Βασικές αρχές οργάνωσης & λειτουργίας υπολογιστικών συστημάτων

Χαρίδημος Θ. Βέργος

Καθηγητής
Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών

Τα Τυπικά

- Ώρες και διάρθρωση
  - 3 ώρες παράδοση
  - 1 ώρα φροντιστήριο
  - 13 εβδομάδες
  - Ενεργή συμμετοχή
- Ύλη: Ότι καλυφθεί στο αμφιθέατρο
- Ώρες Γραφείου: Καθημερινά 08.30 - 13.00 & 17.00 - 21.00
- E-mail: vergos@upatras.gr
- Ιστοσελίδα μαθήματος: http://pc-vlsi18.ceid.upatras.gr/introduction_to_computer_systems.html
- Eclass: https://eclass.upatras.gr/courses/CEID1244/ Password:

Υπολογισμός - Υπολογιστής

- Υπολογισμός: Εύρεση $f(x)$, δοθέντων f και x.

Υπολογισμός - Υπολογιστής

- Υπολογισμός: Εύρεση $f(x)$, δοθέντων f και x.
- Υπολογιστής: Ο εκτελών τον υπολογισμό.

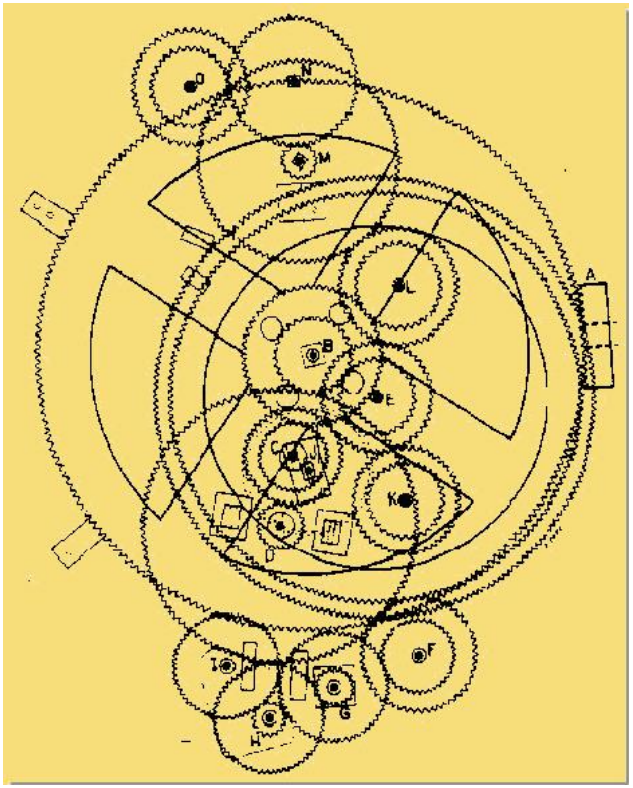
Υπολογισμός - Υπολογιστής

- Υπολογισμός: Εύρεση $f(x)$, δοθέντων f και x.
- Υπολογιστής: Ο εκτελών τον υπολογισμό.
- Ηλεκτρονικός: σχεδόν αποκλειστικά από ηλεκτρονικά στοιχεία

Υπολογισμός - Υπολογιστής

- Υπολογισμός: Εύρεση $f(x)$, δοθέντων f και x.
- Υπολογιστής: Ο εκτελών τον υπολογισμό.
- Ηλεκτρονικός: σχεδόν αποκλειστικά από ηλεκτρονικά στοιχεία
- Υπήρξαν και μη ηλεκτρονικοί υπολογιστές ?

Ο πρώτος υπολογιστής : Μηχανισμός Αντικηθύρων



"Άλλη ονομασία που χρησιμοποιείται για τον χαρακτηρισμό αυτού του ευρήματος είναι υπολογιστής. Υπονοείται ότι πρόκειται για ένα μηχανικό υπολογιστή, ο οποίος βάσει των γεωμετρικών σχέσεων των γραναζιών του αναπαράγει διάφορα αστρονομικά φαινόμενα. Προφανώς πρόκειται για ένα αναλογικό υπολογιστή, αφού τα αποτελέσματα που αναφέρονται σε χρονικές στιγμές και περιόδους, προκύπτουν ως αναλογία προς μηχανικά μεγέθη."

Προγραμματισμός (1/2)

- Τι διαφορετικό έχει ο ηλεκτρονικός υπολογιστής από άλλες ηλεκτρονικές συσκευές ?

Προγραμματισμός (1/2)

- Τι διαφορετικό έχει ο ηλεκτρονικός υπολογιστής από άλλες ηλεκτρονικές συσκευές ?
  - Οι υπόλοιπες εκτελούν ένα συγκεκριμένο σύνολο υπολογισμών !
  - Οι ηλεκτρονικοί υπολογιστές επιτρέπουν την αλλαγή της προς υπολογισμό συνάρτησης !
  - Ενγενής ευελιξία (versatility) !

Προγραμματισμός (2/2)

- Πως επιτυγχάνεται αυτή η αλλαγή ?

Προγραμματισμός (2/2)

- Πως επιτυγχάνεται αυτή η αλλαγή ?
  - Ο ηλεκτρονικός υπολογιστής έχει πραγματική δυνατότητα υπολογισμού
    - Εξαιρετικά λίγων
    - Εξαιρετικά απλών (πρωταρχικών - primitives)
    - Με πολύ μικρό χρόνο εκτέλεσης συναρτήσεων

Προγραμματισμός (2/2)

- Πως επιτυγχάνεται αυτή η αλλαγή ?
  - Ο ηλεκτρονικός υπολογιστής έχει πραγματική δυνατότητα υπολογισμού
    - Εξαιρετικά λίγων
    - Εξαιρετικά απλών (πρωταρχικών - primitives)
    - Με πολύ μικρό χρόνο εκτέλεσης συναρτήσεων
  - Ολες οι υπόλοιπες συναρτήσεις εκφράζονται σα σύνθεση (ακολουθία) των απλών συναρτήσεων.

Προγραμματισμός (2/2)

- Πως επιτυγχάνεται αυτή η αλλαγή ?
  - Ο ηλεκτρονικός υπολογιστής έχει πραγματική δυνατότητα υπολογισμού
    - Εξαιρετικά λίγων
    - Εξαιρετικά απλών (πρωταρχικών - primitives)
    - Με πολύ μικρό χρόνο εκτέλεσης συναρτήσεων
  - Ολες οι υπόλοιπες συναρτήσεις εκφράζονται σα σύνθεση (ακολουθία) των απλών συναρτήσεων.
  - Η περιγραφή της ακολουθίας πρωταρχικών συναρτήσεων για τον υπολογισμό μιας σύνθετης συνάρτησης ονομάζεται προγραμματισμός, η ακολουθία πρόγραμμα και ο περιγράφων προγραμματιστής.
  - Επαναχρησιμοποίηση (reusability)

Παράδειγμα Προγραμματισμού

- Εστω ένας υπολογιστής με τις πρωταρχικές συναρτήσεις
  - $f_1(X, Y) = X + Y$
  - $f_2(X, Y) = X - Y$

Παράδειγμα Προγραμματισμού

- Εστω ένας υπολογιστής με τις πρωταρχικές συναρτήσεις
  - $f_1(X, Y) = X + Y$
  - $f_2(X, Y) = X - Y$
- Πως μπορεί να υπολογίσει την $f_3(A, B, C) = 2A - B + 3C$?

Παράδειγμα Προγραμματισμού

- Εστω ένας υπολογιστής με τις πρωταρχικές συναρτήσεις
  - $f_1(X, Y) = X + Y$
  - $f_2(X, Y) = X - Y$
- Πως μπορεί να υπολογίσει την $f_3(A, B, C) = 2A - B + 3C$?
  - $f_3(A, B, C) = (2A) - B + ((2C) + C)$

Παράδειγμα Προγραμματισμού

- Εστω ένας υπολογιστής με τις πρωταρχικές συναρτήσεις
  - $f_1(X, Y) = X + Y$
  - $f_2(X, Y) = X - Y$
- Πως μπορεί να υπολογίσει την $f_3(A, B, C) = 2A - B + 3C$?
  - $f_3(A, B, C) = (2A) - B + ((2C) + C)$
  - $f_3(A, B, C) = (f_1(A, A) - B) + ((f_1(C, C)) + C)$

Παράδειγμα Προγραμματισμού

- Εστω ένας υπολογιστής με τις πρωταρχικές συναρτήσεις
  - $f_1(X, Y) = X + Y$
  - $f_2(X, Y) = X - Y$
- Πως μπορεί να υπολογίσει την $f_3(A, B, C) = 2A - B + 3C$?
  - $f_3(A, B, C) = (2A) - B + ((2C) + C)$
  - $f_3(A, B, C) = (f_1(A, A) - B) + ((f_1(C, C)) + C)$
  - $f_3(A, B, C) = f_2(f_1(A, A), B) + f_1(f_1(C, C), C)$

Παράδειγμα Προγραμματισμού

- Εστω ένας υπολογιστής με τις πρωταρχικές συναρτήσεις
  - $f_1(X, Y) = X + Y$
  - $f_2(X, Y) = X - Y$
- Πως μπορεί να υπολογίσει την $f_3(A, B, C) = 2A - B + 3C$?
  - $f_3(A, B, C) = (2A) - B + ((2C) + C)$
  - $f_3(A, B, C) = (f_1(A, A) - B) + ((f_1(C, C)) + C)$
  - $f_3(A, B, C) = f_2(f_1(A, A), B) + f_1(f_1(C, C), C)$
  - $f_3(A, B, C) = f_1(f_2(f_1(A, A), B), f_1(f_1(C, C), C))$

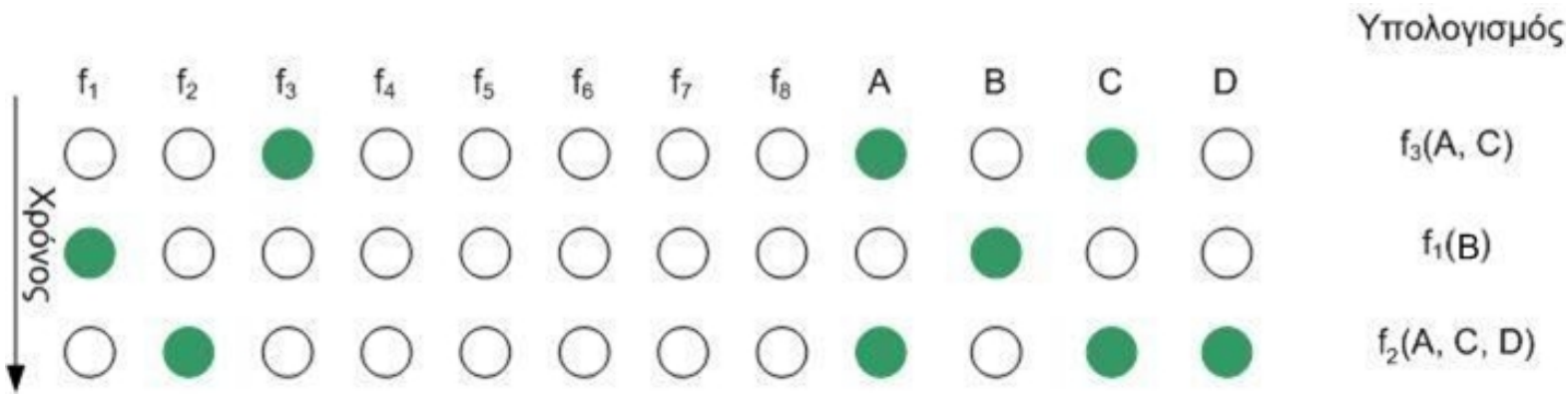
Παράδειγμα Προγραμματισμού

- Εστω ένας υπολογιστής με τις πρωταρχικές συναρτήσεις
  - $f_1(X, Y) = X + Y$
  - $f_2(X, Y) = X - Y$
- Πως μπορεί να υπολογίσει την $f_3(A, B, C) = 2A - B + 3C$?
  - $f_3(A, B, C) = (2A) - B + ((2C) + C)$
  - $f_3(A, B, C) = (f_1(A, A) - B) + ((f_1(C, C)) + C)$
  - $f_3(A, B, C) = f_2(f_1(A, A), B) + f_1(f_1(C, C), C)$
  - $f_3(A, B, C) = f_1(f_2(f_1(A, A), B), f_1(f_1(C, C), C))$
- Οι παρενθέσεις δίνουν σειρά στον τρόπο εφαρμογής των πρωταρχικών συναρτήσεων και συνεπώς ορίζουν μια ακολουθία τους.

