμε από αυτές.



Σχήμα 3.4.3 Διάδρομοι και σήματα ελέγχου

Στο σχήμα 3.4.3 φαίνεται η κατεύθυνση των δεδομένων και τα σήματα ελέγχου (RE, WE) με τις μονάδες που συνδέονται.

Ας υποθέσουμε ότι η μονάδα εξόδου είναι μια οθόνη και η μονάδα εισόδου ένα πληκτρολόγιο. Και έστω ότι θέλουμε να εμφανίζεται στην οθόνη ο χαρακτήρας που αντιστοιχεί στο πλήκτρο που πατάμε από τον πληκτρολόγιο.

Το πληκτρολόγιο σαν μονάδα εισόδου δίνει δεδομένα όταν επιλεγεί. Ας υποθέσουμε στο παράδειγμά μας ότι η διεύθυνση του πληκτρολογίου είναι 200. Έτσι ο επεξεργαστής, όταν θέλει να διαβάσει από το πληκτρολόγιο, επιλέγει το πληκτρολόγιο δίνοντας στον διάδρομο των διευθύνσεων τον αριθμό 200. Ύστερα δίνει την εντολή για εγγραφή δεδομένου στον διάδρομο των δεδομένο μέσω του σήματος RE (Read Enable). Όταν το πληκτρολόγιο, που είναι ήδη επιλεγμένο, λάβει την εντολή να δώσει ένα δεδομένο στον διάδρομο δεδομένων τότε ο κωδικός του πλήκτρου που μόλις πατήθηκε εμφανίζεται στον διάδρομο δεδομένων και από εκεί το διαβάζει ο επεξεργαστής.

Κατόπιν ο επεξεργαστής πρέπει αυτό το δεδομένο να το γράψει στην οθόνη. Υποθέτουμε ότι η οθόνη έχει την διεύθυνση 100. Η επιλογή της οθόνης θα γίνει, όταν στον διάδρομο των διευθύνσεων εμφανιστεί η διεύθυνση 100. Η οθόνη είναι συσκευή εξόδου και ο επεξεργαστής της δίνει δεδομένα. Έτσι για να στείλει το χαρακτήρα στην οθόνη βάζει στον διάδρομο των διευθύνσεων την διεύθυνση 100 προκαλώντας την επιλογή της οθόνης και στην συνέχεια δίνει το χαρακτήρα, που έχει διαβάσει από το πληκτρολόγιο, στον διάδρομο των δεδομένων. Τέλος, με το σήμα WE (Write Enable) επιτρέπει στην οθόνη, που έχει ήδη επιλεγεί, να πάρει το δεδομένο από τον αντίστοιχο διάδρομο.

Λειτουργία διαδρόμου

Σε ένα υπολογιστικό σύστημα, αυτός που ρυθμίζει την διακίνηση των δεδομένων στο διάδρομο είναι ο επεξεργαστής. Έτσι σε κάθε μεταφορά δεδομένου πρέπει απαραίτητα να συμμετέχει. Το πότε θα δώσουν ή θα πάρουν δεδομένα οι περιφερειακές μονάδες καθώς και η μνήμη κανονίζεται από τα σήματα των διευθύνσεων και ελέγχου που δίνει ο επεξεργαστής.

Κάθε περιφερειακή μονάδα ενός υπολογιστικού συστήματος πρέπει να έχει μια διεύθυνση με τη οποία προσδιορίζεται. Η διεύθυνση αυτή είναι το όνομά της και πρέπει κάθε περιφερειακή μονάδα να έχει μία μοναδική και διαφορετική διεύθυνση.

Το ίδιο ισχύει και για την μνήμη. Το σύνολο των διευθύνσεων είναι διαθέσιμο και για την μνήμη και για τις περιφερειακές μονάδες εισόδου και εξόδου. Μια μνήμη, όμως επειδή διαθέτει πολλές θέσεις στις οποίες μπορούν να αποθηκευτούν διαφορετικά δεδομένα, επιλέγεται για ένα σύνολο διευθύνσεων. Κάθε φορά που ο επεξεργαστής θέλει να γράψει ή να διαβάσει μία θέση από το συγκρότημα αυτό της μνήμης, επιλέγει ολόκληρη τη μνήμη. Βέβαια κάθε μνήμη έχει και δικό της εσωτερικό κύκλωμα αποκωδικοποίησης για την επιλογή της συγκεκριμένης θέσης μέσα στο συγκρότημα της μνήμης αυτής.

Οι μονάδες εισόδου και εξόδου συνήθως έχουν μία διεύθυνση με την οποία επιλέγονται. Βέβαια υπάρχουν και περιφερειακές μονάδες που διαθέτουν περισσότερες από μία θέσεις αποθήκευσης ή ανάγνωσης και επιλέγονται από περισσότερες από μία διευθύνσεις.

Κάθε περιφερειακή μονάδα, όπως και η μνήμη, πρέπει να διαθέτει ένα λογικό κύκλωμα αποκωδικοποίησης της διεύθυνσης. Στο κύκλωμα αυτό κάθε μονάδα συγκρίνει τη διεύθυνση του διαδρόμου με τη δική της διευθύνση. Αν η διεύθυνση είναι ίδια, τότε η μονάδα αυτή συνδέεται στο διάδρομο και επικοινωνεί ανταλλάσσοντας δεδομένα με τον επεξεργαστή.

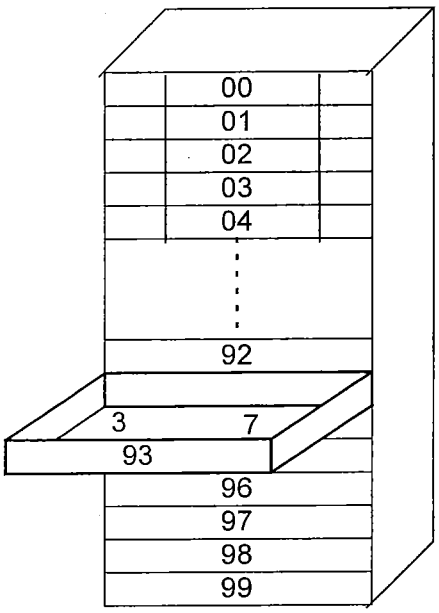
Στο διάδρομο δεδομένων είναι συνδεδεμένες και μπορούν να δώσουν δεδομένα ή να πάρουν δεδομένα όλες οι μονάδες του υπολογιστικού συστήματος. Αυτές φαίνονται σαν να είναι συνδεδεμένες μεταξύ τους αφού όλες συνδέονται στο κοινό διάδρομο του συστήματος. Στη μεταφορά όμως δεδομένων μόνο μία συσκευή πρέπει να στέλνει δεδομένα και μόνο μια άλλη να τα λαμβάνει. Τα δεδομένα αυτά δεν πρέπει να επηρεάζονται από τις υπόλοιπες μονάδες. Για την λύση του προβλήματος αυτού κάθε μονάδα που συνδέεται πάνω στον διάδρομο δεδομένων πρέπει να διαθέτει ειδικούς απομονωτές. Όταν μια μονάδα δεν είναι επιλεγμένη τότε με την βοήθεια των απομονωτών η μονάδα αυτή αποσυνδέεται και δεν επηρεάζει τα δεδομένα του διαδρόμου. Ενώ όταν η μονάδα είναι ενεργοποιημένη, δηλαδή έχει επιλεγεί, είναι πλήρως συνδεδεμένη στον διάδρομο.

Η μνήμη

Εισαγωγή – Χαρακτηριστικά στοιχεία

Ας θεωρήσουμε ένα απλό υπολογιστικό σύστημα, και να υποθέσουμε ότι πρέπει να εισάγουμε από το πληκτρολόγιο εκατό 100 διψήφιους αριθμούς τους οποίους θέλουμε να εισάγουμε στο υπολογιστικό σύστημα και μετά να υπολογίσουμε το άθροισμά τους.

Για να γίνει αυτό, πρέπει το υπολογιστικό σύστημα να αποθηκεύσει αυτούς τους αριθμούς στην μνήμη του. Το μέγεθος της μνήμης πρέπει να είναι τέτοιο ώστε να χωρούν και οι εκατό (100) αριθμοί. Το πλήθος των αριθμών που μπορεί να αποθηκεύσει μία μνήμη εξαρτάται από το πόσες διαφορετικές θέσεις έχει η μνήμη αυτή.



Σχήμα 3.5.1 Η μνήμη

Ας φανταστούμε την μνήμη, όπως φαίνεται στο σχήμα 3.5.1, σαν ένα πλήθος από 100 συρτάρια. Το κάθε συρτάρι είναι μία θέση αποθήκευσης δεδομένων και έχει σαν διεύθυνση έναν αριθμό. Έτσι η αρίθμηση των συρταριών αποτελεί τη διεύθυνση τους ενώ το περιεχόμενό τους αντιστοιχεί στα δεδομένα. Για παράδειγμα στη διεύθυνση ή στη θέση μνήμης 93 είναι ο αριθμός 37. Γενικά μπορούμε να πούμε ότι κάθε θέση μνήμης έχει μία διεύθυνση δηλαδή έναν αριθμό που την χαρακτηρίζει για να δώσουμε ή να πάρουμε ένα δεδομένο από την θέση αυτή. Μια μνήμη λοιπόν, όσες θέσεις αποθήκευσης έχει, τόσες διαφορετικές διευθύνσεις πρέπει να περιλαμβάνει. Το πλήθος των διαφορετικών θέσεων και συνεπώς διευθύνσεων που έχει μία μνήμη ονομάζεται **μέγεθος της μνήμης**.

Ας γυρίζουμε στο σχήμα 3.5.1 και να παρατηρήσουμε ότι για την παράσταση της διεύθυνσης που προσδιορίζει τη θέση μνήμης χρησιμοποιούμε δύο δεκαδικά ψηφία. Με αυτόν τον περιορισμό μπορούμε να έχουμε το πολύ 100 διαφορετικές διευθύνσεις (από 0 έως 99). Αν χρησιμοποιούσαμε 3 δεκαδικά ψηφία, τότε το πλήθος των θέσεων μνήμης που θα μπορούσαμε να αριθμήσουμε είναι 1000, από 000 έως 999. Άρα μπορούμε να πούμε ότι το μέγεθος μίας μνήμης καθορίζει των αριθμό των ψηφίων που χρησιμοποιούμε για την παράσταση των διευθύνσεων.

Στα υπολογιστικά συστήματα τα ψηφία που χρησιμοποιούμε είναι δυαδικά. Έτσι και η παράσταση της διεύθυνσης μίας μνήμης γίνεται από δυαδικά ψηφία. Για παράδειγμα μία μνήμη που χρησιμοποιεί 3 δυαδικά ψηφία για την παράσταση της διεύθυνσης μπορεί να έχει 2^3=8 διαφορετικές θέσεις μνήμης.

8 διαφορετικές διευθύνσεις

ΔΙΕΥΘΥΝΣΗ	ΔΕΔΟΜΕΝΑ
000=0	0011 0101=53
001=1	0100 0011=67
010=2	0101 0101=85
011=3	0010 0001=33
100=4	0001 0111=23
101=5	0011 0110=54
110=6	0010 0101=37
111=7	0010 1000=40

8 θέσεις για αποθήκευση δεδομένων

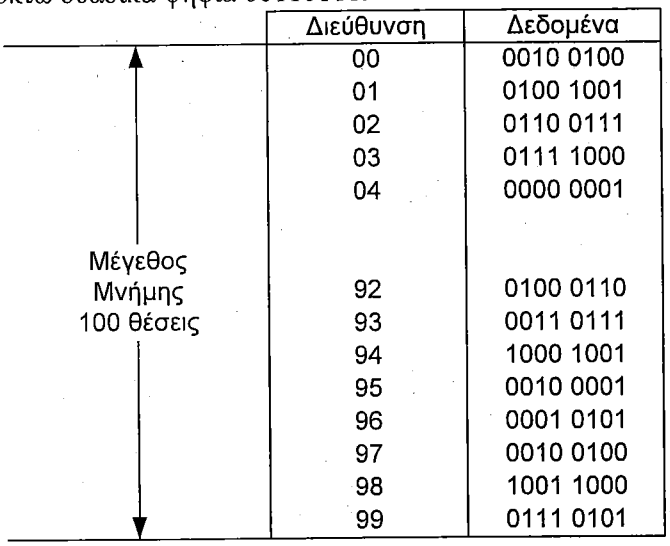
Αν υποθέσουμε ότι ο αριθμός των δυαδικών ψηφίων της διεύθυνσης της μνήμης είναι «b» και το μέγεθος της μνήμης «μ» τότε ισχύει η σχέση

$\mu=2^b$

Στον παρακάτω πίνακα υπάρχουν τυπικά μεγέθη μνημών.

Αριθμός δυαδικών ψηφίων (b) της διεύθυνσης μιας μνήμης	Μέγεθος μνήμης (μ) Πλήθος θέσεων
3	$2^3=8$
8	$2^8=256$
10	$2^{10}=1024$ 1Kbyte
16	$2^{16}=65536$ 64Kbyte
20	$2^{20}=1048576$ 1Mbyte

Ας παρατηρήσουμε πάλι το σχήμα 3.5.1 και να δούμε ότι στην διεύθυνση 93 ο αριθμός που είναι αποθηκευμένος είναι ο δεκαδικός 37. Επίσης στο σχήμα 3.5.2 μπορούμε να παρατηρήσουμε ενδεικτικά κάποια περιεχόμενα μιας μνήμης. Όλοι οι αριθμοί που είναι αποθηκευμένοι έχουν οκτώ δυαδικά ψηφία. Κάθε θέση μνήμης μπορούμε να φανταστούμε ότι είναι χωρισμένη σε 8 θέσεις στις οποίες μπορούμε νααποθηκεύσουμε ένα δυαδικό ψηφίο. Έτσι στο παράδειγμά μας, ο αριθμός 00110111, που αποτελείται από 8 ψηφία, αποθηκεύεται στην διεύθυνση 93 βάζοντας σε κάθε θέση τα οκτώ δυαδικά ψηφία 00110111.