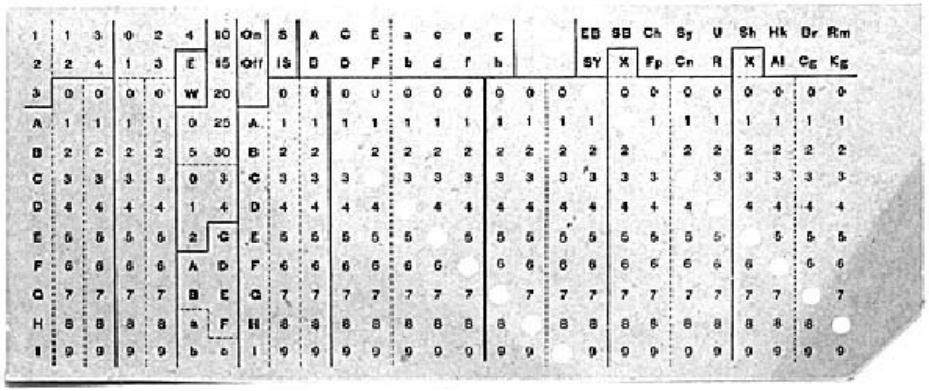
Παράδειγμα Προγραμματισμού

- Εστω ένας υπολογιστής με τις πρωταρχικές συναρτήσεις
  - $f_1(X, Y) = X + Y$
  - $f_2(X, Y) = X - Y$
- Πως μπορεί να υπολογίσει την $f_3(A, B, C) = 2A - B + 3C$?
  - $f_3(A, B, C) = (2A) - B + ((2C) + C)$
  - $f_3(A, B, C) = (f_1(A, A) - B) + ((f_1(C, C)) + C)$
  - $f_3(A, B, C) = f_2(f_1(A, A), B) + f_1(f_1(C, C), C)$
  - $f_3(A, B, C) = f_1(f_2(f_1(A, A), B), f_1(f_1(C, C), C))$
- Οι παρενθέσεις δίνουν σειρά στον τρόπο εφαρμογής των πρωταρχικών συναρτήσεων και συνεπώς ορίζουν μια ακολουθία τους.
- Το αποτέλεσμα μιας πρωταρχικής συνάρτησης αποτελεί τελούμενο (έντελο) για την επόμενη χρονικά.

Προγραμματισμός στους πρώτους υπολογιστές



- Προγραμματισμός = Τοποθέτηση χιλιάδων διακοπτών σε συγκεκριμένες θέσεις
- Μπορούσαν να υπάρχουν λίγα τελούμενα, που εισάγονταν από διάτρητες κάρτες. Ενδιάμεσα αποτελέσματα αποθηκεύονταν πάλι σε κάρτες.



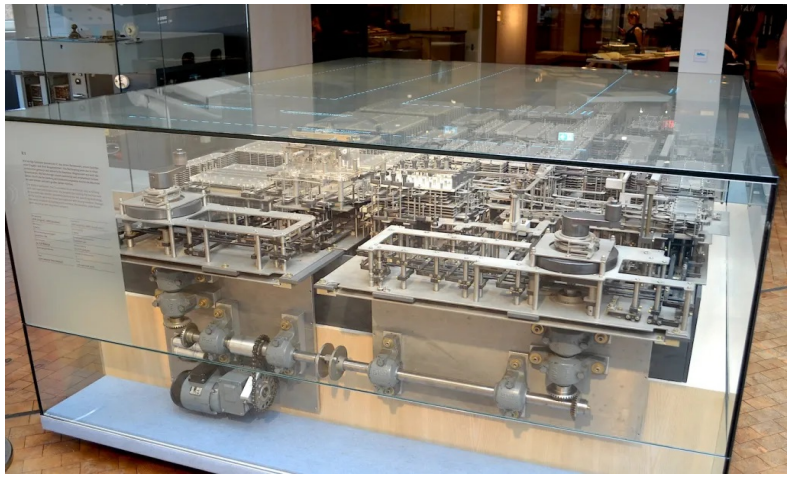
Προβλήματα !

- Η εισαγωγή μεγάλου όγκου δεδομένων ήταν εξαιρετικά επίπονη διαδικασία.
- Η μορφή των αποτελεσμάτων δεν ήταν εύκολη στην επεξεργασία.
- Θα θέλαμε να μπορούμε να "θυμόμαστε" τα ενδιάμεσα αποτελέσματα.
- Κάθε προγραμματισμός χανόταν μόλις κάναμε τον επόμενο.
- Προγραμματισμός επίπονος, χρονοβόρος και αναξιόπιστος.

Προβλήματα !

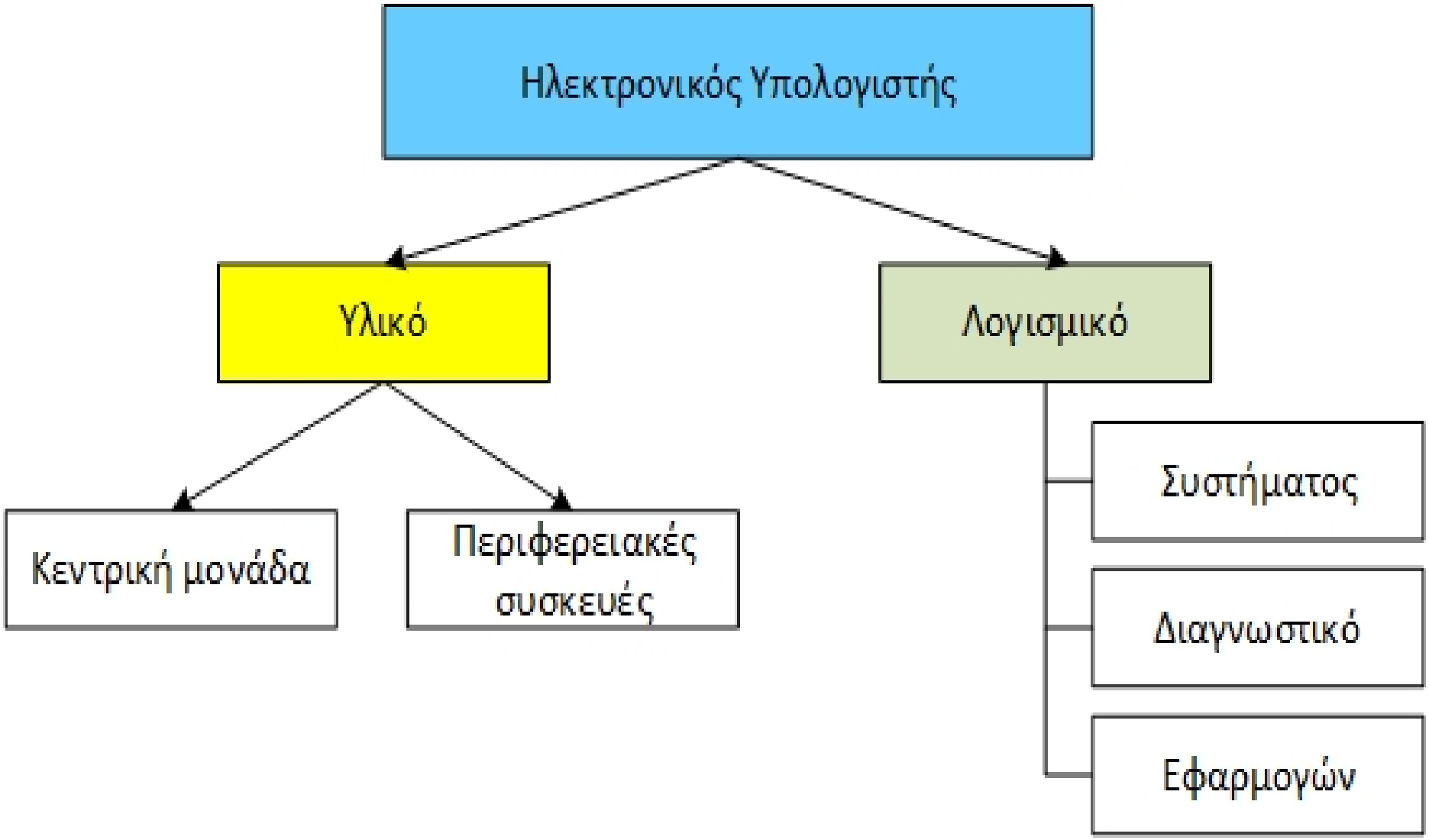
- Η εισαγωγή μεγάλου όγκου δεδομένων ήταν εξαιρετικά επίπονη διαδικασία.
- Η μορφή των αποτελεσμάτων δεν ήταν εύκολη στην επεξεργασία.
- Θα θέλαμε να μπορούμε να "θυμόμαστε" τα ενδιάμεσα αποτελέσματα.
- Κάθε προγραμματισμός χανόταν μόλις κάναμε τον επόμενο.
- Προγραμματισμός επίπονος, χρονοβόρος και αναξιόπιστος.
- Λύση στα περισσότερα προβλήματα: ΜΝΗΜΗ !!!

Konrad Zuse : ο αδικημένος

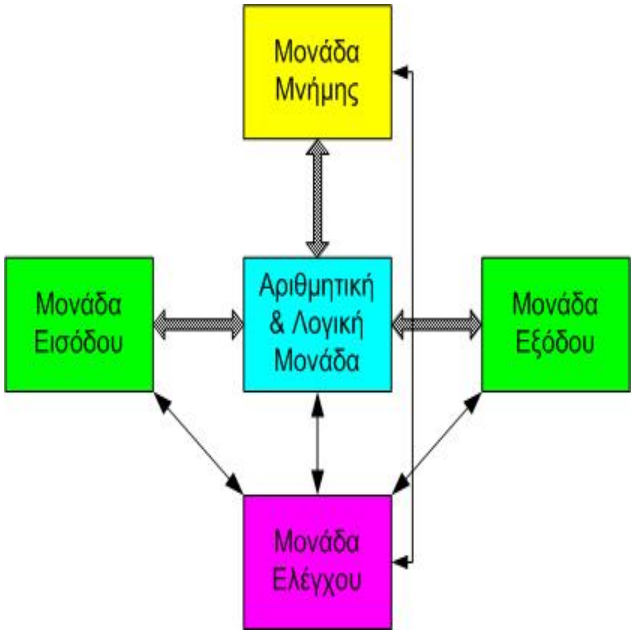


"One of the most difficult aspects of doing a large calculation with either a slide rule or a mechanical adding machine is keeping track of all intermediate results and using them, in their proper place, in later steps of the calculation. Konrad Zuse wanted to overcome that difficulty. He realized that an automatic-calculator device would require three basic elements: a control, a memory, and a calculator for the arithmetic."

Αφαιρετική δομή ενός ηλεκτρονικού υπολογιστή

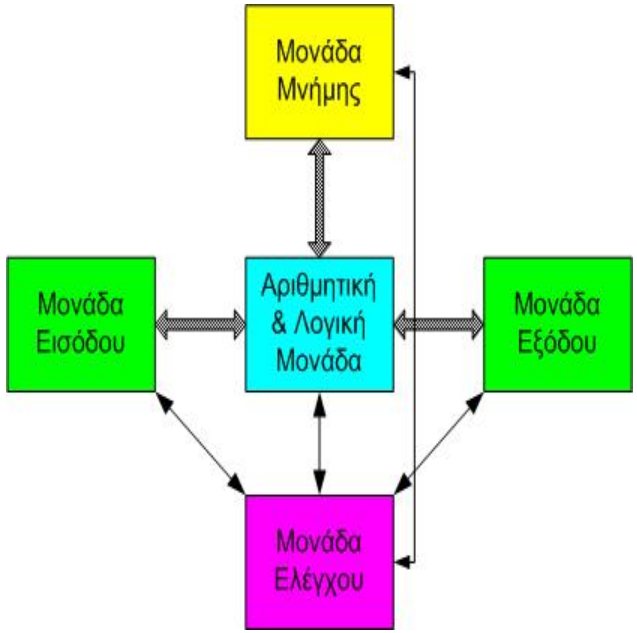


Von Neumann : ο πατέρας ?



- Μονάδα Εισόδου (Input Unit) : Εισάγουμε πληροφορίες (=δεδομένα + προγράμματα).
- Μονάδα Μνήμης (Memory Unit) : Αποθηκεύει τις πληροφορίες και τα ενδιάμεσα αποτελέσματα.
- Αριθμητική Λογική Μονάδα (ALU) : Εκτελεί τις πρωταρχικές συναρτήσεις.
- Μονάδα Ελέγχου (Control Unit) : Συντονίζει (χρονίζει) τις διάφορες υπομονάδες.
- Μονάδα Εξόδου (Output Unit) : Απεικονίζει τα αποτελέσματα.
- ALU + Control = Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ / CPU).

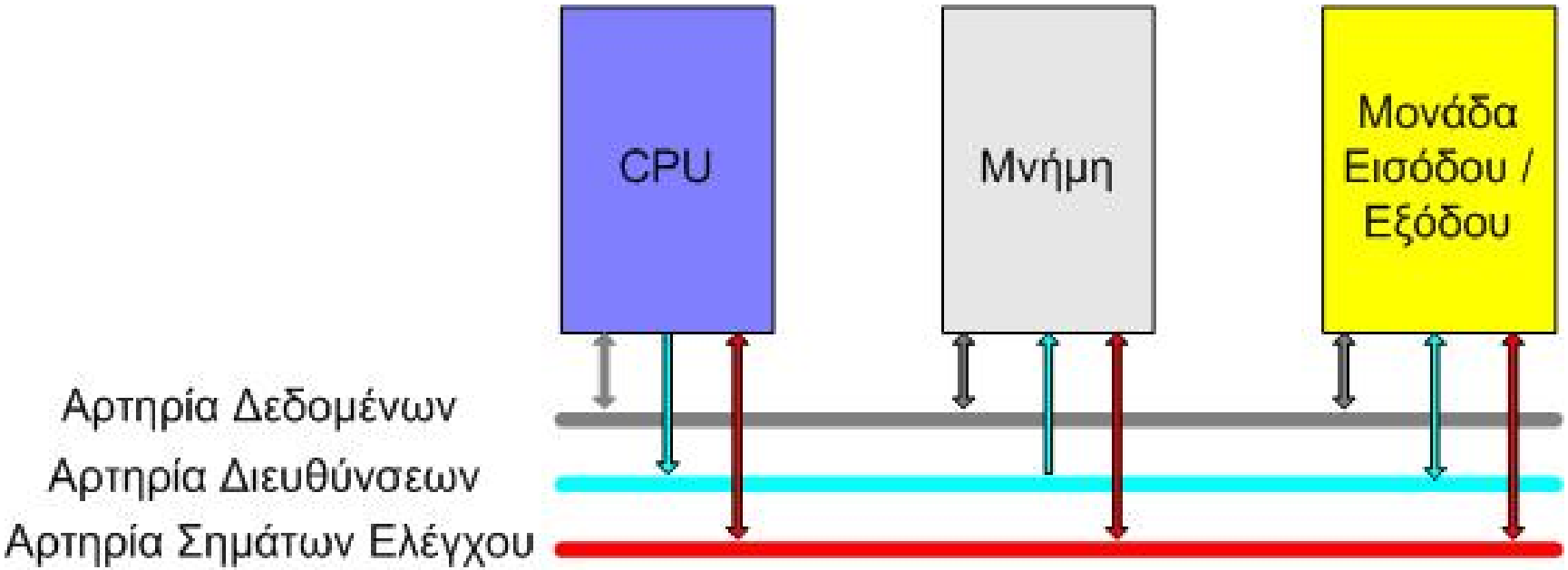
Von Neumann : ο πατέρας ?



- Μονάδα Εισόδου (Input Unit) : Εισάγουμε πληροφορίες (=δεδομένα + προγράμματα).
- Μονάδα Μνήμης (Memory Unit) : Αποθηκεύει τις πληροφορίες και τα ενδιάμεσα αποτελέσματα.
- Αριθμητική Λογική Μονάδα (ALU) : Εκτελεί τις πρωταρχικές συναρτήσεις.
- Μονάδα Ελέγχου (Control Unit) : Συντονίζει (χρονίζει) τις διάφορες υπομονάδες.
- Μονάδα Εξόδου (Output Unit) : Απεικονίζει τα αποτελέσματα.
- ALU + Control = Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ / CPU).

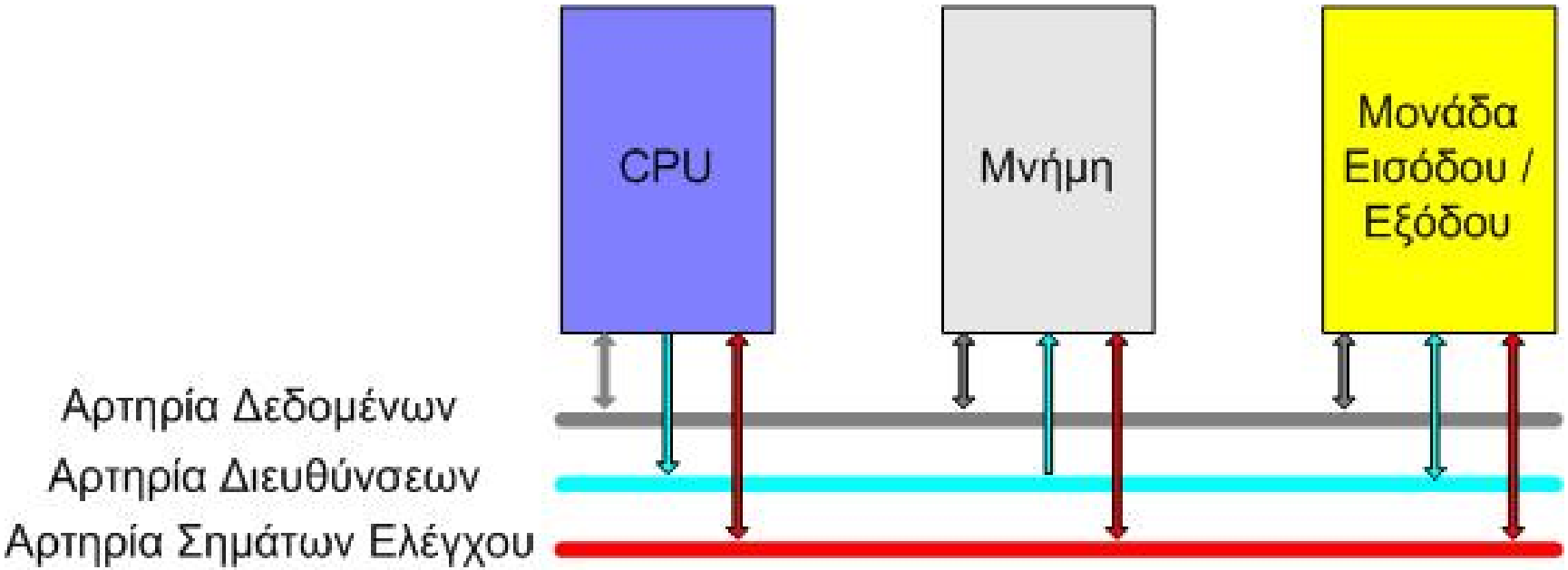
Ενοποίηση προγράμματος & τιμών μεταβλητών σε μια οντότητα : πληροφορία και ενιαίος χειρισμός τους.