Διεύθυνση	Δεδομένα
00	0010 0100
01	0100 1001
02	0110 0111
03	0111 1000
04	0000 0001
92	0100 0110
93	0011 0111
94	1000 1001
95	0010 0001
96	0001 0101
97	0010 0100
98	1001 1000
99	0111 0101

Σχήμα 3.5.2 Πίνακας περιεχομένων μιας μνήμης

Το πλήθος των κελίων που υπάρχουν σε μία θέση μνήμης ονομάζεται **μήκος λέξης** της μνήμης. Στο παράδειγμά μας η μνήμη που χρησιμοποιούμε έχει μήκος λέξης 8 και στο κάθε κελί μίας θέσης μνήμης μπορεί να αποθηκευτεί ένα δυαδικό ψηφίο, δηλαδή το '1' ή το '0'. Ο αριθμός των κελιών μίας θέσης μνήμης, λέγεται μήκος λέξης και είναι συνήθως του 2.

Κατηγορίες μνημών

Στο παράδειγμα της αριθμομηχανής που αναφέρθηκε μπορούμε να διακρίνουμε μνήμες που τις χρησιμοποιούμε για διαφορετικούς σκοπούς. Για παράδειγμα χρησιμοποιούμε μνήμη για να διαβάζει η κεντρική μονάδα επεξεργασίας της εντολές του προγράμματος. Ακόμα υπάρχει μνήμη όπου γράφουμε κάποια στοιχεία – δεδομένα και έπειτα τα διαβάζουμε. Ανάλογα με την λειτουργία και την εφαρμογή στην οποία θα χρησιμοποιηθεί μια μνήμη επιλέγουμε τον κατάλληλο τύπο. Έτσι οι μνήμες, ανάλογα με τις δυνατότητες που έχουν, διακρίνονται στις δύο παρακάτω κατηγορίες:

Οι μνήμες, από τις οποίες μπορούμε να διαβάζουμε και να γράφουμε, ονομάζονται **RAM** (Random Access Memory Μνήμη τυχαίας προσπέλασης).

Οι μνήμες, από τις οποίες μπορούμε να διαβάζουμε μόνο το περιεχόμενο τους, ονομάζονται **ROM** (Read Only Memory Μνήμη μόνο ανάγνωσης). Το περιεχόμενο των μνημών αυτών, δηλαδή τα δεδομένα τους, δεν μπορούμε να τα αλλάξουμε.

Οι μνήμες RAM χρησιμοποιούν δύο διαφορετικές τεχνολογίες για την αποθήκευση των δεδομένων. Ανάλογα με τον τρόπο αποθήκευσης της πληροφορίας οι μνήμες RAM χωρίζονται σε δύο κατηγορίες, τις **στατικές μνήμες** και τις **δυναμικές μνήμες**.

Οι στατικές μνήμες έχουν το σημαντικό πλεονέκτημα ότι είναι πολύ γρήγορες. Το κυριώτερο μειονέκτημα είναι η τιμή τους. Οι στατικές μνήμες επειδή χρειάζονται πολύ χώρο για να υλοποιηθούν, σε σύγκριση με τις υπόλοιπες μνήμες, είναι ακριβώτερες.

Οι δυναμικές μνήμες έχουν το πλεονέκτημα ότι είναι φτηνές και έχουν μεγάλη χωρητικότητα λόγω του μικρού τους μεγέθους.

Επίσης είναι χαμηλής κατανάλωσης αφού πρακτικά δεν καταναλώνουν ενέργεια για την φύλαξη των δεδομένων. Το μειονέκτημα των δυναμικών μνημών είναι η ταχύτητα. Επειδή η λειτουργία τους στηρίζεται στη χρήση πυκνωτών, και ένας φορτισμένος πυκνωτής με την πάροδο του χρόνου εκφορτίζεται με αποτέλεσμα να χάνει την τιμή που αποθηκεύσαμε. Για το λόγο αυτό σε τακτά χρονικά διαστήματα απαιτείται οι δυναμικές μνήμες να ανανεώνουν το περιεχόμενό τους. Το χρονικό διάστημα μεταξύ δύο διαδοχικών ανανεώσεων ονομάζεται χρόνος ανανέωσης. Σε μια διαδικασία ανανέωσης διαβάζεται το περιεχόμενο της μνήμης και αυτόματα ξαναγράφεται το ίδιο. Έτσι εξασφαλίζεται η ανανέωση του περιεχομένου. Η διαδικασία της ανανέωσης σε παλαιότερης τεχνολογίας δυναμικές μνήμες γίνεται με εξωτερικό κύκλωμα, ενώ στις νεότερου τύπου μνήμες το κύκλωμα αυτό είναι ενσωματωμένο στην ίδια ψηφίδα της μνήμης.

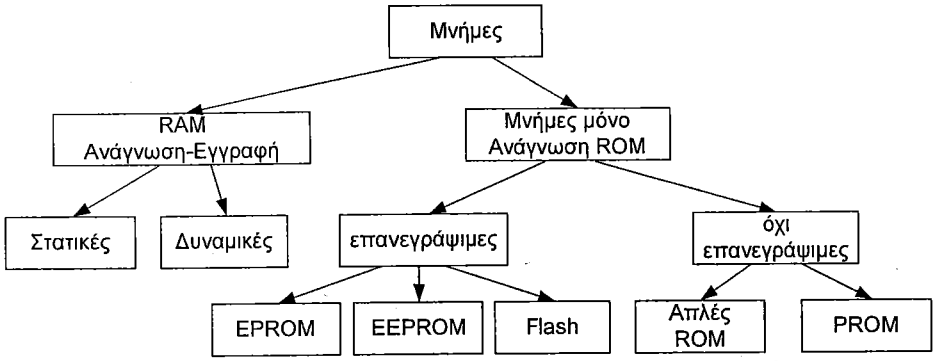
Ο όρος «Μνήμες μόνο ανάγνωσης» - ROM χρησιμοποιείται για να χαρακτηρίσει μια ολόκληρη κατηγορία μνημών από τις οποίες μπορούμε μόνο να διαβάζουμε και περιλαμβάνουν τις απλές ROM, PROM, EPROM, EEPROM, Flash. Μια σημαντική ιδιότητα τους είναι ότι δεν χάνουν τα δεδομένα τους όταν διακόψουμε την τροφοδοσία.

Οι απλές μνήμες ROM είναι κατασκευασμένες από το εργοστάσιο και έχουν συγκεκριμένα δεδομένα που δεν μπορούν να αλλάξουν. Αν χρησιμοποιούμε τέτοια μνήμη και πρέπει να αλλάξουμε το περιεχόμενό της, αυτό είναι αδύνατον. Θα πρέπει να την πετάξουμε και να χρησιμοποιήσουμε μια άλλη μνήμη, με το καινούργιο επιθυμητό περιεχόμενο.

Μια παραλλαγή της μνήμης ROM που μας δίνει ένα βαθμό ελευθερίας είναι η **PROM (Programmable Read Only Memory)**. Το περιεχόμενο της μνήμης αυτής δεν καθορίζεται κατά την κατασκευή της αλλά μπορεί να προγραμματιστεί αργότερα, αλλά μία μόνο φορά. Μια μνήμη PROM αφού προγραμματιστεί μία φορά, μετά το περιεχόμενο της δεν μπορεί να αλλάξει.

Οι μνήμες ανάγνωσης των οποίων το περιεχόμενο μπορούμε να σβήσουμε μετά τον προγραμματισμό και να τις ξαναπρογραμματίσουμε ονομάζονται **EPROM (Erasable Programmable Read Only Memory)** και **EEPROM (Electrically Erasable Programmable Read Only Memory)**. Εδώ πρέπει να τονιστεί ότι οι μνήμες EPROM και EEPROM δεν είναι μνήμες RAM. Οι μνήμες EPROM και EEPROM μας δίνουν την δυνατότητα να προγραμματίσουμε το περιεχόμενο τους πολλές φορές όχι όμως να γράφουμε σε αυτές, σαν να ήταν μνήμες RAM.

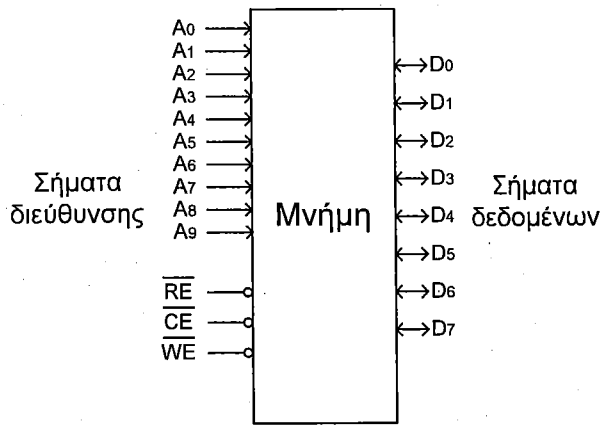
Οι μνήμες EPROM και EEPROM έχουν διαφορετικό τρόπο σβησίματος. Στις μνήμες EPROM το περιεχόμενο τους σβήνεται με υπεριώδες φως, ενώ στις μνήμες EEPROM το περιεχόμενο τους σβήνεται με ηλεκτρικό τρόπο. Τέλος, μια ειδική κατηγορία των επανεγράψιμων μνημών είναι και οι μνήμες FLASH που έχουν παρόμοια χαρακτηριστικά με τις EEPROM. Η διαφορά τους είναι ότι, ενώ στις EEPROM μπορούμε να σβήσουμε όποια θέση μνήμης θέλουμε, στις FLASH η εντολή σβησίματος καθαρίζει όλες τις θέσεις.



Σχήμα 3.5.5 Κατηγορίες μνημών

Τα εξωτερικά σήματα μιας μνήμης

Μία μνήμη επικοινωνεί με τα υπόλοιπα κυκλώματα μέσω τριών ομάδων σημάτων. Ας πάρουμε σαν παράδειγμα μια μνήμη με χωρητικότητα 1024 byte= 1Kbyte. Στο σχήμα 3.5.6, το οποίο απεικονίζει μια τέτοια μνήμη, μπορούμε να διακρίνουμε τις εξής ομάδες σημάτων:



Σχήμα 3.5.6 Συμβολική παράσταση μνήμης

Τα σήματα της διεύθυνσης (A_0-A_9).

Τα σήματα της διεύθυνσης συνδέονται στον διάδρομο διευθύνσεων του υπολογιστικού συστήματος. Τα σήματα αυτά δηλώνονται με το γράμμα A και δείκτη έναν αριθμό που δείχνει την δυαδική αξία του bit της διεύθυνσης. Το σήμα με δείκτη τον αριθμό 0 αντιστοιχεί στο bit με την μικρότερη δυαδική αξία (LSB-least significant bit) και το σήμα με τον δείκτη 9 στο bit με την μεγαλύτερη δυαδική αξία (MSB-most significant bit).

Τα σήματα των δεδομένων (D_0-D_7)

Τα σήματα των δεδομένων συνδέονται στο διάδρομο δεδομένων του υπολογιστικού συστήματος. Τα σήματα αυτά δηλώνονται με το γράμμα D και δείκτη έναν αριθμό που δείχνει την δυαδική αξία του bit όπως στα σήματα της διεύθυνσης. Τα σήματα των δεδομένων είναι κοινά για τα δεδομένα εισόδου και τα δεδομένα εξόδου.

Τα σήματα ελέγχου ($\overline{\text{RE}}, \overline{\text{CE}}, \overline{\text{WE}}$).

Τα σήματα ελέγχου συνδέονται στο διάδρομου ελέγχου του υπολογιστικού συστήματος. Όπως κάθε συσκευή που συνδέεται σε διάδρομο υπολογιστικού συστήματος, έτσι και η μνήμη διαθέτει σήμα ενεργοποίησης $\overline{CE}$ (**Chip Enable**). Γραμμή πάνω από το σήμα δηλώνει ότι η μνήμη ενεργοποιείται, δηλαδή επιλέγεται όταν το σήμα $\overline{CE}$ έχει χαμηλή τάση, ενώ μένει απενεργοποιημένη όσο το σήμα $\overline{CE}$ έχει υψηλή τάση. Το σήμα αυτό πολλές φορές συμβολίζεται και ως $\overline{CS}$ (Chip Select).

Πρόσθετα οι μνήμες RAM διαθέτουν σήμα επιλογής εγγραφής στην μνήμη $\overline{WE}$ (**Write Enable**). Αντίστοιχα όταν το σήμα $\overline{WE}$ έχει χαμηλή τάση τότε η μνήμη εκτελεί την διαδικασία εγγραφής δεδομένων, ενώ όταν το σήμα $\overline{WE}$ βρίσκεται σε υψηλή τάση τότε τα περιεχόμενα της μνήμη δεν μπορούν να αλλάξουν.

Τέλος οι μνήμες RAM διαθέτουν σήμα επιλογής ανάγνωσης ή εγγραφής στην μνήμη $\overline{RE}$ (**Read Enable**). Όταν το σήμα $\overline{RE}$ έχει χαμηλή τάση τότε η μνήμη εκτελεί την διαδικασία ανάγνωσης δεδομένων, ενώ όταν το σήμα $\overline{RE}$ βρίσκεται σε υψηλή τάση τότε η μνήμη δεν εμφανίζει τα δεδομένα στην έξοδο της.

Λειτουργίες Μνήμης

Ανάγνωση Μνήμης

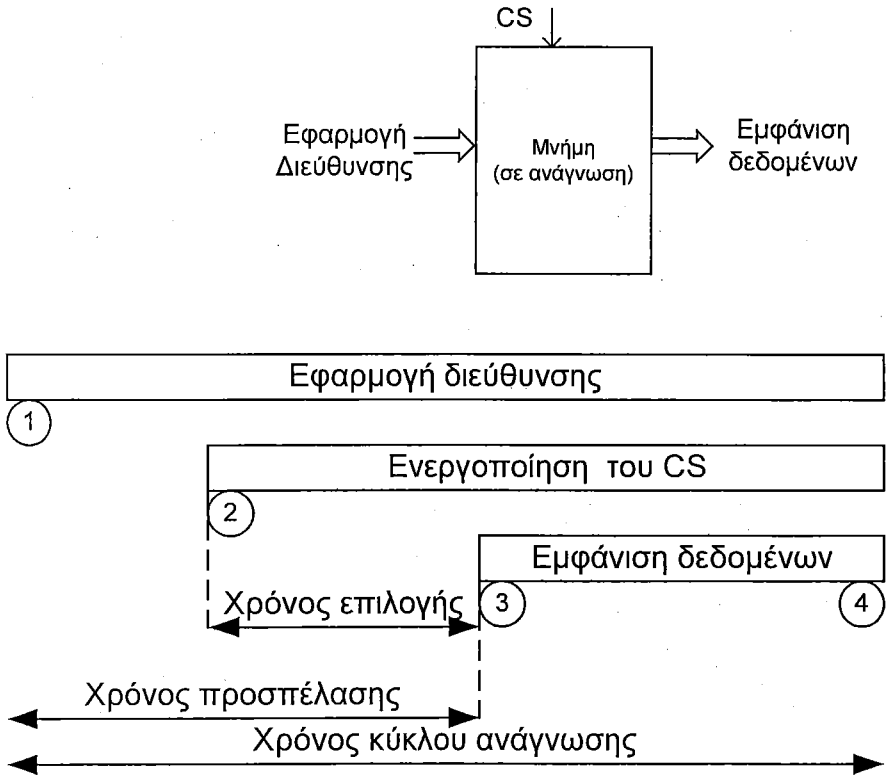
Για την λειτουργία ανάγνωσης μιας μνήμης, πρέπει τα σήματα της διεύθυνσης, των δεδομένων και τα σήματα ελέγχου να πάρουν κατάλληλες τιμές σε συγκεκριμένους χρόνους. Οι χρόνοι αυτοί είναι διαφορετικοί από μνήμη σε μνήμη και είναι χαρακτηριστικοί για την ταχύτητα της μνήμης.

Η απλή λειτουργία της ανάγνωσης της μνήμης γίνεται στα τέσσερα παρακάτω βήματα:
Η είσοδος της διεύθυνσης της μνήμης παίρνει την τιμή της διεύθυνσης που θέλουμε να διαβάσουμε.
Ενεργοποιείται το σήμα επιλογής της μνήμης CS (chip select).
Το περιεχόμενο της θέσης μνήμης, που έχει επιλεγεί, εμφανίζεται ως έξοδος στις γραμμές των δεδομένων τις μνήμης.

Ολοκλήρωση ανάγνωσης δεδομένου

Στο σχήμα 3.6.1 φαίνεται το διάγραμμα χρονισμού μιας στατικής μνήμης, όπου παρουσιάζονται τα χρονικά διαστήματα που είναι χαρακτηριστικά στην λειτουργία της ανάγνωσης.
Η διαδικασία της ανάγνωσης ξεκινάει από την στιγμή που εφαρμόζουμε την διεύθυνση. Η μνήμη χρειάζεται κάποιο χρόνο για να κάνει την αποκωδικοποίηση της διεύθυνσης και να προσδιορίσει την θέση μνήμης που θέλουμε να διαβάσουμε. Στην συνέχεια, με το σήμα επιλογής (Chip Select (CS) τα δεδομένα της θέσης αυτής εμφανίζονται στην έξοδο.

Ο χρόνος από τη στιγμή που επιλέγουμε την διεύθυνση που θέλουμε να διαβάσουμε, μέχρι την στιγμή που εμφανίζονται τα δεδομένα της θέσης μνήμης στις γραμμές των δεδομένων ονομάζεται **χρόνος προσπέλασης** .



Σχήμα 3.6.1 Διάγραμμα χρονισμού κύκλου ανάγνωσης

Χρόνος επιλογής ονομάζεται το χρονικό διάστημα που χρειάζεται για να εμφανιστούν τα δεδομένα της θέσης μνήμης από την στιγμή που θα ενεργοποιηθεί το σήμα chip select.

Ο χρόνος προσπέλασης είναι πάντα μεγαλύτερος από τον χρόνο επιλογής επειδή πρέπει πρώτα να εφαρμόσουμε την διεύθυνση και στην συνέχεια να ενεργοποιήσουμε την μνήμη.

Η μνήμη κρατάει τα δεδομένα στην έξοδο της όσο χρονικό διάστημα η διεύθυνση μένει σταθερή και η μνήμη είναι ενεργοποιημένη. Ο ελάχιστος χρόνος μεταξύ δύο διαδοχικών αναγνώσεων διαφορετικών θέσεων μνήμης ονομάζεται **χρόνος κύκλου ανάγνωσης**, και είναι χαρακτηριστικός της ταχύτητας της μνήμης. Μας δείχνει πόσο γρήγορα μπορούμε να διαβάσουμε δεδομένα. Για παράδειγμα αν έχουμε χρόνο κύκλου ανάγνωσης 10nsec τότε σε 1 sec μπορούμε να διαβάσουμε 100 εκατομμύρια θέσεις μνήμης.

Εγγραφή μνήμης

Τα βασικά βήματα που πρέπει να γίνουν για την εγγραφή ενός δεδομένου σε μία θέση μνήμης είναι τα εξής:

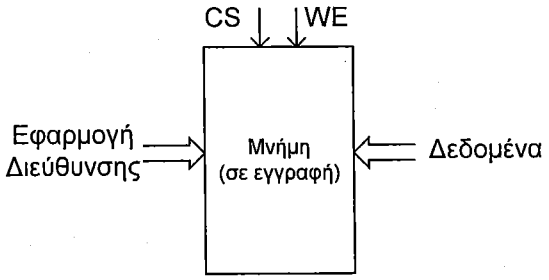
- Δίνεται η διεύθυνση στην οποία θέλουμε να γράψουμε.
- Επιλέγεται η μνήμη με την ενεργοποίηση του CS (chip select) .
- Η είσοδος των δεδομένων παίρνει την τιμή που θέλουμε να γράψουμε.
- Ενεργοποιείται η διαδικασία εγγραφής με το σήμα WE (write enable)

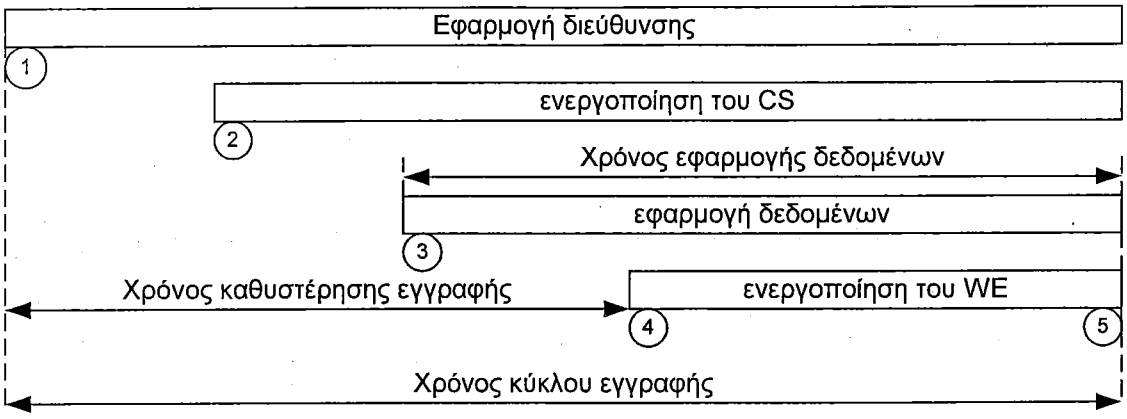
Η εγγραφή έχει ολοκληρωθεί και μπορεί να ξεκινήσει νέα εγγραφή ή ανάγνωση.

Στο σχήμα 3.6.2 φαίνεται το διάγραμμα χρονισμού της εγγραφής μίας στατικής μνήμης.

Η εγγραφή του δεδομένου στην μνήμη ξεκινάει όταν ενεργοποιηθεί το σήμα write enable (WE). Η ενεργοποίηση του σήματος αυτού πρέπει να γίνει αφού περάσει ένα χρονικό διάστημα από τη στιγμή που εφαρμόστηκε η διεύθυνση εγγραφής. Το χρονικό αυτό διάστημα χρειάζεται για τον προσδιορισμό της θέσης μνήμης και ονομάζεται **χρόνος καθυστέρησης εγγραφής** (t_{AW}).

Η διαδικασία της εγγραφής ξεκινάει με την εφαρμογή της διεύθυνσης που θέλουμε να γράψουμε και την ενεργοποίηση της μνήμης με το σήμα chip select. Στην συνέχεια πρέπει να εφαρμόσουμε τα δεδομένα που θέλουμε να γράψουμε και να τα διατηρήσουμε σταθερά. Ο χρόνος που απαιτείται να παραμείνουν τα δεδομένα σταθερά ονομάζεται **χρόνος εφαρμογής δεδομένων** και συμβολίζεται με t_{DW}.





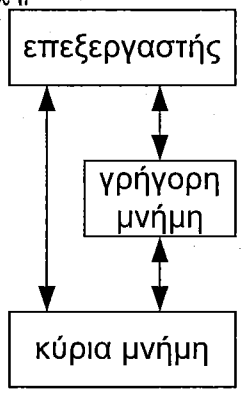
Σχήμα 3.6.2 Διάγραμα χρονισμού κύκλου εγγραφής

Η κάθε μνήμη χρειάζεται κάποιο χρόνο για να ολοκληρωθεί η εγγραφή του δεδομένου και να μπορεί να ξεκινήσει ένας νέος κύκλος εγγραφής ή ανάγνωσης. Το άθροισμα των χρόνων που χρειάζονται για τα πέντε αυτά βήματα ονομάζεται **χρόνος κύκλου εγγραφής.** (t_{WCY}). Είναι χαρακτηριστικός της ταχύτητας μίας μνήμης γιατί δείχνει το χρόνο που διαρκεί μια εγγραφή.

Λανθάνουσα μνήμη

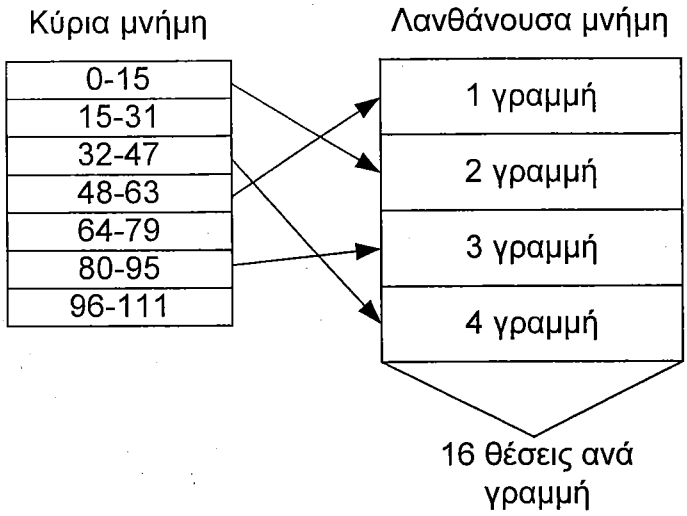
Η ταχύτητα ενός υπολογιστικού συστήματος εξαρτάται σε μεγάλο βαθμό και από την ταχύτητα της μνήμης. Όσο γρηγορότερα γίνεται η ανάγνωση και η εγγραφή στην μνήμη τόσο πιο γρήγορα εκτελούνται τα προγράμματα σε ένα υπολογιστικό συστήμα. Η χρήση όμως γρήγορης κύριας μνήμης δεν είναι δυνατόν να γίνει επειδή το κόστος του υπολογιστικού συστήματος θα ήταν πολύ υψηλό.

Για την αύξηση της ταχύτητας του υπολογιστικού συστήματος χρησιμοποιείται μία γρήγορη μνήμη, που είναι φυσικά πολύ μικρότερη σε μέγεθος από την κύρια μνήμη. Στη μνήμη αυτή κρατάμε το αντίγραφο ενός μικρού τμήματος της κύριας μνήμης. Η μνήμη αυτή που ονομάζεται **λανθάνουσα μνήμη** (cache memory) παρεμβάλλεται στη διακίνηση των δεδομένων μεταξύ του επεξεργαστή και της κύριας μνήμης, όπως φαίνεται στο σχήμα 3.6.3.



Σχήμα 3.6.3 Θέση λανθάνουσας μνήμης

Η λανθάνουσα μνήμη χωρίζεται σε ομάδες θέσεων μνήμης που ονομάζονται **γραμμές** (cache line). Για παράδειγμα μία γραμμή μπορεί να αποτελείται από 16 θέσεις μνήμης. Σε κάθε γραμμή υπάρχει το αντίγραφο 16 θέσεων της κύριας μνήμης που είναι γειτονικά στην κύρια μνήμη όπως φαίνεται και στο σχήμα 3.6.4.



Σχήμα 3.6.4 Οργάνωση λανθάνουσας μνήμης

Ας υποθέσουμε τώρα ότι ο επεξεργαστής θέλει να διαβάσει ένα δεδομένο από μία διεύθυνση της μνήμης έστω την 37.

Τότε γίνονται τα παρακάτω βήματα:

- Ελέγχεται αν η διεύθυνση 37 βρίσκεται στην λανθάνουσα μνήμη.
- Ας υποθέσουμε ότι βρίσκεται στην 4 γραμμή της λανθάνουσας μνήμης, όπως φαίνεται στο σχήμα. Τότε η ανάγνωση γίνεται από τη λανθάνουσα μνήμη χωρίς προσπέλαση στην σχετικά πιο αργή κύρια μνήμη.

- Αν δεν βρίσκεται, τότε η ανάγνωση γίνεται από την κύρια μνήμη. Ταυτόχρονα όμως οι 15 γειτονικές θέσεις μνήμης, μαζί με τη διεύθυνση που διαβάσαμε, αντιγράφονται σε μία γραμμή της λανθάνουσας μνήμης. Ο λόγος είναι ότι αναμένουμε να ακολουθήσουν και άλλες αναγνώσεις από γειτονικές θέσεις. Έτσι στα επόμενα βήματα οι αναγνώσεις που θα ακολουθήσουν θα γίνουν από τη λανθάνουσα μνήμη, η οποία είναι πιο γρήγορη.