Μοντέλο αρτηριών συστήματος



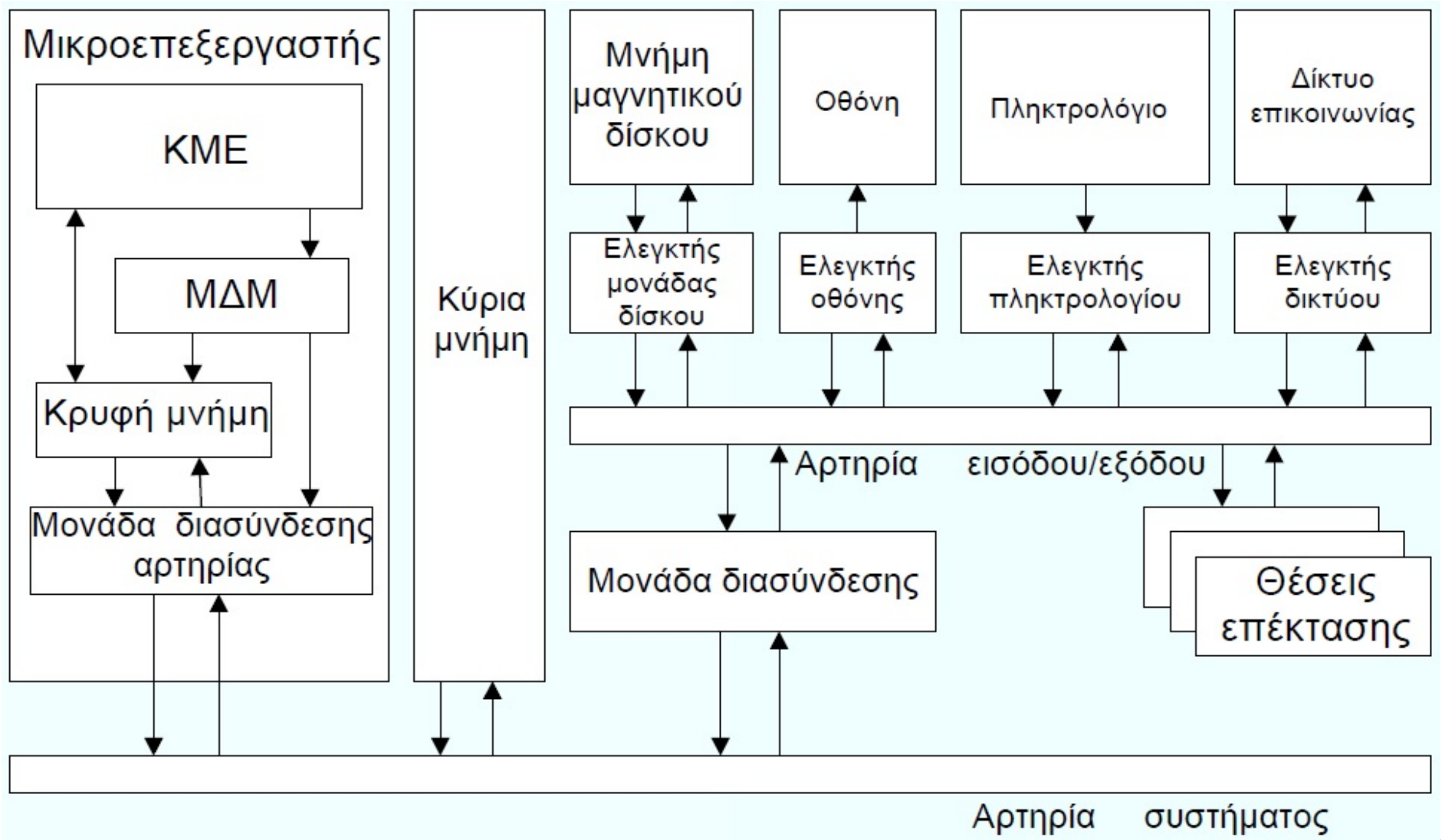
- Οι αρτηρίες είναι οι διάδρομοι μετάδοσης της πληροφορίας.

Μοντέλο αρτηριών συστήματος



- Οι αρτηρίες είναι οι διάδρομοι μετάδοσης της πληροφορίας.
- Κάθε πληροφορία χαρακτηρίζεται μοναδικά βάσει της διευθύνσεώς της.

Πιο συγκεκριμένο μοντέλο ενός προσωπικού υπολογιστή



Ιεραρχία : Γιατί ?

- Τα σημερινά συστήματα είναι εξαιρετικά πολύπλοκα για να αναπτυχθούν μονομιάς σαν αυτοτελείς οντότητες.

Ιεραρχία : Γιατί ?

- Τα σημερινά συστήματα είναι εξαιρετικά πολύπλοκα για να αναπτυχθούν μονομιάς σαν αυτοτελείς οντότητες.
- Ο καταμερισμός σε μικρότερα εξυπηρετεί την συνεργασία ατόμων από διαφορετικές επιστημονικές περιοχές αλλά και βοηθά στον περιορισμό της πολυπλοκότητας.

Ιεραρχία : Γιατί ?

- Τα σημερινά συστήματα είναι εξαιρετικά πολύπλοκα για να αναπτυχθούν μονομιάς σαν αυτοτελείς οντότητες.
- Ο καταμερισμός σε μικρότερα εξυπηρετεί την συνεργασία ατόμων από διαφορετικές επιστημονικές περιοχές αλλά και βοηθά στον περιορισμό της πολυπλοκότητας.
- Ο καταμερισμός δεν είναι μόνο οριζόντιος, αλλά και κάθετος.
- Δημιουργείται έτσι μια ιεραρχία επιπέδων.

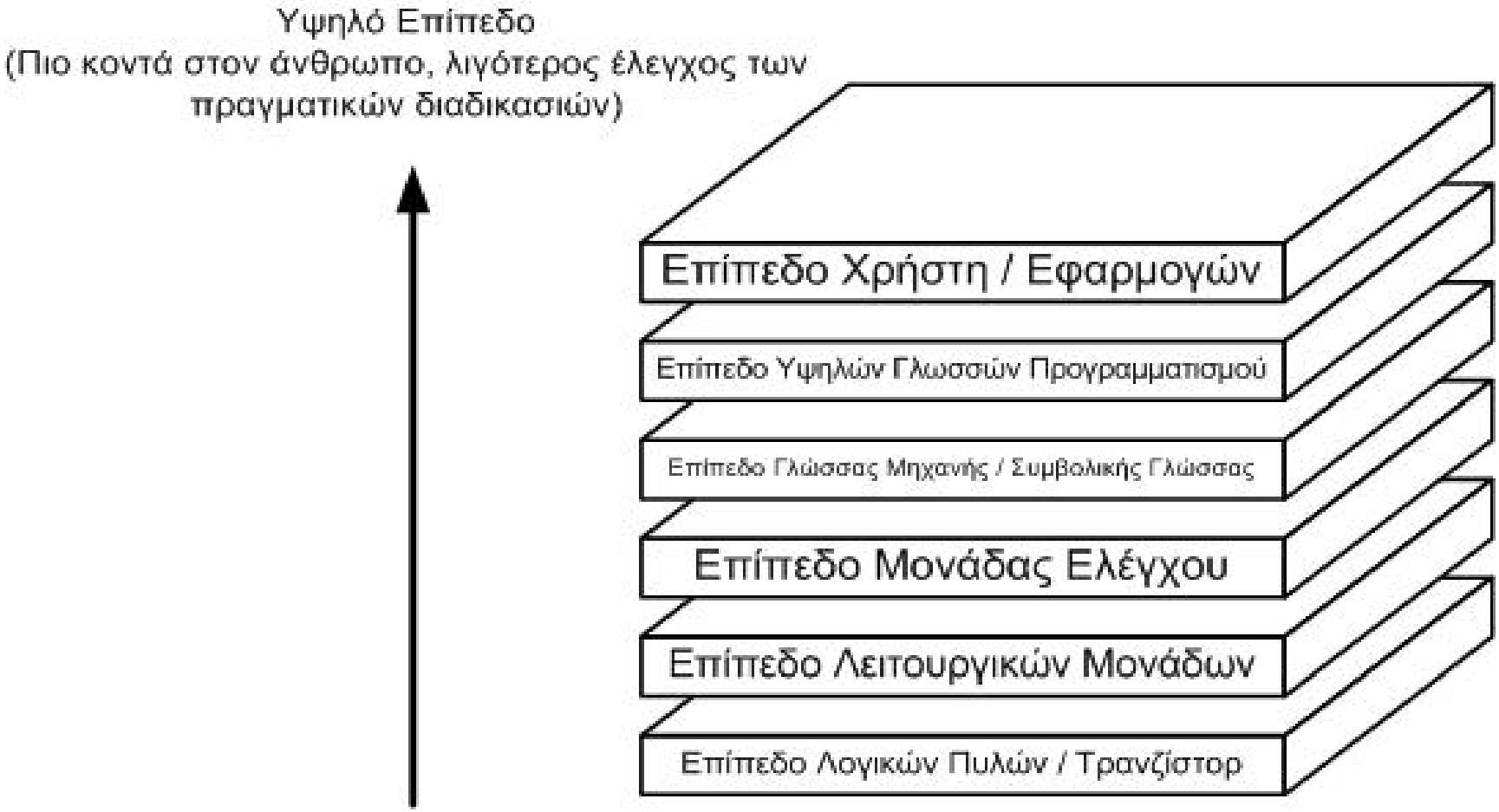
Ιεραρχία : Γιατί ?

- Τα σημερινά συστήματα είναι εξαιρετικά πολύπλοκα για να αναπτυχθούν μονομιάς σαν αυτοτελείς οντότητες.
- Ο καταμερισμός σε μικρότερα εξυπηρετεί την συνεργασία ατόμων από διαφορετικές επιστημονικές περιοχές αλλά και βοηθά στον περιορισμό της πολυπλοκότητας.
- Ο καταμερισμός δεν είναι μόνο οριζόντιος, αλλά και κάθετος.
- Δημιουργείται έτσι μια ιεραρχία επιπέδων.
- Κάθε επίπεδο προσφέρει κάποιες συναρτήσεις (primitive services).
- Το επόμενο επίπεδο της ιεραρχίας μπορεί να χρησιμοποιεί αυτές τις συναρτήσεις για να χτίζει (συνθέτει) πιο πολύπλοκες οι οποίες θα είναι οι δικές του πρωταρχικές !

Ιεραρχία : Γιατί ?

- Τα σημερινά συστήματα είναι εξαιρετικά πολύπλοκα για να αναπτυχθούν μονομιάς σαν αυτοτελείς οντότητες.
- Ο καταμερισμός σε μικρότερα εξυπηρετεί την συνεργασία ατόμων από διαφορετικές επιστημονικές περιοχές αλλά και βοηθά στον περιορισμό της πολυπλοκότητας.
- Ο καταμερισμός δεν είναι μόνο οριζόντιος, αλλά και κάθετος.
- Δημιουργείται έτσι μια ιεραρχία επιπέδων.
- Κάθε επίπεδο προσφέρει κάποιες συναρτήσεις (primitive services).
- Το επόμενο επίπεδο της ιεραρχίας μπορεί να χρησιμοποιεί αυτές τις συναρτήσεις για να χτίζει (συνθέτει) πιο πολύπλοκες οι οποίες θα είναι οι δικές του πρωταρχικές !
- Ο χρήστης συναρτήσεων ενός επιπέδου, σε αντίθεση με τον προγραμματιστή του ίδιου επιπέδου, δε χρειάζεται να γνωρίζει τίποτε για το πως τα πιο κάτω επίπεδα υλοποιούν τις δικές τους συναρτήσεις.

Τα επίπεδα ενός υπολογιστικού συστήματος



Επίπεδο τρανζίστορ - λογικών πυλών

- Τρανζίστορ : Συναρτήσεις μεταξύ δυναμικών και ρευμάτων.
- Λογικές πύλες :
  - Φτιάχνονται από διασύνδεση τρανζίστορ.
  - Υλοποιούν λογικές συναρτήσεις.
- Μαθήματα :
  - Αναλογικά και Ψηφιακά Ηλεκτρονικά.
  - Εισαγωγή σε VLSI.
  - Σχεδιασμός Συστημάτων VLSI.
  - Λογική Σχεδίαση.

Επίπεδο λειτουργικών μονάδων

- Ομάδες από λογικές πύλες οργανώνονται σε μονάδες ικανές να πραγματοποιήσουν πολύ πιο ενδιαφέρουσες συναρτήσεις.
- Συναρτήσεις :
  - Αποθήκευση πληροφορίας.
  - Πραγματοποίηση αριθμητικών πράξεων.
- Συνήθεις λειτουργικές μονάδες :
  - Καταχωρητές - Μνήμες.
  - Αριθμητικές & Λογικές Μονάδες.
  - Κεντρικές Μονάδες Επεξεργασίας.
- Μαθήματα :
  - Λογική Σχεδίαση.
  - Αρχιτεκτονική Υπολογιστών.
  - Μικροεπεξεργαστές.

Επίπεδο μονάδας ελέγχου

- Σκοπός η ανάπτυξη ακολουθιών μεταφοράς πληροφορίας μεταξύ λειτουργικών μονάδων για την υλοποίηση εντολών.

Επίπεδο μονάδας ελέγχου

- Σκοπός η ανάπτυξη ακολουθιών μεταφοράς πληροφορίας μεταξύ λειτουργικών μονάδων για την υλοποίηση εντολών.
- Π.χ.για την υλοποίηση της εντολής ADD : $C = A + B$ χρειάζεται η ακολουθία :

Επίπεδο μονάδας ελέγχου

- Σκοπός η ανάπτυξη ακολουθιών μεταφοράς πληροφορίας μεταξύ λειτουργικών μονάδων για την υλοποίηση εντολών.
- Π.χ.για την υλοποίηση της εντολής ADD : $C = A + B$ χρειάζεται η ακολουθία :
  - Διάβασμα από μνήμη του A και μεταφορά πληροφορίας στον αθροιστή.

Επίπεδο μονάδας ελέγχου

- Σκοπός η ανάπτυξη ακολουθιών μεταφοράς πληροφορίας μεταξύ λειτουργικών μονάδων για την υλοποίηση εντολών.
- Π.χ.για την υλοποίηση της εντολής ADD : $C = A + B$ χρειάζεται η ακολουθία :
  - Διάβασμα από μνήμη του A και μεταφορά πληροφορίας στον αθροιστή.
  - Διάβασμα από μνήμη του B και μεταφορά πληροφορίας στον αθροιστή.

Επίπεδο μονάδας ελέγχου

- Σκοπός η ανάπτυξη ακολουθιών μεταφοράς πληροφορίας μεταξύ λειτουργικών μονάδων για την υλοποίηση εντολών.
- Π.χ.για την υλοποίηση της εντολής ADD : $C = A + B$ χρειάζεται η ακολουθία :
  - Διάβασμα από μνήμη του A και μεταφορά πληροφορίας στον αθροιστή.
  - Διάβασμα από μνήμη του B και μεταφορά πληροφορίας στον αθροιστή.
  - Ενεργοποίηση αθροιστή.

Επίπεδο μονάδας ελέγχου

- Σκοπός η ανάπτυξη ακολουθιών μεταφοράς πληροφορίας μεταξύ λειτουργικών μονάδων για την υλοποίηση εντολών.
- Π.χ.για την υλοποίηση της εντολής ADD : $C = A + B$ χρειάζεται η ακολουθία :
  - Διάβασμα από μνήμη του A και μεταφορά πληροφορίας στον αθροιστή.
  - Διάβασμα από μνήμη του B και μεταφορά πληροφορίας στον αθροιστή.
  - Ενεργοποίηση αθροιστή.
  - Μεταφορά του αποτελέσματος από αθροιστή στη μνήμη / μονάδα εξόδου.

Επίπεδο μονάδας ελέγχου

- Σκοπός η ανάπτυξη ακολουθιών μεταφοράς πληροφορίας μεταξύ λειτουργικών μονάδων για την υλοποίηση εντολών.
- Π.χ.για την υλοποίηση της εντολής ADD : $C = A + B$ χρειάζεται η ακολουθία :
  - Διάβασμα από μνήμη του A και μεταφορά πληροφορίας στον αθροιστή.
  - Διάβασμα από μνήμη του B και μεταφορά πληροφορίας στον αθροιστή.
  - Ενεργοποίηση αθροιστή.
  - Μεταφορά του αποτελέσματος από αθροιστή στη μνήμη / μονάδα εξόδου.
- Μικροεντολή: Κάθε εντολή μεταφοράς πληροφορίας / ενεργοποίησης μιας λειτουργικής μονάδας.
- Μικροπρόγραμμα: Κάθε διαφορετική ακολουθία μικροεντολών.
- Μάθημα: Αρχιτεκτονική Υπολογιστών.

Επίπεδο γλώσσας μηχανής / Συμβολικής γλώσσας (Assembly)

- Κάθε εντολή της γλώσσας μηχανής => εκτέλεση ενός διαφορετικού μικροπρογράμματος.
- Γλώσσα μηχανής : κάθε διαφορετική εντολή = διαφορετική ακολουθία από 0 & 1.
- Assembly : συμβολικά ονόματα για κάθε εντολή. Μεταφράζονται σε 0 και 1 από τον assembler.
- Στο μεταίχμιο του πλήρους ελέγχου του υλικού και του τι καταλαβαίνει ο άνθρωπος.
- Ιδιο ρεπερτόριο εντολών => συμβατότητα.
- Μάθημα: Βασικές αρχές οργάνωσης & λειτουργίας υπολογιστικών συστημάτων.

Επίπεδο υψηλών γλωσσών προγραμματισμού

- Η γλώσσα που χρησιμοποιείται είναι πολύ πιο κοντά στον άνθρωπο από ότι στη μηχανή.
- Πιο ευρέως χρησιμοποιούμενες γλώσσες: Python, Java, C/C++, Go, Ruby, HTML, ...
- Μεταφραστής (compiler): source code $\rightarrow$ object code.
- Διερμηνευτής (interpreter): μετάφραση μιας γραμμής πηγαίου κώδικα, εκτέλεσή της και πάλι από την αρχή.
- Αποσφαλμάτωση (debugging).
- Μαθήματα : Σχεδόν όλα του Τομέα Λογικού των Υπολογιστών.

Επίπεδο χρήστη / εφαρμογών

- Συναρτήσεις : έτοιμα προγράμματα για επεξεργασία κειμένου, λογιστικά φύλλα, επεξεργασία εικόνας / ήχου, παιχνίδια, ...
- Το επίπεδο ενός καθημερινού χρήστη υπολογιστικού συστήματος.

Συμβατότητα

- Είναι επιθυμητό οι υπολογιστές επόμενης γενιάς να μπορούν να εκτελέσουν προγράμματα που αναπτύχθηκαν για συστήματα περασμένων ετών.
- Ακόμα κι αν τα πιο κάτω επίπεδα δύο υπολογιστικών συστημάτων υλοποιούνται με διαφορετικό τρόπο, μπορεί οι συναρτήσεις που προσφέρουν προς τα πιο πάνω να είναι πανομοιότυπες ή το ένα σύνολο να είναι γνήσιο υποσύνολο του άλλου.
- Σε αυτή την περίπτωση θα λέμε ότι αυτοί οι υπολογιστές είναι συμβατοί ως προς το πιο πάνω επίπεδο.
- Αυτός είναι ακόμη ένας λόγος της ιεραρχικής δόμησης των υπολογιστικών συστημάτων.
- Παραδείγματα: Intel x86, Power PC, HP PA 2.0

Αναπαράσταση Πληροφορίας - Ορισμοί

- Πριν την επεξεργασία της πληροφορίας, θα πρέπει να βρούμε κάποιο τρόπο έκφρασής της σε μορφή κατανοητή από τον υπολογιστή.

Αναπαράσταση Πληροφορίας - Ορισμοί

- Πριν την επεξεργασία της πληροφορίας, θα πρέπει να βρούμε κάποιο τρόπο έκφρασής της σε μορφή κατανοητή από τον υπολογιστή.
- Αναπαράσταση - representation ή κωδικοποίηση - encoding της πληροφορίας ονομάζεται η έκφραση της πληροφορίας σε μορφή κατανοητή από τον υπολογιστή.

Αναπαράσταση Πληροφορίας - Ορισμοί

- Πριν την επεξεργασία της πληροφορίας, θα πρέπει να βρούμε κάποιο τρόπο έκφρασής της σε μορφή κατανοητή από τον υπολογιστή.
- Αναπαράσταση - representation ή κωδικοποίηση - encoding της πληροφορίας ονομάζεται η έκφραση της πληροφορίας σε μορφή κατανοητή από τον υπολογιστή.
- Ένα σύνολο κανόνων βάσει των οποίων μπορούμε συστηματικά να φτιάξουμε τις αναπαραστάσεις διαφόρων πληροφοριών ονομάζεται κώδικας.

Αναπαράσταση Πληροφορίας - Ορισμοί

- Πριν την επεξεργασία της πληροφορίας, θα πρέπει να βρούμε κάποιο τρόπο έκφρασής της σε μορφή κατανοητή από τον υπολογιστή.
- Αναπαράσταση - representation ή κωδικοποίηση - encoding της πληροφορίας ονομάζεται η έκφραση της πληροφορίας σε μορφή κατανοητή από τον υπολογιστή.
- Ένα σύνολο κανόνων βάσει των οποίων μπορούμε συστηματικά να φτιάξουμε τις αναπαραστάσεις διαφόρων πληροφοριών ονομάζεται κώδικας.
- Έχουν αναπτυχθεί διαφορετικοί κώδικες για την ίδια πληροφορία. Σκοπός είναι οι αναπαραστάσεις να διέπονται από συγκεκριμένες ιδιότητες που επιτρέπουν την εύκολη ανάπτυξη μεθοδολογιών επεξεργασίας.
  Π.χ. οι αριθμητικοί κώδικες επιτρέπουν γρήγορες αριθμητικές πράξεις, οι κώδικες ανίχνευσης και διόρθωσης λαθών τον γρήγορο και αποτελεσματικό έλεγχο ορθότητας της πληροφορίας, ...

Δυαδικό ψηφίο

- Το στοιχειώδες υλικό ενός υπολογιστή μπορεί να λειτουργήσει αξιόπιστα έχοντας σα βάση 2 διακριτές στάθμες δυναμικού. Η κατάσταση του στοιχειώδους κομματιού υλικού είναι και η ελάχιστη αναπαραστίσιμη πληροφορία.

Δυαδικό ψηφίο

- Το στοιχειώδες υλικό ενός υπολογιστή μπορεί να λειτουργήσει αξιόπιστα έχοντας σα βάση 2 διακριτές στάθμες δυναμικού. Η κατάσταση του στοιχειώδους κομματιού υλικού είναι και η ελάχιστη αναπαραστίσιμη πληροφορία.
- Αν θεωρήσουμε μία εκ των δύο $1 \to 1$ και επί συναρτήσεων του $\{0, 1\}$ με τις δύο αυτές στάθμες δυναμικού μπορούμε να πούμε ότι η στοιχειώδης πληροφορία μας αναπαρίσταται σαν 0 ή 1 στον υπολογιστή.

Δυαδικό ψηφίο

- Το στοιχειώδες υλικό ενός υπολογιστή μπορεί να λειτουργήσει αξιόπιστα έχοντας σα βάση 2 διακριτές στάθμες δυναμικού. Η κατάσταση του στοιχειώδους κομματιού υλικού είναι και η ελάχιστη αναπαραστίσιμη πληροφορία.
- Αν θεωρήσουμε μία εκ των δύο $1 \to 1$ και επί συναρτήσεων του $\{0,\ 1\}$ με τις δύο αυτές στάθμες δυναμικού μπορούμε να πούμε ότι η στοιχειώδης πληροφορία μας αναπαρίσταται σαν 0 ή 1 στον υπολογιστή.
- Π.χ έστω ότι υιοθετούμε τη $\{0, 1\} \to \{0V, 5V\}$. Η πληροφορία για το αν είναι αναμμένο ή σβηστό ένα φως μπορεί συνεπώς να αντιστοιχιστεί σαν {σβηστό, αναμμένο}$\to \{0, 1\}$, ενώ στον υπολογιστή στην ουσία θα παρασταθεί σαν {σβηστό, αναμμένο}$\to \{0V, 5V\}$.