Όταν γίνονται αναγνώσεις από τον επεξεργαστή πολλά κομμάτια από την κύρια μνήμη αντιγράφονται στις γραμμές της λανθάνουσας μνήμης, μέχρι να γεμίσουν όλες οι γραμμές. Όταν χρειαστεί να μεταφερθεί ένα κομμάτι της μνήμης στην ήδη γεμάτη λανθάνουσα μνήμη τότε πρέπει μία τουλάχιστον γραμμή της λανθάνουσας μνήμης να διαγραφεί. Υπάρχουν δύο τεχνικές που συνήθως εφαρμόζονται στην επιλογή της γραμμής που πρέπει να διαγραφεί.

Διαγραφή με βάση το χρόνο παραμονής (FIFO first in - first out). Σύμφωνα με την τεχνική αυτή η γραμμή που έχει παραμείνει στην λανθάνουσα μνήμη περισσότερο χρόνο θα πρέπει να διαγραφεί. Δηλαδή διαγράφεται η παλαιότερη χρονικά εγγραφή.

Διαγραφή με βάση την λιγότερη χρήση (LRU - least recently used).
Σύμφωνα με την τεχνική αυτή η γραμμή που θα διαγραφεί είναι αυτή που έχει μείνει αχρησιμοποίητη περισσότερο χρόνο από τις άλλες. Έτσι μια παλιά εγγραφή που χρησιμοποιείται συχνά δεν διαγράφεται σε αντίθεση με την προηγούμενη τεχνική.

Στην περίπτωση που ο επεξεργαστής θέλει να γράψει στην μνήμη διακρίνουμε δύο περιπτώσεις:
Αν η διεύθυνση που θέλουμε να γράψουμε βρίσκεται μόνο στην κύρια μνήμη και όχι στην λανθάνουσα μνήμη, τότε η εγγραφή γίνεται κανονικά.

Αν το δεδομένο της διεύθυνσης που θέλουμε να γράψουμε βρίσκεται και στην λανθάνουσα μνήμη, τότε εγγραφή μόνο στην κύρια μνήμη θα δημιουργήσει ασυμφωνία με την λανθάνουσα. Για να ξεπεραστεί το πρόβλημα αυτό, χρησιμοποιεί μία από τις παρακάτω τεχνικές:

Διεγγραφής (write throught). Σύμφωνα με αυτή την τεχνική η εγγραφή γίνεται ταυτόχρονα και στην κύρια μνήμη και στην λανθάνουσα. Έτσι δεν υπάρχει περίπτωση η κύρια μνήμη και η λανθάνουσα μνήμη να έχουν διαφορετικές τιμές.

Επανεγγραφής (write on). Με την μέθοδο της επανεγγραφής η εγγραφή γίνεται μόνο στην λανθάνουσα μνήμη. Το περιεχόμενο της κύριας μνήμης δεν ενημερώνεται, παρά μόνο όταν χρειαστεί να διαγραφεί η λέξη από την λανθάνουσα μνήμη.

Η αποτελεσματικότητα της λανθάνουσας μνήμης στηρίζεται στο γεγονός ότι τα προγράμματα που εκτελεί ένας επεξεργαστής έχουν την ιδιότητα, οι αναγνώσεις και οι εγγραφές στην μνήμη να μην γίνονται σε τυχαίες διευθύνσεις αλλά σε γειτονικές και κοντινές μεταξύ τους θέσεις. Αυτό ονομάζεται **τοπικότητα της αναφοράς** των προγραμμάτων. Το τελικό αποτέλεσμα είναι ότι οι αναγνώσεις και οι εγγραφές της μνήμης να γίνονται με την ταχύτητα της λανθάνουσας μνήμης και όχι της αργής κύριας μνήμης.

Με την ανάπτυξη της τεχνολογίας των μνήμων και την αύξηση της ταχύτητας και της πυκνότητας αποθήκευσης έχει γίνει εφικτό να χρησιμοποιούνται πλέον δύο επίπεδα λανθάνουσας μνήμης.

Το πρώτο επίπεδο λανθάνουσας μνήμης (level 1) βρίσκεται μέσα στο ολοκληρωμένο του επεξεργαστή. Η μνήμη αυτή είναι ταχύτερη. Εργάζεται με την ταχύτητα του επεξεργαστή

Το δεύτερο επίπεδο λανθάνουσας μνήμης (level 2) στους επεξεργαστές προηγούμενης τεχνολογίας βρισκόταν σε ξεχωριστό κύκλωμα από αυτό του επεξεργαστή. Στους σύγχρονους όμως επεξεργαστές και το δεύτερο επίπεδο της λανθάνουσας μνήμης, όπως φυσικά και το πρώτο επίπεδο είναι ενσωματωμένο στο ίδιο ολοκληρωμένο του επεξεργαστή. Τα δύο επίπεδα λανθάνουσας μνήμης βελτιώνουν ακόμα περισσότερο την ταχύτητα διακίνησης (εγγραφών – αναγνώσεων) δεδομένων από την μνήμη.

Τεχνικές μεταφοράς δεδομένων

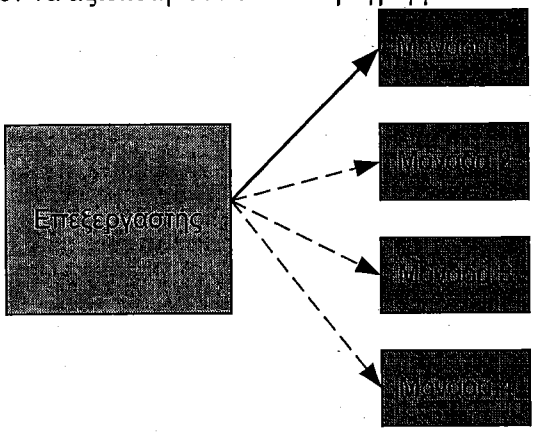
Επικοινωνία Συσκευών με τον επεξεργαστή

Στα προηγούμενα μαθήματα περιγράφονται οι τρόποι με τους οποίους ο επεξεργαστής επικοινωνεί με την μνήμη και τις περιφερειακές του μονάδες. Στο αυτό θα δούμε τεχνικές με τις οποίες οι περιφερειακές μονάδες ειδοποιούν τον επεξεργαστή ότι χρειάζονται να επικοινωνήσουν μαζί του.

Ας υποθέσουμε ότι έχουμε ένα πληκτρολόγιο συνδεδεμένο σε ένα υπολογιστικό σύστημα. Το πληκτρολόγιο είναι μια μονάδα εισόδου. Κάθε φορά που πατάμε ένα πλήκτρο πρέπει το πληκτρολόγιο να επικοινωνήσει με τον επεξεργαστή και να πει ποιο πλήκτρο πατήθηκε. Οι χρονικές όμως στιγμές στις οποίες εμείς πατάμε ένα πλήκτρο είναι τυχαίες. Ο επεξεργαστής δεν ξέρει πότε πρέπει να περιμένει πάτημα πλήκτρου και αν πρέπει να περιμένει. Αν ο επεξεργαστής κοιτάει συνέχεια το πληκτρολόγιο αν πατήθηκε κάποιο πλήκτρο τότε δεν θα μπορεί να εκτελεί κανένα άλλο πρόγραμμα συγχρόνως. Θα είναι διαρκώς απασχολημένος με την παρακολούθηση του πληκτρολογίου

Μια απλή τεχνική που εφαρμόζεται για την λύση αυτού του προβλήματος σε απλά υπολογιστικά συστήματα είναι η τεχνική τακτικής σάρωσης **(polling).** Ας υποθέσουμε ότι έχουμε τέσσερις συσκευές εισόδου, συνδεδεμένες σε ένα υπολογιστικό σύστημα. (Σχήμα 3.8.1) Σύμφωνα με την τεχνική αυτή ο επεξεργαστής ελέγχει κατά τακτά χρονικά διαστήματα κάθε συσκευή εισόδου, αν έχει κάτι να «πει». Δηλαδή ελέγχει ένα σήμα κάθε μονάδας εισόδου που του «λέει» αν είναι ενεργοποιημένο. Με το σήμα αυτό η μονάδα δηλώνει ότι έχει δεδομένα που πρέπει να στείλει στον επεξεργαστή. Με αυτό τον τρόπο ο επεξεργαστής μπορεί να τρέχει ένα πρόγραμμα και σε τακτές χρονικές στιγμές διακόπτει την εκτέλεση του προγράμματος και κοιτάει μία-μία τις περιφερειακές συσκευές αν έχουν κάποιο καινούργιο δεδομένο. Όπως φαίνεται και στο σχήμα 3.8.1 ο επεξεργαστής κοιτάει πρώτα την μονάδα 1 αν έχει καινούργιο δεδομένο. Αν έχει εξυπηρετεί την μονάδα 1, ενώ στην αντίθετη περίπτωση κοιτάει την μονάδα 2. Η διαδικασία αυτή συνεχίζεται ώσπου να ελεγχούν όλες οι τέσσερις μονάδες. Συνήθως οι μονάδες εισόδου είναι αργές συσκευές. Ας πάρουμε για παράδειγμα το πλητρολόγιο και ας υποθέσουμε ότι μπορούμε να πλητρολογούμε 10 πλήκτρα το δευτερόλεπτο. Με

αυτό το ρυθμό το πληκτρολόγιο θα ενεργοποιεί το σήμα που δείχνει ότι έχει νέο δεδομένο κάθε 100 msec. Σήμερα ένας απλός επεξεργαστής με συχνότητα λειτουργίας στα 20MHz μπορεί να εκτελέσει περίπου 100.000 εντολές μέσα στο χρονικό διάστημα των 100 msec. Από αυτό το παράδειγμα καταλαβαίνουμε ότι ο επεξεργαστής μπορεί να εκτελεί ένα πρόγραμμα και ανά τακτά χρονικά διαστήματα να ελέγχει τις μονάδες εισόδου, χωρίς κανένα πρόβλημα. Έτσι αν ελέγχει κάθε 100 msec έχει διαθέσιμες 100.000 εντολές. Ένας μικρός αριθμός από αυτές (~10 εντολές) αρκούν για τον έλεγχο. Οι υπόλοιπες μπορούν να αξιοποιηθούν από το πρόγραμμα που εκτελεί.



Σχήμα 3.8.1 Ελέγχος μονάδων με Polling

Το πλεονέκτημα αυτής της τεχνικής είναι ότι η σχεδίαση του υπολογιστικού συστήματος είναι αρκετά απλή και το πρόγραμμα που χρειάζεται ο επεξεργαστής να τρέχει είναι εύκολο στην σχεδίαση και στην υλοποίησή του.

Τα μειονεκτήματα της είναι ότι η εξυπηρέτηση των μονάδων εισόδου γίνεται μέσα από πρόγραμμα και όχι από το υλικό. Αυτό συνεπάγεται ότι κάθε φορά που θέλουμε να ενσωματώσουμε μια νέα συσκευή πρέπει να αλλάξουμε όλο το πρόγραμμα του επεξεργαστή. Ένα άλλο σοβαρό μειονέκτημα αυτής της τεχνικής είναι ότι χάνουμε αρκετό χρόνο όταν ο επεξεργαστής σταματάει την εκτέλεση του προγράμματος για να ελέγξει όλες τις περιφερειακές συσκευές. Τις περισσότερες φορές ο επεξεργαστής δεν βρίσκει καινούργιο δεδομένο σε όλες τις συσκευές με αποτέλεσμα ο χρόνος που κάνει να ελέγξει τις συσκευές να χάνεται από τον χρόνο εκτέλεσης του προγράμματος. Τέλος, η τεχνική αυτή λειτουργεί καλά μόνο για αργές συσκευές.

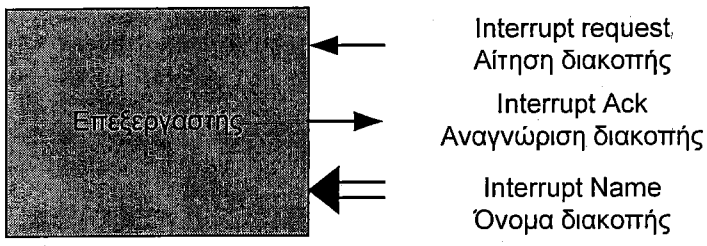
Διακοπές

Μια άλλη τεχνική που χρησιμοποιείται ευρέως την επικοινωνία των μονάδων εισόδου με τον επεξεργαστή είναι αυτή των **διακοπών (interrupt)**.

Σύμφωνα με την τεχνική αυτή ο επεξεργαστής πρέπει να έχει ένα σήμα εισόδου με το οποίο τον ειδοποιούμε ότι τουλάχιστον μία εξωτερική μονάδα θέλει να επικοινωνήσει μαζί του. Ο επεξεργαστής κάθε φορά που τελειώνει την εκτέλεση μιας εντολής κοιτάει το σήμα αυτό. Αυτό δεν γίνεται μέσω προγράμματος αλλά είναι ενσωματωμένο στο υλικό του επεξεργαστή και γίνεται αυτόματα με την εκτέλεση κάθε εντολής. Έτσι ο ρυθμός με τον οποίο ο επεξεργαστής καταλαβαίνει πότε μία εξωτερική μονάδα έχει καινούργιο δεδομένο είναι πολύ ψηλός (στο τέλος κάθε εντολής), με αποτέλεσμα να μπορεί να εξυπηρετήσει και να ανταποκριθεί σε αιτήσεις (διακοπές) γρήγορων συσκευών.

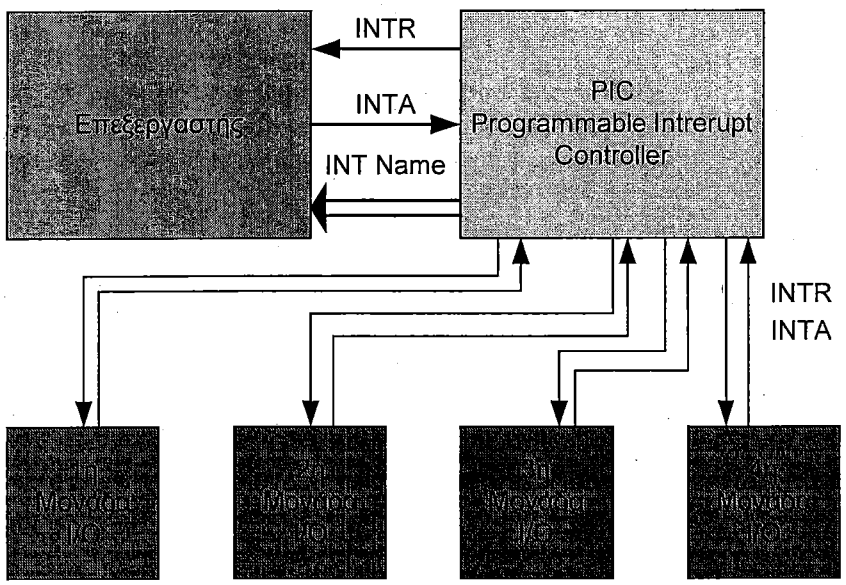
Όταν ο επεξεργαστής αναγνωρίσει ότι πρέπει να εξυπηρετήσει μια μονάδα εισόδου, σταματάει προσωρινά την εκτέλεση του προγράμματος που εκτελεί και αρχίζει την εξυπηρέτηση της μονάδας αυτής. Όταν τελειώσει την εξυπηρέτησει της μονάδας επιστρέφει στο πρόγραμμα που εκτελούσε και το συνεχίζει από το σημείο που είχε σταματήσει. Το σήμα διακοπής που ελέγχει ο επεξεργαστής ονομάζεται INTR (interrupt request – αίτηση διακοπής). Το σήμα αυτό το στέλνει η συσκευή που ζητάει εξυπηρέτησει. Όταν ο επεξεργαστής είναι έτοιμος να εξυπηρετήσει την συσκευή, που έχει ζήτησει εξυπηρέτηση, ενεργοποιεί το σήμα αναγνώρισης διακοπής INTA (interrupt ack). Η συσκευή

περιμένει από τον επεξεργαστή το σήμα INTA και μόλις αυτό ενεργοποιηθεί συνδέονται ο επεξεργαστής και η συσκευή για μεταφορά δεδομένων.



Σχήμα 3.8.2 Σήματα διακοπών

Βέβαια, στην πραγματικότητα υπάρχουν περισσότερες από μια περιφερειακές συσκευές σε ένα υπολογιστικό σύστημα. Κάθε μία από αυτές στέλνει το δικό της σήμα διακοπής σε ένα κύκλωμα που ονομάζεται **προγραμματιζόμενος ελεγκτής διακοπών (PIC).** Το κύκλωμα αυτό είναι υπεύθυνο για την παραγωγή του σήματος διακοπής που θα συνδεθεί στον επεξεργαστή. Το σήμα αυτό ενεργοποιείται, όταν μία τουλάχιστον μονάδα ζητάει να επικοινωνήσει με τον επεξεργαστή. Για κάθε περιφερειακή μονάδα ο επεξεργαστής πρέπει να εκτελέσει διαφορετικές λειτουργίες. Έτσι ο επεξεργαστής πρέπει να ξέρει ποια μονάδα ζήτησε εξυπηρέτηση. Για αυτό το λόγο ο προγραμματιζόμενος ελεγκτής διακοπών εκτός από το σήμα διακοπής, παράγει και έναν αριθμό (συνήθως 255) με το οποίο δίνει στον επεξεργαστή τον αριθμό της μονάδας εισόδου που προκαλεί τη διακοπή και πρέπει να εξυπηρετήσει.



Σχήμα 3.8.3 Σύνδεση πολλών μονάδων μέσω PIC

Στο σχήμα 3.8.3 φαίνεται η σύνδεση τεσσάρων συσκευών με τον επεξεργαστή μέσω ενός προγραμματιζόμενου ελεγκτή διακοπών. Κάθε συσκευή διαθέτει τα δύο σήματα (INTR - INTA) τα οποία συνδέονται στον ελεγκτή διακοπών. Το πρώτο (INTR) είναι έξοδος και με αυτό ζητάει εξυπηρέτηση. Το δεύτερο (INTA) είναι είσοδος και αποτελεί απάντηση στο σήμα INTR που δηλώνει ότι έχει ξεκινήσει η διαδικασία της εξυπηρέτησης.

Το πλεονέκτημα αυτής της τεχνικής είναι ότι μπορεί να εξυπηρετήσει πολύ γρήγορες περιφερειακές μονάδες. Επειδή ο επεξεργαστής ελέγχει με πολύ γρήγορο ρυθμό (σε κάθε εντολή) το σήμα διακοπής η συσκευή, που θέλει να επικοινωνήσει με τον επεξεργαστή, δεν περιμένει σχεδόν καθόλου και εξυπηρετείται αμέσως. Ακομά ο επεξεργαστής δεν χάνει χρόνο να ελέγχει αν μία μονάδα θέλει να επικοινωνήσει μαζί του όπως στην μέθοδο polling. Ο έλεγχος που εκτελεί ο επεξεργαστής για το σήμα διακοπής γίνεται από το υλικό και δεν αφαιρεί πολύτιμη υπολογιστική ισχύ του επεξεργαστή.

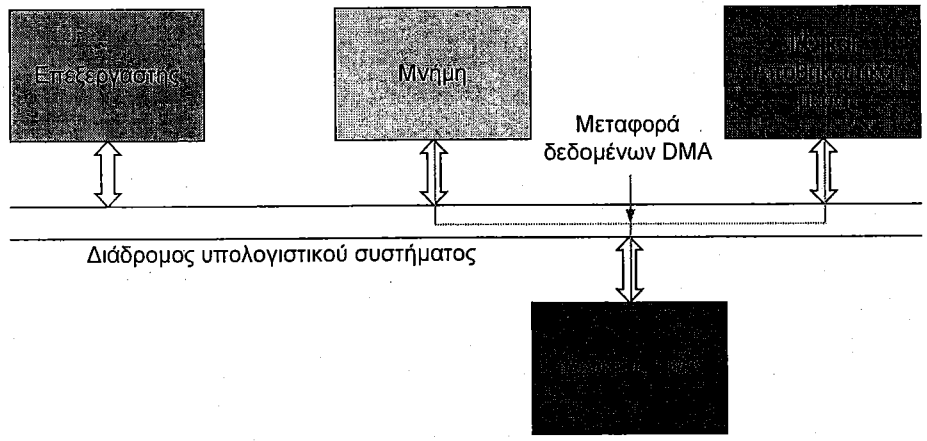
Τα σημαντικότερα μειονεκτήματα της τεχνικής αυτής είναι ότι η σχεδίαση του υπολογιστικού συστήματος είναι πλέον πιο πολύπλοκη, και το προγράμμα του επεξεργαστή γίνεται συνθετότερο.

Άμεση Προσπέλαση Μνήμης

Η τεχνική άμεσης προσπέλασης της μνήμης (DMA - (Direct Memory Access) είναι ένας τρόπος μαζικής μεταφοράς δεδομένων μεταξύ των περιφερειακών μονάδων και της μνήμης χωρίς την μεσολάβηση του επεξεργαστή.

Η τεχνική αυτή συνήθως χρησιμοποιείται με την μεταφορά δεδομένων από τα αποθηκευτικά μέσα προς την μνήμη και αντίστροφα παρακάμπτοντας τον επεξεργαστή. Για τη υλοποίηση της τεχνικής αυτή χρειάζεται ένας **ελεγκτής DMA**. Ο ελεγκτής αυτός είναι ένα ολοκληρωμένο κύκλωμα που συνδέεται πάνω στον διάδρομο του υπολογιστικού συστήματος. Όταν έχουμε άμεση προσπέλαση μνήμης ο ελεγκτής DMA αναλαμβάνει αντί της ΚΜΕ τον έλεγχο του διαδρόμου.

Σύμφωνα την τεχνική αυτή, όταν θέλουμε να μεταφέρουμε μεγάλο πλήθος δεδομένων από την μνήμη προς μονάδες εισόδου-εξόδου και αντίστροφα, ο επεξεργαστής δίνει τις κατάλληλες πληροφορίες στον DMA ελεγκτή. Με αυτές την πληροφορίες ο DMA ελεγκτής μπορεί να εκτελέσει την μεταφορά των δεδομένων χωρίς την παρέμβαση του επεξεργαστή. Το χρονικό διάστημα, που διαρκεί αυτή η μεταφορά, ο επεξεργαστής μπορεί να εκτελεί άλλες λειτουργίες αυξάνοντας έτσι την απόδοση του υπολογιστικού συστήματος.



Σχήμα 3.8.4 ελεγκτής DMA

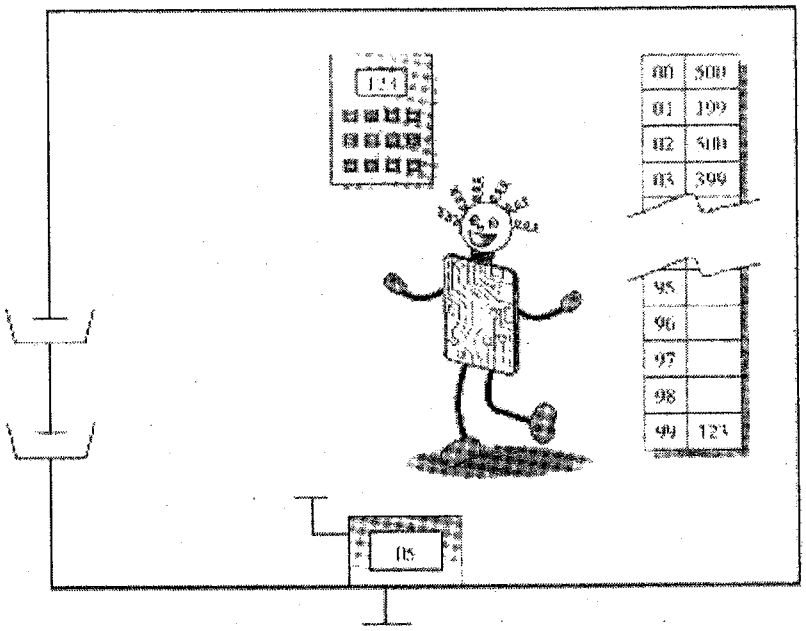
Ο ελεγκτής DMA έχει τουλάχιστον τέσσερις καταχωρητές. Σε αυτούς τους καταχωρητές ο επεξεργαστής δίνει τις πληροφορίες για την μεταφορά των δεδομένων που θέλει να κάνει. Στον ένα καταχωρητή δίνεται η αρχική διεύθυνση της μνήμης στην οποία θα αποθηκευτούν τα δεδομένα ή θα διαβαστούν από αυτή. Σε έναν άλλο καταχωρητή δίνεται το πλήθος των bytes που πρέπει να μεταφερθούν. Στον τρίτο καταχωρητή το όνομα της μονάδας εισόδου-εξόδου που θα λάβει μέρος στη μεταφορά. Τέλος, σε ένα τέταρτο καταχωρητή δίνεται η κατεύθυνση των δεδομένων. Δηλαδή αν θα μεταφερθούν δεδομένα από την μνήμη προς τη περιφερειακή μονάδα ή αντίστροφα.

Κατά την διάρκεια της λειτουργίας του ελεγκτή DMA ο έλεγχος του διαδρόμου του υπολογιστικού συστήματος μεταφέρεται από τον επεξεργαστή στον ελεγκτή DMA. Όταν τελειώσει η μεταφορά όλων των δεδομένων, ο ελεγκτής DMA δίνει με την σειρά του τον έλεγχο του διαδρόμου του υπολογιστικού συστήματος στον επεξεργαστή. Αυτός ο τρόπος λειτουργίας εξασφαλίζει την ταχύτερη μεταφορά δεδομένων και ονομάζεται «Μεταφορά Ριπής» (**Burst Mode**)

Ένας άλλος τρόπος λειτουργίας του DMA ελεγκτή είναι να μην παίρνει τον έλεγχο του διαδρόμου από τον επεξεργαστή. Η μεταφορά των δεδομένων τότε γίνεται μόνο στις χρονικές στιγμές όπου ο επεξεργαστής δεν χρησιμοποιεί τον διάδρομο. Ο τρόπος αυτός λειτουργίας ονομάζεται «**κλέψιμο κύκλου**» (**Cycle Stealing**) Η λειτουργία αυτή δεν εξασφαλίζει την ταχύτερη μεταφορά δεδομένων αλλά εξασφαλίζει την ταυτόχρονη λειτουργία του επεξεργαστή και μιας DMA μεταφοράς δεδομένων.

Το τρόπος λειτουργίας του ελεγκτή DMA, αν δηλαδή χρησιμοποιηθεί η μεταφορά ριπής ή το κλέψιμο κύκλου, εξαρτάται από την ταχύτητα της περιφερειακής μονάδας. Αν η ταχύτητα της περιφερειακής μονάδας είναι περίπου ίδια με αυτή της μνήμης, η μεταφορά ριπής είναι ο καλύτερος τρόπος. Επειδή η μεταφορά δεδομένων γίνεται με την ταχύτητα της μνήμης ο επεξεργαστής δεν μπορεί να «βρεί» χρόνο στον οποίο η μνήμη δεν δουλεύει. Έτσι αν διακόψει την άμεση μεταφορά των δεδομένων θα έχουμε καθυστέρησει την εξυπηρέτηση της γρήγορης περιφερειακής μονάδας. Σε μία όμως αργή μεταφορά ο επεξεργαστή βρίσκει χρόνο, στον οποίο η μνήμη δεν δουλεύει και μπορεί να επικοινωνήσει και αυτός με την μνήμη. Έτσι για αργές περιφερειακές μονάδες καλύτερος τρόπος είναι αυτός του κλέψιμου κύκλου. Τέλος, η επιλογή του τρόπου λειτουργίας του ελεγκτή DMA καθορίζεται από το πλήθος των δεδομένων και από το πόσο σημαντική είναι η μεταφορά των συγκεκριμένων δεδομένων.

The Little Man Computer (LMC)



Ένα κλειστό δωμάτιο που μέσα υπάρχουν διάφορα αντικείμενα και ένας μικρός άνθρωπος, που μπορεί να κάνει μόνο συγκεκριμένες και καθαρά προσδιορισμένες εργασίες.