Δυαδικό ψηφίο

- Το στοιχειώδες υλικό ενός υπολογιστή μπορεί να λειτουργήσει αξιόπιστα έχοντας σα βάση 2 διακριτές στάθμες δυναμικού. Η κατάσταση του στοιχειώδους κομματιού υλικού είναι και η ελάχιστη αναπαραστίσιμη πληροφορία.
- Αν θεωρήσουμε μία εκ των δύο $1 \to 1$ και επί συναρτήσεων του $\{0,\ 1\}$ με τις δύο αυτές στάθμες δυναμικού μπορούμε να πούμε ότι η στοιχειώδης πληροφορία μας αναπαρίσταται σαν 0 ή 1 στον υπολογιστή.
- Π.χ έστω ότι υιοθετούμε τη $\{0, 1\} \to \{0V, 5V\}$. Η πληροφορία για το αν είναι αναμμένο ή σβηστό ένα φως μπορεί συνεπώς να αντιστοιχιστεί σαν {σβηστό, αναμμένο}$\to \{0, 1\}$, ενώ στον υπολογιστή στην ουσία θα παρασταθεί σαν {σβηστό, αναμμένο}$\to \{0V, 5V\}$.
- Άρα η στοιχειώδης πληροφορία που μπορεί να παρασταθεί στον υπολογιστή ισοδυναμεί με τη κατάσταση ενός δυαδικού ψηφίου (BInary digiT - bit), ψηφίου που μπορεί να πάρει τις τιμές 0 ή 1.
- δυαδικό ψηφίο $\equiv$ δυαδικό σήμα $\equiv$ δυαδική μεταβλητή

Ενα ψηφίο δεν είναι ποτέ αρκετό !

- Αν ήθελα να παραστήσω κατασκευαστές κινητών τηλεφώνων τότε θα μπορούσα με ένα δεκαδικό ψηφίο να κάνω την αντιστοίχιση : $0 \leftrightarrow Nokia, 3 \leftrightarrow Siemens, 5 \leftrightarrow LG, 6 \leftrightarrow Sony, 7 \leftrightarrow Alcatel.$

Ενα ψηφίο δεν είναι ποτέ αρκετό !

- Αν ήθελα να παραστήσω κατασκευαστές κινητών τηλεφώνων τότε θα μπορούσα με ένα δεκαδικό ψηφίο να κάνω την αντιστοίχιση : $0 \leftrightarrow Nokia, 3 \leftrightarrow Siemens, 5 \leftrightarrow LG, 6 \leftrightarrow Sony, 7 \leftrightarrow Alcatel.$
- Οι δύο καταστάσεις του ενός δυαδικού ψηφίου δε φτάνουν για να παραστήσω όλους τους παραπάνω. Ομοια 1 δεκαδικό ψηφίο δε θα έφτανε για 12 διαφορετικούς κατασκευαστές.

Ενα ψηφίο δεν είναι ποτέ αρκετό !

- Αν ήθελα να παραστήσω κατασκευαστές κινητών τηλεφώνων τότε θα μπορούσα με ένα δεκαδικό ψηφίο να κάνω την αντιστοίχιση : $0 \leftrightarrow Nokia, 3 \leftrightarrow Siemens, 5 \leftrightarrow LG, 6 \leftrightarrow Sony, 7 \leftrightarrow Alcatel.$
- Οι δύο καταστάσεις του ενός δυαδικού ψηφίου δε φτάνουν για να παραστήσω όλους τους παραπάνω. Ομοια 1 δεκαδικό ψηφίο δε θα έφτανε για 12 διαφορετικούς κατασκευαστές.
- Για τη κωδικοποίηση k διαφορετικών καταστάσεων, θα χρειαστώ b δυαδικά ψηφία, όπου $b \geq \log_2 k$.

Ενα ψηφίο δεν είναι ποτέ αρκετό !

- Αν ήθελα να παραστήσω κατασκευαστές κινητών τηλεφώνων τότε θα μπορούσα με ένα δεκαδικό ψηφίο να κάνω την αντιστοίχιση : $0 \leftrightarrow Nokia, 3 \leftrightarrow Siemens, 5 \leftrightarrow LG, 6 \leftrightarrow Sony, 7 \leftrightarrow Alcatel.$
- Οι δύο καταστάσεις του ενός δυαδικού ψηφίου δε φτάνουν για να παραστήσω όλους τους παραπάνω. Ομοια 1 δεκαδικό ψηφίο δε θα έφτανε για 12 διαφορετικούς κατασκευαστές.
- Για τη κωδικοποίηση k διαφορετικών καταστάσεων, θα χρειαστώ b δυαδικά ψηφία, όπου $b \geq \log_2 k$.
- Π.χ. $101 \leftrightarrow Nokia, 111 \leftrightarrow Siemens, 000 \leftrightarrow LG, 011 \leftrightarrow Sony, 100 \leftrightarrow Alcatel.$

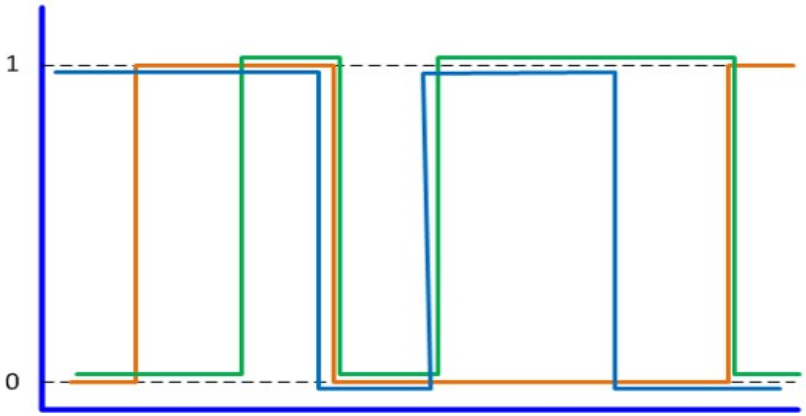
Ενα ψηφίο δεν είναι ποτέ αρκετό !

- Αν ήθελα να παραστήσω κατασκευαστές κινητών τηλεφώνων τότε θα μπορούσα με ένα δεκαδικό ψηφίο να κάνω την αντιστοίχιση : $0 \leftrightarrow Nokia, 3 \leftrightarrow Siemens, 5 \leftrightarrow LG, 6 \leftrightarrow Sony, 7 \leftrightarrow Alcatel.$
- Οι δύο καταστάσεις του ενός δυαδικού ψηφίου δε φτάνουν για να παραστήσω όλους τους παραπάνω. Ομοια 1 δεκαδικό ψηφίο δε θα έφτανε για 12 διαφορετικούς κατασκευαστές.
- Για τη κωδικοποίηση k διαφορετικών καταστάσεων, θα χρειαστώ b δυαδικά ψηφία, όπου $b \geq \log_2 k$.
- Π.χ. $101 \leftrightarrow Nokia, 111 \leftrightarrow Siemens, 000 \leftrightarrow LG, 011 \leftrightarrow Sony, 100 \leftrightarrow Alcatel.$
- Η συνάρτηση απεικόνισης είναι μέρος των κανόνων του κώδικα.

Ενα ψηφίο δεν είναι ποτέ αρκετό !

- Αν ήθελα να παραστήσω κατασκευαστές κινητών τηλεφώνων τότε θα μπορούσα με ένα δεκαδικό ψηφίο να κάνω την αντιστοίχιση : $0 \leftrightarrow Nokia, 3 \leftrightarrow Siemens, 5 \leftrightarrow LG, 6 \leftrightarrow Sony, 7 \leftrightarrow Alcatel.$
- Οι δύο καταστάσεις του ενός δυαδικού ψηφίου δε φτάνουν για να παραστήσω όλους τους παραπάνω. Ομοια 1 δεκαδικό ψηφίο δε θα έφτανε για 12 διαφορετικούς κατασκευαστές.
- Για τη κωδικοποίηση k διαφορετικών καταστάσεων, θα χρειαστώ b δυαδικά ψηφία, όπου $b \geq \log_2 k$.
- Π.χ. $101 \leftrightarrow Nokia, 111 \leftrightarrow Siemens, 000 \leftrightarrow LG, 011 \leftrightarrow Sony, 100 \leftrightarrow Alcatel.$
- Η συνάρτηση απεικόνισης είναι μέρος των κανόνων του κώδικα.
- Περισσότερα δυαδικά ψηφία $\Rightarrow$ Περισσότερα στοιχειώδη κομμάτια υλικού $\Rightarrow \uparrow$ κόστος

Αναπαράσταση σήματος 8 σταθμών με δυαδικά σήματα



8 στάθμες => 3 δυαδικά σήματα
(x, y, z) για τη κωδικοποίηση.
Έστω η κωδικοποίηση :

0 -> 000
1 -> 001
2 -> 010
3 -> 011
4 -> 100
5 -> 101
6 -> 110
7 -> 111

Πολλαπλάσια

- Για πρακτικά προβλήματα χρειαζόμαστε ποσότητες πολλαπλάσιες του ενός δυαδικού ψηφίου.
- Ευρέως χρησιμοποιούμενα πολλαπλάσια :
  - Τετράδα (nibble) = τέσσερα (2^2) δυαδικά ψηφία
  - Οκτάδα (Byte) = οκτώ (2^3) δυαδικά ψηφία
  - Kilobyte - KByte (KB) = 1024 (2^{10}) Bytes
  - Megabyte - MByte (MB) = 1024 KBytes = 2^{20} Bytes
  - Gigabyte - GByte (GB) = 1024 MBytes = 2^{30} Bytes
  - Terabyte - TByte (TB) = 1024 GByte = 2^{40} Bytes
  - 1GB = 2^{33} δυαδικά ψηφία.

Ψηφία Αριθμητικού Συστήματος

- r-δικό σύστημα : καθε ψηφίο του παίρνει τις φυσικές τιμές του $[0, r-1]$.

Ψηφία Αριθμητικού Συστήματος

- r-δικό σύστημα : καθε ψηφίο του παίρνει τις φυσικές τιμές του $[0, r-1]$.
- Στο γνωστό δεκαδικό μας σύστημα κάθε ψηφίο παίρνει τους φυσικούς του $[0, 9]$.
- Στο δυαδικό κάθε ψηφίο είναι 0 ή 1.
- Στο 8-δικό το ψηφίο μπορεί να είναι 0,1,2,3,4,5,6 ή 7.

Ψηφία Αριθμητικού Συστήματος

- r-δικό σύστημα : καθε ψηφίο του παίρνει τις φυσικές τιμές του $[0, r-1]$.
- Στο γνωστό δεκαδικό μας σύστημα κάθε ψηφίο παίρνει τους φυσικούς του $[0, 9]$.
- Στο δυαδικό κάθε ψηφίο είναι 0 ή 1.
- Στο 8-δικό το ψηφίο μπορεί να είναι 0,1,2,3,4,5,6 ή 7.
- Οταν $r > 10$, χρησιμοποιούνται τα λατινικά γράμματα για τις τιμές που είναι 10 και πάνω.

Ψηφία Αριθμητικού Συστήματος

- r-δικό σύστημα : καθε ψηφίο του παίρνει τις φυσικές τιμές του $[0, r-1]$.
- Στο γνωστό δεκαδικό μας σύστημα κάθε ψηφίο παίρνει τους φυσικούς του $[0, 9]$.
- Στο δυαδικό κάθε ψηφίο είναι 0 ή 1.
- Στο 8-δικό το ψηφίο μπορεί να είναι 0,1,2,3,4,5,6 ή 7.
- Οταν $r > 10$, χρησιμοποιούνται τα λατινικά γράμματα για τις τιμές που είναι 10 και πάνω.
- Στο 16-δικό, κάθε ψηφίο μπορεί να πάρει τις τιμές 0,1,...,9, A(=10), B(=11), C(=12), D(=13), E(=14) ή F(=15).

Κώδικες με Βάρη (1/2)

- Κώδικας με βάρη ή θεσιακό σύστημα είναι κάθε κώδικας στον οποίο ένα ψηφίο έχει μια βαρύτητα η οποία εξαρτάται από τη θέση την οποία καταλαμβάνει.

Κώδικες με Βάρη (1/2)

- Κώδικας με βάρη ή θεσιακό σύστημα είναι κάθε κώδικας στον οποίο ένα ψηφίο έχει μια βαρύτητα η οποία εξαρτάται από τη θέση την οποία καταλαμβάνει.
- Έστω η λέξη $d_{n-1}d_{n-2}\ldots d_1d_0$, με d_i ψηφίο του r-δικού συστήματος, σε έναν κώδικα με βάρη $w_{n-1}w_{n-2}\ldots w_1w_0$. Δηλαδή η θέση του d_{n-1} έχει βάρος w_{n-1}, η θέση του d_{n-2} έχει βάρος $w_{n-2}, \ldots$.

Κώδικες με Βάρη (1/2)

- Κώδικας με βάρη ή θεσιακό σύστημα είναι κάθε κώδικας στον οποίο ένα ψηφίο έχει μια βαρύτητα η οποία εξαρτάται από τη θέση την οποία καταλαμβάνει.
- Έστω η λέξη $d_{n-1}d_{n-2}\ldots d_1d_0$, με d_i ψηφίο του r-δικού συστήματος, σε έναν κώδικα με βάρη $w_{n-1}w_{n-2}\ldots w_1w_0$. Δηλαδή η θέση του d_{n-1} έχει βάρος w_{n-1}, η θέση του d_{n-2} έχει βάρος $w_{n-2}, \ldots$.
- Η λέξη αυτή αντιπροσωπεύει τη ποσότητα $\sum_{i=0}^{i=n-1}(w_id_i)$

Κώδικες με Βάρη (2/2)

- Π.χ. Επταδικά ψηφία, αναπαράσταση 2 3 6 1, βάρη 6-4-3-1

Κώδικες με Βάρη (2/2)

- Π.χ. Επταδικά ψηφία, αναπαράσταση 2 3 6 1, βάρη 6-4-3-1
- Αυτή είναι η αναπαράσταση του
  $2361_7 = 6 \times 2 + 4 \times 3 + 3 \times 6 + 1 \times 1 = 43.$

Κώδικες με Βάρη (2/2)

- Π.χ. Επταδικά ψηφία, αναπαράσταση 2 3 6 1, βάρη 6-4-3-1
- Αυτή είναι η αναπαράσταση του
  $2361_7 = 6 \times 2 + 4 \times 3 + 3 \times 6 + 1 \times 1 = 43.$
- Πρόβλημα : Η ίδια ποσότητα μπορεί να έχει πολλές αναπαραστάσεις.
- Π.χ. $6110_7 = 6 \times 6 + 4 \times 1 + 3 \times 1 + 1 \times 0 = 43.$

Κώδικες με Βάρη (2/2)

- Π.χ. Επταδικά ψηφία, αναπαράσταση 2 3 6 1, βάρη 6-4-3-1
- Αυτή είναι η αναπαράσταση του
  $2361_7 = 6 \times 2 + 4 \times 3 + 3 \times 6 + 1 \times 1 = 43.$
- Πρόβλημα : Η ίδια ποσότητα μπορεί να έχει πολλές αναπαραστάσεις.
- Π.χ. $6110_7 = 6 \times 6 + 4 \times 1 + 3 \times 1 + 1 \times 0 = 43.$
- Λύση: Αριθμητικό σύστημα, δηλαδή $w_i = r^i$.

Κώδικες με Βάρη (2/2)

- Π.χ. Επταδικά ψηφία, αναπαράσταση 2 3 6 1, βάρη 6-4-3-1
- Αυτή είναι η αναπαράσταση του
  $2361_7 = 6 \times 2 + 4 \times 3 + 3 \times 6 + 1 \times 1 = 43.$
- Πρόβλημα : Η ίδια ποσότητα μπορεί να έχει πολλές αναπαραστάσεις.
- Π.χ. $6110_7 = 6 \times 6 + 4 \times 1 + 3 \times 1 + 1 \times 0 = 43.$
- Λύση: Αριθμητικό σύστημα, δηλαδή $w_i = r^i$.
- Η αναπαράσταση μιας ποσότητας στο r-δικό σύστημα θα την γράφουμε ως $d_{n-1}d_{n-2}\ldots d_1 d_0\,{}_r$.

Παραδείγματα Αναπαραστάσεων & Μετρατροπές

- $827 = 827_{10} = 8 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$

Παραδείγματα Αναπαραστάσεων & Μετατροπές

- $827 = 827_{10} = 8 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$
- $1010011_2 = 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0 = 64 + 16 + 2 + 1 = 83_{10}$

Παραδείγματα Αναπαραστάσεων & Μετρατροπές

- $827 = 827_{10} = 8 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$
- $1010011_2 = 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0 = 64 + 16 + 2 + 1 = 83_{10}$
- $AD2_{16} = A(= 10) \times 16^2 + D(= 13) \times 16^1 + 2 \times 16^0 =$
  $2560 + 208 + 1 = 2769_{10}$

Παραδείγματα Αναπαραστάσεων & Μετρατροπές

- $827 = 827_{10} = 8 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$
- $1010011_2 = 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0 = 64 + 16 + 2 + 1 = 83_{10}$
- $AD2_{16} = A(= 10) \times 16^2 + D(= 13) \times 16^1 + 2 \times 16^0 = 2560 + 208 + 1 = 2769_{10}$
- Η αναπαριστούμενη ποσότητα προκύπτει από την εφαρμογή του κανόνα του πολυωνύμου

Παραδείγματα Αναπαραστάσεων & Μετρατροπές

- $827 = 827_{10} = 8 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$
- $1010011_2 = 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0 = 64 + 16 + 2 + 1 = 83_{10}$
- $AD2_{16} = A(= 10) \times 16^2 + D(= 13) \times 16^1 + 2 \times 16^0 = 2560 + 208 + 1 = 2769_{10}$
- Η αναπαριστούμενη ποσότητα προκύπτει από την εφαρμογή του κανόνα του πολυωνύμου
- Ταυτόχρονα αυτή είναι και μια διαδικασία μετατροπής του $K_r \leftrightarrow M_{10}$

Παραδείγματα Αναπαραστάσεων & Μετρατροπές

- $827 = 827_{10} = 8 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$
- $1010011_2 = 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0 = 64 + 16 + 2 + 1 = 83_{10}$
- $AD2_{16} = A(= 10) \times 16^2 + D(= 13) \times 16^1 + 2 \times 16^0 = 2560 + 208 + 1 = 2769_{10}$
- Η αναπαριστούμενη ποσότητα προκύπτει από την εφαρμογή του κανόνα του πολυωνύμου
- Ταυτόχρονα αυτή είναι και μια διαδικασία μετατροπής του $K_r \leftrightarrow M_{10}$
- Πως μπορώ να κάνω το $K_r \leftrightarrow N_s$?
- Διαδικασία 2 βημάτων $K_r \leftrightarrow M_{10} \leftrightarrow N_s$.

Από το 10δικό σε άλλη βάση

- Εστω M_{10}. Στόχος $N_s = n_{k-1}n_{k-2}...n_2n_1n_0\ _s = M_{10}$

Από το 10δικό σε άλλη βάση

- Εστω M_{10}. Στόχος $N_s = n_{k-1}n_{k-2}...n_2n_1n_0{}_s = M_{10}$
- $n_{k-1} \times s^{k-1} + n_{k-2} \times s^{k-2} + ... + n_1 \times s^1 + n_0 \times s^0 = M_{10} \Rightarrow$
  $\frac{M_{10}}{s} = [n_{k-1} \times s^{k-2} + n_{k-2} \times s^{k-3} + ... + n_1 \times s^0] + \frac{n_0}{s}$

Από το 10δικό σε άλλη βάση

- Εστω M_{10}. Στόχος $N_s = n_{k-1}n_{k-2}...n_2n_1n_0\ _s = M_{10}$
- $n_{k-1} \times s^{k-1} + n_{k-2} \times s^{k-2} + ... + n_1 \times s^1 + n_0 \times s^0 = M_{10} \Rightarrow$
  $\frac{M_{10}}{s} = [n_{k-1} \times s^{k-2} + n_{k-2} \times s^{k-3} + ... + n_1 \times s^0] + \frac{n_0}{s}$
- Η διαίρεση της δεκαδικής αναπαράστασης με τη στοχευόμενη βάση, μας έδωσε το ψηφίο ελάχιστης σημαντικότητας σαν υπόλοιπο! Ταυτόχρονα μετέτρεψε ένα πρόβλημα καθορισμού k αγνώστων, σε ένα καθορισμού $k-1$ αγνώστων.

Από το 10δικό σε άλλη βάση

- Εστω M_{10}. Στόχος $N_s = n_{k-1}n_{k-2}...n_2n_1n_0{}_s = M_{10}$
- $n_{k-1} \times s^{k-1} + n_{k-2} \times s^{k-2} + ... + n_1 \times s^1 + n_0 \times s^0 = M_{10} \Rightarrow$
  $\frac{M_{10}}{s} = [n_{k-1} \times s^{k-2} + n_{k-2} \times s^{k-3} + ... + n_1 \times s^0] + \frac{n_0}{s}$
- Η διαίρεση της δεκαδικής αναπαράστασης με τη στοχευόμενη βάση, μας έδωσε το ψηφίο ελάχιστης σημαντικότητας σαν υπόλοιπο! Ταυτόχρονα μετέτρεψε ένα πρόβλημα καθορισμού k αγνώστων, σε ένα καθορισμού $k-1$ αγνώστων.
- Συνεπώς μπορώ να συνεχίσω να εκτελώ αυτή τη διαδικασία επαναληπτικά μέχρι να καθορίσω έναν έναν όλους τους k αγνώστους.