Υπάρχοντα αντικείμενα:

1. 100 γραμματοθυρίδες - κάθε μια έχει ένα αριθμό (0-99) και θέση που μπορεί να δεχθεί μόνο ένα μικρό χαρτί
2. Ένα πολύ απλό κομπιουτεράκι που μπορεί να κάνει πρόσθεση και αφαίρεση
3. Ένα μετρητή χειρός, που ενώ η αρίθμηση γίνεται από το εσωτερικό του δωματίου, ο μηδενισμός γίνεται μόνο από έξω.

Η μόνη επαφή με το έξω κόσμο είναι ένα καλάθι εισερχομένων μηνυμάτων και ένα εξερχόμενων. Κάποιος από έξω επικοινωνεί με το μικρό ανθρωπάκι μόνο μέσω αυτών των καλαθιών. Ο μικρός άνθρωπος, μπορεί κάθε φορά να παίρνει μόνο ένα χαρτάκι από το IN καλάθι, ή να βάζει μόνο ένα χαρτάκι στο OUT καλάθι.

Όλες οι επικοινωνίες γίνονται με χρήση κωδικοποιημένων τριψηφίων αριθμών

Οι εντολές είναι γραμμένες μια σε κάθε χαρτάκι και τοποθετημένες στις θυρίδες, (Μια σε κάθε θυρίδα) με τη σειρά που θέλουμε να εκτελεστούν. Οι εντολές αυτές έχουν την εξής μορφή:

Εντολή	Αριθμός θυρίδας
1 - 5	(00 - 99)
Operation Code (OPCODE)	(Mailbox address)

Η πρώτη εντολή που θα εκτελέσει είναι στη θυρίδα με αριθμό 00.

Ο άνθρωπος καταλαβαίνει μόνο τις εξής εντολές:

INSTRUCTION	OP CODE	ΠΕΡΙΓΡΑΦΗ
LOAD	1nn	Διάβασε από τη γραμματοθυρίδα της οποίας η διεύθυνση είναι στην εντολή τον τριψήφιο αριθμό, πληκτρολόγησέ τον στο κομπιουτεράκι
STORE	2nn	Διάβασε τον αριθμό που είναι στο κομπιουτεράκι, γράψε το σε ένα χαρτάκι, και βάλε το στη θυρίδα της οποίας η διεύθυνση ήταν στο πρώτο μέρος της εντολής.
ADD	3nn	Διάβασε τον αριθμό που υπάρχει στο χαρτάκι στη γραμματοθυρίδα της οποίας η διεύθυνση είναι στην

		εντολή τον τριψήφιο αριθμό, (μην πάρεις το χαρτάκι), πήγαινε στο κομπιουτεράκι, και πρόσθεσε τον αριθμό αυτό στον ήδη υπάρχοντα.
SUBTRACT	4nn	Ίδια με την ADD αλλά αφαίρεσε όχι πρόσθεσε.
INPUT	5xx	Πήγαινε στο IN καλάθι, και πάρε το πάνω-πάνω χαρτάκι. Διάβασέ το και πήγαινε στο κομπιουτεράκι και πληκτρολόγησε τον αριθμό αυτό. Πέτα το χαρτάκι.
OUTPUT	6xx	Διάβασε τον αριθμό που είναι στο κομπιουτεράκι, γράψτο σε ένα χαρτάκι και βάλτο μέσα στο OUT καλάθι.
COFFE BRAKE (halt)	700	Μην κάνεις κάτι, μέχρι νεωτέρας.
SKIP ON CONDITION	8nn	Διάβασε τον αριθμό στο κομπιουτεράκι. Αν μια συγκεκριμένη συνθήκη ικανοποιείται μην εκτελέσεις την εντολή που είναι στην επόμενη θυρίδα (και πάτα δυο φορές το μετρητή - όχι μία όπως πάντα)
JUMP	9nn	Πήγαινε στο μετρητή και άλλαξε τον αριθμό του ώστε να δείχνει τον αριθμό θυρίδας που είναι στην εντολή. Η επόμενη εντολή που θα εκτελέσεις είναι αυτή που βρίσκεται στη θυρίδα με αυτό το αριθμό.

Στο τέλος κάθε εντολής πατά μια φορά τον μετρητή (εκτός από τις εντολές 8 και 9).

Παραδείγματα εντολών

INPUT		500
STORE	**99**	299
OUTPUT		600
ADD	**99**	399
SUBTRACT		

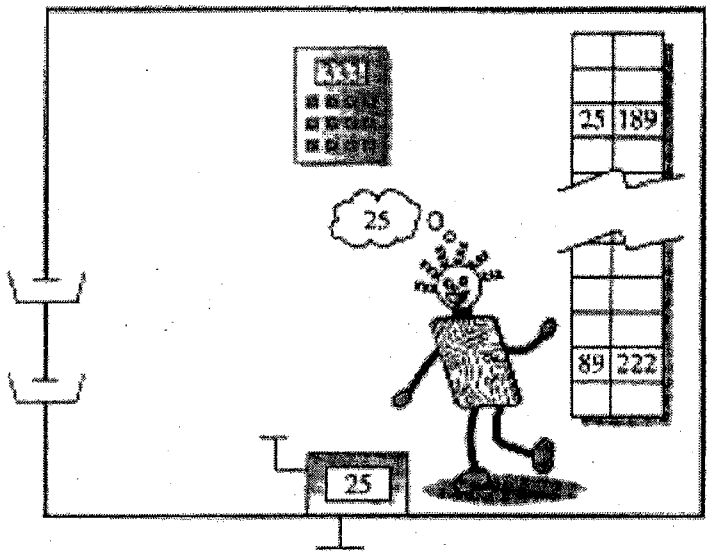
Mnemonic Codes for LMC

Mnemonic	Description	OP CODE	
LDA	Load	1nn	
STA	Store	2nn	
ADD	Add	3nn	
SUB	Subtract	4nn	
IN	Input	500	
OUT	Output	600	
HLT	Halt (Cofee brake)	700	
SKN	Skip if Negative	800	8 είναι η εντολή SKIP. 00 η παραλλαγή της (if negative)
SKZ	Skip if Zero	801	
SKP	Skip if Positive	802	
JMP	Jump		

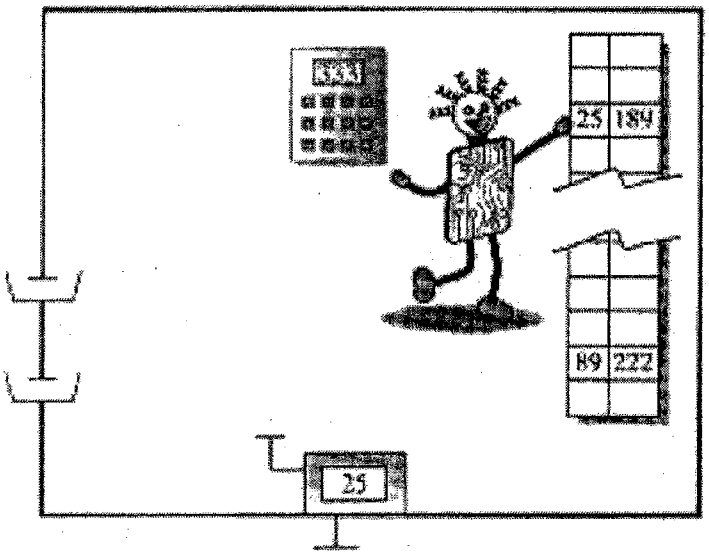
Παράδειγμα Προγράμματος

(Οι OPCODE γράφονται εδώ σαν να είναι δύο διαφορετικοί αριθμοί, ώστε να ξεχωρίζει η εντολή από τη διεύθυνση - αριθμός θυρίδας - όπου αυτό είναι απαραίτητο).

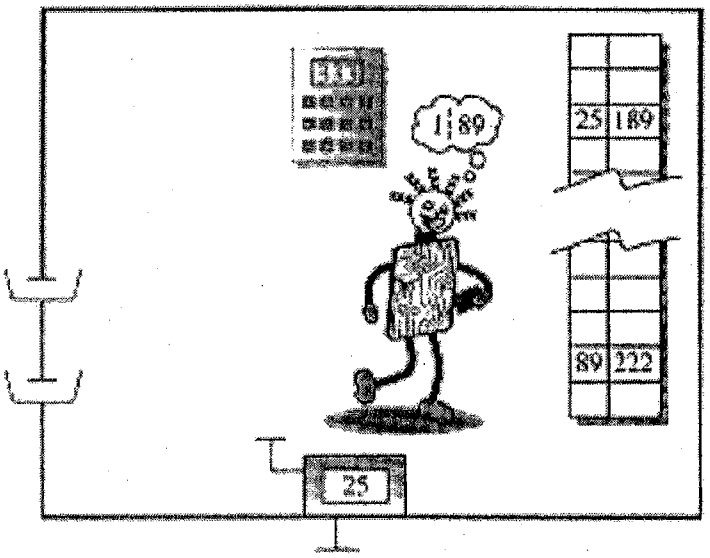
Αρ. Θυρίδας	Mnemonic	Address	OPCODE	Παρατηρήσεις
00	IN		5 00	Διάβασε το Α' αριθμό από το καλάθι, και γράψε το στο κομπιουτεράκι
01	STA	11	2 11	Αποθήκευσε τον αριθμό που είναι στο κομπιουτεράκι, στη θυρίδα 11
02	IN		5 00	Διάβασε το Β' αριθμό από το καλάθι, και γράψε το στο κομπιουτεράκι
03	STA	12	2 12	Αποθήκευσε τον αριθμό που είναι στο κομπιουτεράκι, στη θυρίδα 12
04	SUB	11	4 11	Αφαίρεσε από το δεύτερο αριθμό, που βρίσκεται ήδη στο κομπιουτεράκι τον αριθμό που υπάρχει στη θυρίδα 11
05	OUT		6 00	Δώσε το αποτέλεσμα έξω
06	HLT		7 00	Σταμάτα.



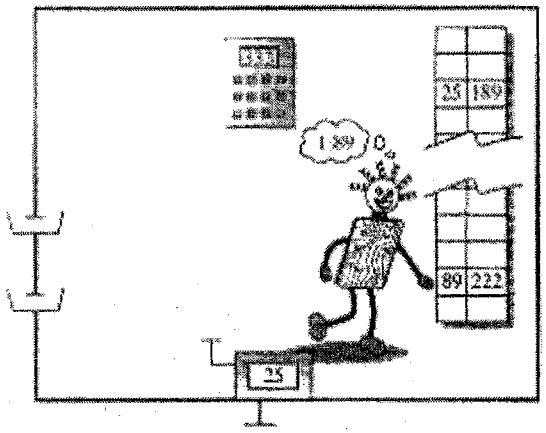
(1) The Little Man reads the address from the location counter



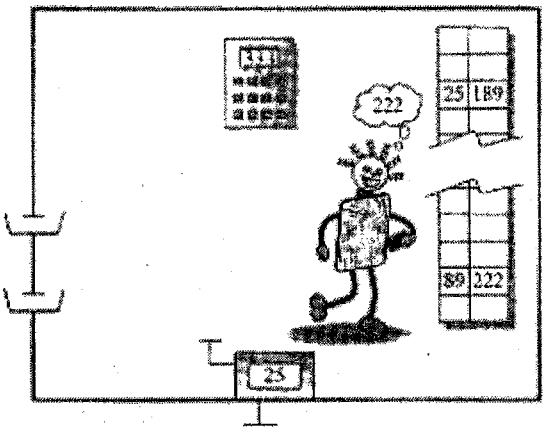
(2) . . . walks over to the mailbox that corresponds to the location counter



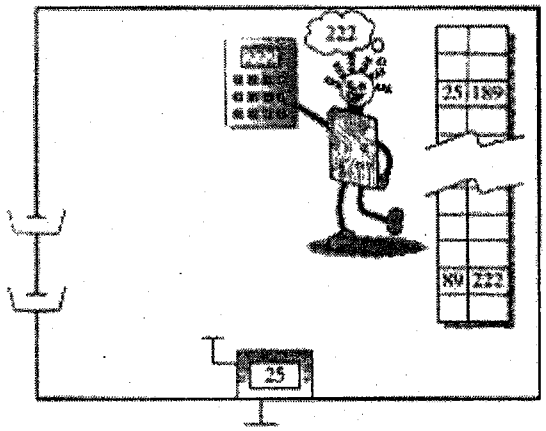
(3) . . . and reads the number on the slip of paper. (He then puts the slip of paper back, in case he should need to read it again later.)



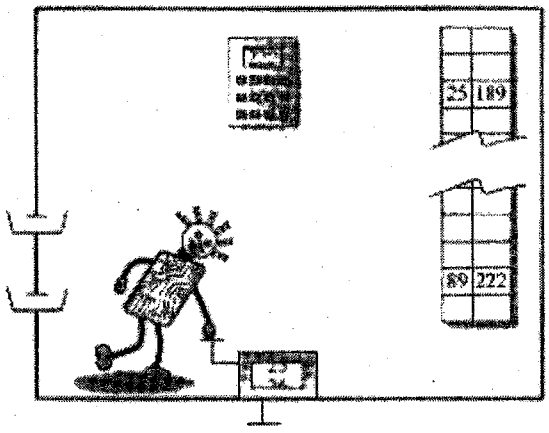
(1) The Little Man goes to the mailbox address specified in the instruction he previously latched



(2) . . . he reads the number in that mailbox (he remembers to replace it in case it's needed again)



(3) . . . he walks over to the calculator and punches the number in



(4) . . . finally, he walks over to the location counter and clicks it, which gets him ready to fetch the next instruction.

Μηχανή Von Neumann

Ο Von Neumann από το 1951 προσδιόρισε τις βασικές αρχές αρχιτεκτονικής των σημερινών υπολογιστών που παραμένουν ίδιες μέχρι σήμερα.

- Στη μνήμη αποθηκεύονται πρόγραμμα και δεδομένα
- Η μνήμη έχει διαδοχικές διευθύνσεις. Κάθε byte στη μνήμη έχει τη δική του διεύθυνση. Οι διευθύνσεις είναι διαδοχικές και συνεχόμενες.
- Η μνήμη "ονομάζεται" - (addressed) από ένα αριθμό, την διεύθυνσή της, ανεξάρτητα από τι δεδομένα (ή εντολή προγράμματος) περιέχει.
- Οι εντολές εκτελούνται σειριακά, η μία μετά την άλλη, εκτός αν υπάρχει συγκεκριμένη εντολή αλλαγής (π.χ SKIP, JUMP κλπ.)
- Οι βασικές λειτουργικές μονάδες ενός υπολογιστή είναι η control unit, η arithmetic/logic unit, η memory, και η I/O

The Little Man Computer (LMC)

From Englander's Book

1 - Layout of the Little Man Computer [A]

You may prefer to think of the *mailboxes* as *pigeonholes*. He has 100 pigeonholes numbered 00 to 99. Notice that the little man can only deal with 3 digit numbers, giving a range of 000 to 999 (unsigned).

2 - Operation of the LMC [A]

Note the format of an instruction for the little man. The first digit, the *op code* (operation code), tells him what action to carry out; the remaining two digits, the *operand,* give the address of a pigeonhole to use.

Note that some instructions do not use pigeonholes. Summary of op codes:

```
1nn  Load the calculator with the number from pigeonhole nn
2nn  Store the number from the calculator in pigeonhole nn
3nn  Add the number in pigeonhole nn to the number in the calculator
4nn  Subtract the number in pigeonhole nn from the number in the calculator
5xx  Input (Read) - Load the calculator with the number from the Input
     basket
6xx  Output (Write) - Put the number from the calculator in the Output
     basket
7xx  Halt (Coffee Break)
```

3 - A Simple Program [A]

Remember that, as well as accessing the input and output baskets, the user can also reset the two digit hand counter to 00. This section is saying that somehow the program, repeated below, is loaded into pigeonholes 00 to O5. The user then punches the Reset button, which starts the program. The little man looks at the counter and. *fetches* the instruction from the pigeonhole of that number, he then *executes* it and *fetches* the next instruction. He goes on doing that until he reaches the Halt instruction:

Pigeonhole (Address)	Contents Instructio n	Description (Mnemonic)	Execution
00	500	In	Take number from input basket and key it into calculator. Increment the counter (to O 1) and fetch the next instruction from that address
01	299	Sta 99	Take the number in the calculator and store it in pigeonhole 99. Increment the counter (to 02) and fetch the next instruction from that address
02	500	In	Take number from input basket and key it into calculator (replacing the first number). Increment the counter (to 03) and fetch the next instruction from that address
03	399	Add 99	Get the number from pigeonhole 99 and add it to the number in the calculator Increment the counter (to 04) and fetch the next instruction from that address
04	600	Out	Take the number from the calculator and place it in the Output basket. Increment counter (to OS) and fetch the next instruction from that address
05	700	Cob	Halt (Coffee Break)

Notice the Fetch / Execute Cycle (see also section 5):
The counter is incremented and the instruction is fetched from the address in the counter.
The op code of the instruction is executed.
When the execution is complete the next instruction is fetched.

ΣΤΟΙΧΕΙΑ ΑΡΙΘΜΗΤΙΚΗΣ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΡΑΓΜΑΤΙΚΟΙ ΑΡΙΘΜΟΙ

Παράσταση με τη Μέθοδο Σταθεράς Υποδιαστολής (Fixed Point)

Αν το πλήθος των ψηφίων του ακέραιου μέρους ενός πραγματικού αριθμού A είναι n και το πλήθος των ψηφίων του κλασματικού του μέρους είναι κ, τότε ο πραγματικός αριθμός A, θα παριστάνεται ως εξής:

$A = d_{n-1}\, d_{n-2} \ldots d_1\, d_0 \,.\, d_{-1}\, d_{-2} \ldots d_{-k}$

Η ερμηνεία της παράστασης (η αξία δηλαδή του αριθμού) αποδίδεται από τη δυναμοσειρά:

$A = d_{n-1} \cdot b^{n-1} + d_{n-2} \cdot b^{n-2} \ldots + d_1 \cdot b^1 + d_0 \cdot b^0 \,.\, + d_{-1} \cdot b^{-1} + d_{-2} \cdot b^{-2} + \ldots + d_{-k} \cdot b^{-k}$

όπου: b η βάση

n φυσικός αριθμός ≥ 0

κ φυσικός αριθμός ≥ 0

και $d_i \in \{0,1,\ldots,b-1\}$ τα ψηφία του αριθμού

Η ερμηνεία αποδίδεται πιο σύντομα ως εξής:

$$A = \sum_{i=-k}^{n-1} d_i b^i$$

Η μέθοδος σταθεράς υποδιαστολής επιτρέπει επίσης και την παράσταση θετικών και αρνητικών αριθμών με χρήση των συμπληρωμάτων τους ως προς τη βάση b και ως προς b-1.

Παραδείγματα

A = 35764.381_{10} και **B** = 04638.700_{10}

Να εκτελεσθούν οι πράξεις:

1. C = A + B
2. D = A - B
3. E = B - A

1. C = A + B

Για τον υπολογισμό του αθροίσματός τους αρκεί να γίνει πρόσθεση. Το μόνο πρόβλημα που μπορεί να προκύψει είναι η υπερχείλιση, δεδομένου ότι το αποτέλεσμα οφείλει να παριστάνεται με 5 ακέραια ψηφία.

```
   35764.381
+  04638.700
--------------
   40403.081
```

Αρα C = A + B = 40403.081

2. D = A - B

Υπολογίζουμε το συμπλήρωμα του B ως προς 10, που παριστάνει τον αντίθετο του B, δηλαδή τον αριθμό -B και κάνουμε πρόσθεση.

$_{10}B_c = 10^5 - B = 100000.000 - B$

```
 100000.000
- 04638.700
---------------
  95361.300
```

Η παράσταση 95361.300 εκφράζει τον αντίθετο του αριθμού +4638.700, δηλαδή τον αριθμό -4638.700. Εκτελούμε τώρα την πρόσθεση A + (-B). Θα πρέπει να σημειωθεί, ότι στην περίπτωση αυτή οι δύο προσθετέοι είναι ετερόσημοι και δεν υπάρχει πρόβλημα. Απλώς, αν προκύψει κρατούμενο δεν θα ληφθεί υπ' όψη.

```
    35764.381
+   95361.300
-----------------
 [1]31125.681
```

Πραγματικά, το κρατούμενο [1] δεν έχει θέση και αγνοείται και έτσι το αποτέλεσμα είναι ο αριθμός 31125.681
(Επαλήθευση: 35674.381 - 4638.7 = 31125.681).

3. E = B - A
Οπως και στην προηγούμενη άσκηση, υπολογίζουμε το συμπλήρωμα του 35764.381, ως προς 10.

```
100000.000
-35764.381
--------------
 64235.619
```

Στη συνέχεια εκτελούμε την πρόσθεση:

```
    64235.619
+   04638.700
---------------
    68874.319
```

Είναι απλό να διαπιστωθεί ότι το αποτέλεσμα με τη μέθοδο του συμπληρώματος ως προς 10, εκφράζει τον αριθμό - 31125.681.

Μέθοδος παράστασης αριθμών με κινητή υποδιαστολή (floating point)

Γενικά ένας αριθμός A είναι δυνατό να παρασταθεί σ'ένα αριθμητικό σύστημα με βάση b σαν ένα διατεταγμένο ζεύγος:

A= (E, F)

όπου: E: προσημασμένος ακέραιος
F: προσημασμένος κλασματικός, τέτοιος ώστε |F|<1

Η σημασία των E και F είναι:
$A=Fxb^E$ όπου b είναι η βάση του χρησιμοποιούμενου συστήματος.

Ετσι, με b=10 θα είναι:

Σταθερά του Boltzman	0.138×10^{-22}	= (-22, .38000000)
Σταθερά του Plank	0.663×10^{-33}	= (-33, .66300000)
Σταθερά του Avogardo	0.60225×10^{24}	= (24, .60225000)
Μάζα της Γης	0.598×10^{25}	= (25, .59800000)

Σημειώνουμε ακόμη ότι, αν το πλήθος των ψηφίων (ή μήκος) του E και του F πρέπει να είναι σταθερά, όπως στα παραδείγματα, τότε το μήκος του E χαρακτηρίζει το πεδίο των απολύτων τιμών των αριθμών οι οποίοι είναι δυνατόν να παρασταθούν, ενώ το F χαρακτηρίζει την ακρίβεια απόδοσης των αριθμών.

Κανονική Μορφή

Ένας αριθμός, σε παράσταση κινητής υποδιαστολής, θα έχει κανονική μορφή αν το πιο σημαντικό ψηφίο του κλασματικού του μέρους F (δηλαδή το ψηφίο αμέσως δεξιά της υποδιαστολής) είναι μη μηδενικό. Στην περίπτωση αυτή θα συμβαίνει:

$$\frac{1}{b} \le |F| < 1$$

Δύο αριθμοί σε παράσταση κινητής υποδιαστολής και με κανονική μορφή συγκρίνονται πολύ εύκολα. Αρχικά συγκρίνονται οι εκθέτες και ο μεγαλύτερος εκθέτης αντιστοιχεί στο μεγαλύτερο αριθμό. Στην περίπτωση που εκθέτες είναι ίσοι, συγκρίνονται τα κλασματικά μέρη και το μεγαλύτερο προσδιορίζει το μεγαλύτερο αριθμό.

Αν δηλαδή $A_1 = (E_1 , F_1)$ και $A_2 = (E_2 , F_2)$ τότε:

$$E_1 > E_2$$

Αν $A_1 > A_2$ **ή**

$$E_1 = E_2 \text{ και } F_1 > F_2$$

Η ΑΡΙΘΜΗΤΙΚΗ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Παραστάσεις προσημασμένων αριθμών

Μία από τις πολλές διαφορετικές μέθοδους παράστασης προσημασμένων αριθμών, χωρίς τη χρήση διαφορετικού συμβόλου από τα χρησιμοποιούμενα ψηφία είναι η χρησιμοποίηση ενός ψηφίου, που συνήθως βρίσκεται στο αριστερό άκρο του αριθμού.
Το ψηφίο αυτό ονομάζεται στην προκειμένη περίπτωση "bit προσήμου".

Συνήθως, αν το "bit προσήμου" είναι 1, θεωρείται ότι συμβολίζει το πρόσημο "πλην" ενώ αν είναι μηδέν, θεωρείται ότι συμβολίζει το πρόσημο "συν".

Συμπλήρωμα ως προς 1

Το συμπλήρωμα ως προς 1 στην περίπτωση των δυαδικών αριθμών, είναι το συμπλήρωμα ως προς b-1, όπου b είναι η βάση δηλαδή ο αριθμός δύο.

Το "συμπλήρωμα ως προς 1" ενός δυαδικού αριθμού, με n ψηφία, υπολογίζεται αφαιρώντας τον αριθμό από ένα αριθμό με n ίδια ψηφία, κάθε ένα από τα οποία εκφράζει τον αριθμό b-1, δηλαδή το μεγαλύτερο ψηφίο του αριθμητικού συστήματος. Στην περίπτωση του δυαδικού συστήματος το ψηφίο αυτό είναι το 1, ενώ στο δεκαδικό σύστημα είναι το 9.

```
  11111111
- 10101000
 -----------
  01010111
```

Παρατηρούμε επίσης ότι ο ίδιος αριθμός προκύπτει με αντικατάσταση κάθε ψηφίου του δοθέντος αριθμού με το συμπληρωματικό του. Οι δύο αυτοί αριθμοί 10101000 και 01010111 είναι συμπληρωματικοί ως προς ένα.

Αν το πρώτο ψηφίο του αριθμού είναι 0 ($d_{n-1} = 0$), τότε ο αριθμός θα εξαρτάται μόνο από τα υπόλοιπα ψηφία του και θα είναι:

1. Μηδέν, αν όλα τα υπόλοιπα ψηφία του είναι μηδέν.
   Πραγματικά η αξία του αριθμού 00000000 είναι:
   0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0
2. Θετικός, αν ένα ή περισσότερα από τα υπόλοιπα ψηφία του είναι 1, που είναι προφανές.

Αν όμως το πρώτο ψηφίο του αριθμού είναι 1, τότε ο αριθμός θα είναι:

1. **Μηδέν αν όλα τα υπόλοιπα ψηφία του αριθμού είναι 1**.Πραγματικά, η ερμηνεία του αριθμού 11111111 είναι η ακόλουθη:
   -127 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = -127 + 127 = 0
2. **Αρνητικός αν ένα ή περισσότερα από τα υπόλοιπα ψηφία του είναι 0**, που είναι προφανές.

(Αριστερότερο bit -> "bit προσήμου" -
"μηδέν" -> θετικός αριθμός,
"ένα" -> αρνητικός).

Ο αριθμός μηδέν έχει τις ακόλουθες δύο διαφορετικές παραστάσεις.
0...00 = +0
1...11 = -0
Τέλος, αν προσθέσουμε σ'ένα αρνητικό τον αντίθετό του, το αποτέλεσμα είναι πάντα το -0 (πλην μηδέν) δηλαδή η παράσταση 11111111.

Παράδειγμα
Σύμφωνα με τη μέθοδο αυτή, σε παράσταση 8 ψηφίων (8 bit), ο δεκαδικός αριθμός 100 γράφεται:

01100100

Πραγματικά ο δυαδικός αριθμός 01100100 ερμηνεύεται ως εξής:

$01100100_2 = 0x(128-1)+64+32+0+0+4+0+0 = 100_{10}$

Σύμφωνα με τα προηγούμενα, το συμπλήρωμα του αριθμού 01100100 ως προς 1 είναι 10011011 και θα πρέπει να αντιστοιχεί στο δεκαδικό αριθμό -100.

Πραγματικά ο δυαδικός 10011011 ερμηνεύεται ως εξής:

$10011011_2 = 0 \chi (128-1)+0+0+16+8+0+2+1 = -101_{10}$

Τα παραδείγματα που ακολουθούν δείχνουν πως πραγματοποιείται η πράξη της πρόσθεσης δύο δυαδικών αριθμών, που είναι εκφρασμένοι με τη μέθοδο του συμπληρώματος ως προς 1. Χρησιμοποιούνται παραστάσεις 8 bit.

Παράδειγμα 1

Εστω οι δεκαδικοί αριθμοί 35, και 26. Ο δεκαδικός αριθμός 35 παριστάνεται στο δυαδικό σύστημα με την ακολουθία ψηφίων 00100011. Ομοίως ο 26 παριστάνεται με 00011010.

δεκαδικοί	αντίστοιχοι δυαδικοί
35	00100011
26	00011010

Προσθέτουμε τους δύο αριθμούς και στα δύο συστήματα.

```
   35            00100011
 + 26          + 00011010
-------------------------
   61            00111101
```

Πραγματικά ο αριθμός 00111101 αντιστοιχεί στο δεκαδικό 61.

Παράδειγμα 2

Εστω οι δεκαδικοί αριθμοί 35 και -26. Σύμφωνα με τα προηγούμενα θα ισχύουν οι αντιστοιχίες:

δεκαδικοί	αντίστοιχοι δυαδικοί
26	00011010
35	00100011
-26	11100101

Εκτελούμε τις πράξεις:

δεκαδικοί	αντίστοιχοι δυαδικοί
35	00100011
- 26	11100101
9	[1]00001000

Ενώ το αποτέλεσμα θα έπρεπε να είναι 9, εμφανίζεται να είναι 8.

Αυτό συμβαίνει εξ αιτίας του κρατουμένου, που δεν έχει θέση στο διατιθέμενο χώρο (που είναι 8 ψηφία για την περίπτωση) και χάνεται. Στις περιπτώσεις που υπάρχει κρατούμενο, θα πρέπει να πραγματοποιείται διόρθωση, ως εξής. Το κρατούμενο θα προστίθεται στο τελευταίο ψηφίο του αριθμού που έχει προκύψει. Δεν το προσθέτουμε στον αριθμό, αλλά στο τελευταίο ψηφίο, ώστε να καλύπτονται και περιπτώσεις των κλασματικών, με σταθερά υποδιαστολή. Είναι φανερό ότι αν πρόκειται περί ακεραίων η διάκριση δεν έχει διαφορά.

Ετσι, όποτε δεν υπάρχει κρατούμενο, τότε το αποτέλεσμα είναι σωστό και δεν χρειάζεται μετατροπή. Αν όμως υπάρχει κρατούμενο, το οποίο δεν μπορεί να καταχωρηθεί στο διατιθέμενο

χώρο των ψηφίων της παράστασης, τότε προστίθεται στο τελευταίο ψηφίο του αριθμού και προκύπτει το σωστό αποτέλεσμα.

Παραδείγματα

```
1. -35   →    11011100
   +26   →    00011010
   --------------------
    -9   →    11110110    ( Συμπλ. ως προς 1: 00001001 ->9)

2. -35   →    11011100
    -26  →    11100101
   --------------------
    -61  → [1]11000001
                   + 1
   --------------------
                11000010    ( Συμπλ. ως προς 1: 00111101 -> 61)
```

Συμπλήρωμα ως προς δύο

Το συμπλήρωμα ως προς 2 ενός δυαδικού αριθμού, με n ψηφία, είναι το συμπλήρωμα ως προς τη βάση του δυαδικού συστήματος. Ο υπολογισμός του συμπληρώματος ως προς 2 ενός δυαδικού αριθμού, με n ψηφία, πραγματοποιείται με δύο διαφορετικού τρόπους, ως εξής:

1. Υπολογίζεται το συμπλήρωμα του αριθμού ως προς 1 και προστίθεται στο τελευταίο ψηφίο του συμπληρώματος το 1,

   "Reverse the Bits, then add 1"

   ή εναλλακτικά

2. Αφαιρείται ο αριθμός από τον 2^n και το αποτέλεσμα που προκύπτει είναι το ζητούμενο συμπλήρωμα.

Παράδειγμα:

+65	01000001	Ο αριθμός	
	10111110	Αντιστροφή bits	Συμπλ. ως προς 1
	1	Πρόσθεση 1	
-65	10111111		Συμπλ. ως προς 2

Ο δυαδικός αριθμός 10111111, είναι ο αριθμός -65, εκφρασμένος με 8 ψηφία σε συμπλήρωμα του 2. Το αριστερότερο ψηφίο δείχνει ότι αυτός είναι αρνητικός. Για να βρούμε το μέτρο του (την αξία του), πρέπει απλά να βρούμε το συμπλήρωμα ως προς 2 του αριθμού αυτού.

Δηλαδή:

65	10111111		Συμπλ. ως προς 2
	01000000	Αντιστροφή bits	Συμπλ. ως προς 1
	1	Πρόσθεση 1	
+65	01000001		Ο αριθμός

Αν προσθέσουμε

```
   01000001  |   +65
   10111111  |   -65
  ----------------------
  [1]00000000|    00
```

Προσέξτε ότι το [1] (κρατούμενο) αγνοείται. **Προσέξτε επίσης ότι το κρατούμενο σε μια πράξη αγνοείται όταν στην διάρκεια της συγκεκριμένης πράξεως έχει προέλθει και σημείο προσήμου (sign bit)!!.**

Παράδειγμα:

Το συμπλήρωμα του 00101101 ως προς 2, με 8 ψηφία, υπολογίζεται:

1. 00101101 ← Ο αριθμός
   11010010 ← Το συμπλήρωμα ως προς 1
   \+ 1

   11010011 ← Το συμπλήρωμα ως προς 2

2. Σχηματισμός του 2^8 = 100000000 και αφαίρεση του αριθμού αυτό.

   100000000 ← 2^8
   - 00101101 ← Ο αριθμός

   11010011 ← Το συμπλήρωμα ως προς 2

Το άθροισμα ενός αριθμού και του συμπληρώματός του ως προς 2, σε παραστάσεις με n ψηφία, είναι μηδέν.

Παράδειγμα

Χρησιμοποιούμε τους δύο συμπληρωματικούς αριθμούς ως προς 2, του προηγουμένου παραδείγματος, στα οκτώ ψηφία:

```
   00101101
  +11010011
 ------------
 [1]00000000
```

Το κρατούμενο δεν έχει θέση στα 8 ψηφία και δε λαμβάνεται υπ'όψη.

Το σημαντικό της μεθόδου του συμπληρώματος ως προς 2 είναι ότι, το αριστερότερο bit, εκτός του ότι προσδιορίζει το πρόσημο, αποτελεί ταυτόχρονα και μέρος του αριθμού. Η μέθοδος είναι παρόμοια με τη μέθοδο του συμπληρώματος ως προς 1, χωρίς όμως τα ίδια προβλήματα.

Παραδείγματα

Σύμφωνα με τα ανωτέρω, οι ακόλουθες παραστάσεις σε 8 ψηφία, θα έχουν τις αντίστοιχες ερμηνείες:

```
10101100 = -2^7  +  0  + 2^5+  0 + 2^3 + 2^2 + 0 + 0 =
         = -128 +      32 +      8 + 4
                                                = - 84

00000000 =    0 +  0 +  0 +  0 + 0 + 0 + 0 + 0 =    0
10000000 = -128 +  0 +  0 +  0 + 0 + 0 + 0 + 0 = -128
11111111 = -128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = -  1
01111111 =    0 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = +127
```

- Παρατηρούμε ότι με τη μέθοδο του συμπληρώματος ως προς 2, το **μηδέν έχει μια μόνο παράσταση**.
- Παρατηρούμε επίσης ότι **η απόλυτη τιμή του μικρότερου αρνητικού αριθμού είναι μεγαλύτερη κατά 1 της μεγαλύτερης θετικής τιμής**.
- Ετσι σε παραστάσεις αριθμών με 8 ψηφία, ο μικρότερος αρνητικός είναι ο **-128**, ενώ ο μεγαλύτερος θετικός αριθμός είναι ο **+127**.
- Σε παραστάσεις αριθμών με 16 ψηφία ο μικρότερος αρνητικός αριθμός είναι ο -32768 ενώ ο μεγαλύτερος θετικός ο +32767.

Η αριθμητική, όπως φαίνεται από τα παραδείγματα που ακολουθούν, είναι πολύ απλή και δεν χρειάζεται διόρθωση όπως στην περίπτωση του κρατούμενου με τη μέθοδο του συμπληρώματος ως προς 1.

Παραδείγματα

Εξετάζουμε τις ακόλουθες πράξεις, με αριθμούς σε παραστάσεις 8 ψηφίων.

1. $43_{10} + 34_{10}$

```
 43 --> 00101011
+34 --> 00100010
--- --------
 77 --> 01001101
```

2. $45_{10} - 19_{10}$

```
 19 --> 00010011  Ο αριθμός
-19 --> 11101101 Το συμπλήρωμα ως προς 2

 45 --> 00101101
-19 --> 11101101
--- --------
 26  [1]00011010 = 26, σωστό αποτέλεσμα
```

Το κρατούμενο δεν έχει θέση στα 8 ψηφία και δεν λαμβάνεται υπ' όψη.

3. $-75_{10} - 27_{10}$

```
 75 --> 01001011 Ο αριθμός
-75 --> 10110101 Το συμπλήρωμα ως προς 2

 27 --> 00011011 Ο αριθμός
-27 --> 11100101 Το συμπλήρωμα ως προς 2

-75 --> 10110101
-27 --> 11100101
-----------------
-102 [1]10011010 = -102, σωστό αποτέλεσμα
```

(Εφ'όσον ο αριθμός που βρήκαμε είναι αρνητικός (sign bit) σε παράσταση συμπληρώματος ως προς 2, για να βρούμε την αξία του, αφ' ενός παραβλέπουμε το κρατούμενο και αφ' ετέρου βρίσκουμε το $_2A_C$, δηλ. το συμπλήρωμα ως προς 2 του 10011010 είναι

```
1001 1010

0110 0101
       +1
---------
0110 0110  => 102 άρα -102
```

Όπως και στο προηγούμενο παράδειγμα, το κρατούμενο δεν έχει θέση στα 8 ψηφία και δεν λαμβάνεται υπ' όψη).

- Αν η πρόσθεση έχει γίνει μεταξύ δύο θετικών αριθμών και υπάρχει κρατούμενο, τότε αυτό μετατρέπει το άθροισμα σε αρνητικό.
- Αν επίσης οι δύο προσθεταίοι είναι αρνητικοί και προκύψει κρατούμενο, τότε αυτό μετατρέπει το άθροισμα σε θετικό.
- Στην περίπτωση που έχουμε να προσθέσουμε δύο ετερόσημους αριθμούς ή να αφαιρέσουμε δύο ομοσήμους, δεν υπάρχει πρόβλημα και το αποτέλεσμα είναι πάντα σωστό.

- Όταν όμως πρόκειται να προστεθούν δύο ομόσημοι αριθμοί, ή να αφαιρεθούν δύο ετερόσημοι, τότε είναι δυνατό να υπάρχει κρατούμενο. Το κρατούμενο στην περίπτωση αυτή μεταφέρεται στη θέση του ψηφίου n-1, με συνέπεια λανθασμένο αποτέλεσμα, λόγω υπερχείλισης.

Παραδείγματα

1. **Να προστεθούν οι δεκαδικοί αριθμοί 109 και 35, αφού μετατραπούν σε δυαδικές παραστάσεις, με τη μέθοδο συμπληρώματος ως προς 2.**

```
Decimal     Binary (2Ac presentation)
  109 -->   01101101
   35 -->   00100011
 -------------------
  144       10010000 = -112
```

(μην ξεχνάμε το $_2A_C$, του 1001 0000, δηλ. 0111 000=>112)

Το λάθος οφείλεται στο γεγονός ότι, ο μεγαλύτερος ακέραιος που μπορεί να παρασταθεί στα 8 ψηφία είναι ο +127.

2. **Να προστεθούν οι δεκαδικοί αριθμοί -119 και -59, αφού μετατραπούν σε δυαδικές παραστάσεις, με τη μέθοδο συμπληρώματος ως προς 2.**

```
- 119       10001001
-  59       11000101
 ----    ------------
 -178    [1]01001110   =  78
```

- Βλέπουμε ότι όταν το άθροισμα δύο αρνητικών είναι μικρότερο του μικρότερου αρνητικού που μπορεί να παρασταθεί με n ψηφία (στα 8 ψηφία είναι -128) και το αποτέλεσμα είναι λανθασμένο.
- Γενικά αν το αποτέλεσμα είναι:
  είτε μικρότερο από -2^{n-1}
  είτε μεγαλύτερο από $2^{n-1} - 1$,

τότε δεν είναι δυνατό να παρασταθεί με n ψηφία και το αποτέλεσμα είναι λανθασμένο.

Συμπλήρωμα ως προς δύο και Σταθερά Υποδιαστολή

Η μέθοδος παράστασης δυαδικών αριθμών με το συμπλήρωμα ως προς 2 επιτρέπει και την παράσταση των κλασματικών αριθμών, με σταθερά υποδιαστολή.

Παραδείγματα

Δίδεται δεκαδικός αριθμός 46.625 και ζητείται ο αντίθετός του, σε δυαδική μορφή, με τη μέθοδο συμπληρώματος ως προς 2, σε παράσταση 8 ακεραίων και τριών κλασματικών ψηφίων.

Κατά τα γνωστά, ο αντίστοιχος δυαδικός είναι ο

```
00101110.101
```

Ισχύει λοιπόν ότι:

$$00101110.101_2 = 46.625_{10}$$

Σύμφωνα με τα προηγούμενα, υπολογίζουμε το συμπλήρωμα του δυαδικού ως προς 1:

```
00101110.101      Ο αριθμός
11010001.010      Το συμπλήρωμα ως προς 1
         +1       Προσθέτουμε στο τελευταίο ψηφίο, όχι αριθμό
------------
11010001.011      Το συμπλήρωμα ως προς 2
```

Ερμηνεία του συμπληρώματος ως προς 2:

```
11010001.011 =
  1     1 0 1  0 0 0 1 . 0   1     1
-128 + 64+0+16+0+0+0+1 + 0+ 1/4 + 1/8 =
-128 + 81 + 3/8 =
    -47 + 0.375 =          -46.625
```