Από το 10δικό σε άλλη βάση

- Εστω M_{10}. Στόχος $N_s = n_{k-1}n_{k-2}...n_2n_1n_0{}_s = M_{10}$
- $n_{k-1} \times s^{k-1} + n_{k-2} \times s^{k-2} + ... + n_1 \times s^1 + n_0 \times s^0 = M_{10} \Rightarrow$
  $\frac{M_{10}}{s} = [n_{k-1} \times s^{k-2} + n_{k-2} \times s^{k-3} + ... + n_1 \times s^0] + \frac{n_0}{s}$
- Η διαίρεση της δεκαδικής αναπαράστασης με τη στοχευόμενη βάση, μας έδωσε το ψηφίο ελάχιστης σημαντικότητας σαν υπόλοιπο! Ταυτόχρονα μετέτρεψε ένα πρόβλημα καθορισμού k αγνώστων, σε ένα καθορισμού $k-1$ αγνώστων.
- Συνεπώς μπορώ να συνεχίσω να εκτελώ αυτή τη διαδικασία επαναληπτικά μέχρι να καθορίσω έναν έναν όλους τους k αγνώστους.
- Σε κάθε επανάληψη θα μου προκύπτει το ψηφίο αμέσως μεγαλύτερης σημαντικότητας. Θα σταματήσω όταν το πηλίκο μηδενιστεί.

Από το 10δικό σε άλλη βάση : Παράδειγμα

- $614_{10} = ?_6$

Από το 10δικό σε άλλη βάση : Παράδειγμα

- $614_{10} = ?_6$
- $614 = 102 \times 6 + 2 \rightarrow LSD$

Από το 10δικό σε άλλη βάση : Παράδειγμα

- $614_{10} = ?_6$
- $614 = 102 \times 6 + 2 \rightarrow LSD$
- $102 = 17 \times 6 + 0$

Από το 10δικό σε άλλη βάση : Παράδειγμα

- $614_{10} = ?_6$
- $614 = 102 \times 6 + 2 \rightarrow LSD$
- $102 = 17 \times 6 + 0$
- $17 = 2 \times 6 + 5$

Από το 10δικό σε άλλη βάση : Παράδειγμα

- $614_{10} = ?_6$
- $614 = 102 \times 6 + 2 \rightarrow LSD$
- $102 = 17 \times 6 + 0$
- $17 = 2 \times 6 + 5$
- $2 = 0 \times 6 + 2 \rightarrow MSD$

Από το 10δικό σε άλλη βάση : Παράδειγμα

- $614_{10} = ?_6$
- $614 = 102 \times 6 + 2 \rightarrow LSD$
- $102 = 17 \times 6 + 0$
- $17 = 2 \times 6 + 5$
- $2 = 0 \times 6 + 2 \rightarrow MSD$
- Άρα $614_{10} = 2502_6$.

Από το 10δικό σε άλλη βάση : Παράδειγμα

- $614_{10} = ?_6$
- $614 = 102 \times 6 + 2 \rightarrow LSD$
- $102 = 17 \times 6 + 0$
- $17 = 2 \times 6 + 5$
- $2 = 0 \times 6 + 2 \rightarrow MSD$
- Άρα $614_{10} = 2502_6$.
- Πράγματι $2502_6 = 2 \times 6^3 + 5 \times 6^2 + 2 \times 6^0 = 432 + 180 + 2 = 614_{10}$

Παρακάμπτοντας τους μεσάζοντες

Όταν θέλουμε $K_r \leftrightarrow N_s$ και είναι $r = s^a$, αντικαθιστώ 1 ψηφίο βάσης r με a ψηφία βάσης s

16δικό ψηφίο	0	1	2	3	4	5	6	7
2δικά ψηφία	0000	0001	0010	0011	0100	0101	0110	0111
16δικό ψηφίο	8	9	A	B	C	D	E	F
2δικά ψηφία	1000	1001	1010	1011	1100	1101	1110	1111

8δικό ψηφίο	0	1	2	3	4	5	6	7
2δικά ψηφία	000	001	010	011	100	101	110	111

Από 16δικό σε 8δικό με 2δικό ενδιάμεσα

- $E43B_{16} = ?_8$

Από 16δικό σε 8δικό με 2δικό ενδιάμεσα

- $E43B_{16} = ?_8$
- $E43B_{16} = 1110\ 0100\ 0011\ 1011_2$

Από 16δικό σε 8δικό με 2δικό ενδιάμεσα

- $E43B_{16} = ?_8$
- $E43B_{16} = 1110\ 0100\ 0011\ 1011_2$
- $1110010000111011_2 = 001\ 110\ 010\ 000\ 111\ 011_2 = 162073_8$

Από 16δικό σε 8δικό με 2δικό ενδιάμεσα

- $E43B_{16} = ?_8$
- $E43B_{16} = 1110\ 0100\ 0011\ 1011_2$
- $1110010000111011_2 = 001\ 110\ 010\ 000\ 111\ 011_2 = 162073_8$
- Πράγματι
  $\mathrm{E43B}_{16} = 14 \times 16^3 + 4 \times 16^2 + 3 \times 16^1 + 11 \times 16^0 = 58427_{10}$ και
  $162073_8 = 1 \times 8^5 + 6 \times 8^4 + 2 \times 8^3 + 7 \times 8^1 + 3 \times 8^0 = 58427_{10}$

Περιορισμοί αναπαραστάσεων - Υπερχείλιση

- Εύρος αναπαράστασης. Στη καθημερινή μας ζωή δε σκεφτόμαστε τον αριθμό των ψηφίων που είναι διαθέσιμα. Κάθε υπολογιστικό σύστημα έχει μια τιμή ⇒ πεπερασμένο υλικό ⇒ μπορεί να αποθηκεύσει πεπερασμένο μήκος πληροφορίας ⇒ σε κάθε ακέραιο αφιερώνεται συγκεκριμένος αριθμός δυαδικών ψηφίων.

Περιορισμοί αναπαραστάσεων - Υπερχείλιση

- Εύρος αναπαράστασης. Στη καθημερινή μας ζωή δε σκεφτόμαστε τον αριθμό των ψηφίων που είναι διαθέσιμα. Κάθε υπολογιστικό σύστημα έχει μια τιμή $\Rightarrow$ πεπερασμένο υλικό $\Rightarrow$ μπορεί να αποθηκεύσει πεπερασμένο μήκος πληροφορίας $\Rightarrow$ σε κάθε ακέραιο αφιερώνεται συγκεκριμένος αριθμός δυαδικών ψηφίων.
- Διάφοροι νόμοι δεν ισχύουν. Π.χ. με ένα δεκαδικό ψηφίο $7 + (8 - 6) \neq (7 + 8) - 6$.

Περιορισμοί αναπαραστάσεων - Υπερχείλιση

- Εύρος αναπαράστασης. Στη καθημερινή μας ζωή δε σκεφτόμαστε τον αριθμό των ψηφίων που είναι διαθέσιμα. Κάθε υπολογιστικό σύστημα έχει μια τιμή $\Rightarrow$ πεπερασμένο υλικό $\Rightarrow$ μπορεί να αποθηκεύσει πεπερασμένο μήκος πληροφορίας $\Rightarrow$ σε κάθε ακέραιο αφιερώνεται συγκεκριμένος αριθμός δυαδικών ψηφίων.
- Διάφοροι νόμοι δεν ισχύουν. Π.χ. με ένα δεκαδικό ψηφίο $7 + (8 - 6) \neq (7 + 8) - 6$.
- Ενα ενδιάμεσο αποτέλεσμα κάποιες φορές δε μπορεί να αναπαρασταθεί στο δεδομένο εύρος αναπαραστάσεων $\Rightarrow$ υπερχείλιση.

Αναπαραστάσεις Ακεραίων

- Οι υπολογισμοί πάνω σε φυσικούς αριθμούς είναι ένα εξαιρετικά μικρό υποσύνολο των υπολογισμών που πρέπει να κάνουμε.

Αναπαραστάσεις Ακεραίων

- Οι υπολογισμοί πάνω σε φυσικούς αριθμούς είναι ένα εξαιρετικά μικρό υποσύνολο των υπολογισμών που πρέπει να κάνουμε.
- Αρα χρειαζόμαστε αναπαραστάσεις ακεραίων και πραγματικών αριθμών.

Αναπαραστάσεις Ακεραίων

- Οι υπολογισμοί πάνω σε φυσικούς αριθμούς είναι ένα εξαιρετικά μικρό υποσύνολο των υπολογισμών που πρέπει να κάνουμε.
- Αρα χρειαζόμαστε αναπαραστάσεις ακεραίων και πραγματικών αριθμών.
- Υπάρχουν 4 κώδικες αναπαράστασης ακεραίων: Πρόσημο - μέτρο, συμπλήρωμα ως προς 1, συμπλήρωμα ως προς 2 και κώδικας πόλωσης.

Κώδικας Πρόσημου - Μέτρου

- Το πλέον αριστερότερο δυαδικό ψηφίο χρησιμοποιείται για το πρόσημο του αριθμού. $1 \leftrightarrow$ αρνητικός, $0 \leftrightarrow$ θετικός. Τα υπόλοιπα ψηφία υποδεικνύουν το μέτρο του αριθμού.
- Αρα η $\beta_{\nu-1}\beta_{\nu-2}\ldots\beta_1\beta_0$ αντιπροσωπεύει τον ακέραιο : $(-1)^{\beta_{\nu-1}}\sum_{i=0}^{\nu-2}(2^i\beta_i)$.
- Με εύρος κ δυαδικών ψηφίων αναπαράστασης, αναπαριστώ τους ακεραίους στο διάστημα $[-(2^{\kappa-1}-1), 2^{\kappa-1}-1]$.
- Ο μεγαλύτερος αριθμός είναι ο $011\ldots11$ και ο μικρότερος ο $111\ldots11$.
- Δυστυχώς παρότι απόλυτα κατανοητός από τον άνθρωπο, ο κώδικας αυτός έχει 2 αναπαραστάσεις για το 0 :
  την $+0 : 000\ldots00$ και
  την $-0 : 100\ldots00$.

Κώδικας Πόλωσης κατά Σ (Πλεονασμού κατά Σ)

- Η αναπαράσταση του Α είναι η δυαδική αναπαράσταση του $A + \Sigma$. Δηλαδή όλοι οι αριθμοί μετατίθενται κατά Σ.
- Έστω ότι ο μικρότερος αριθμός που θέλω να παραστήσω είναι ο $-K$. Διαλέγοντας $\Sigma = K$, ο μικρότερος αριθμός θα έχει παράσταση $-K + \Sigma(= K) = 0_{10} = 000 \ldots 00_{excess-K}$.
- Όταν το εύρος αναπαράστασης είναι κ δυαδικά ψηφία, συνήθως επιλέγουμε $\Sigma = 2^{\kappa-1}$ και παριστούμε όλους τους ακεραίους στο διάστημα $[-2^{\kappa-1}, 2^{\kappa-1} - 1]$.
- Η αποκωδικοποίηση της παράστασης σημαίνει την εφαρμογή του κανόνα του πολυωνύμου και την αφαίρεση του Σ.
- \+ : Καμμία σπατάλη αναπαραστάσεων, πολύ εύκολη σύγκριση αριθμών
  \- : Δύσκολες οι αριθμητικές πράξεις.

Συμπλήρωμα ως προς 1

- Κωδικοποίηση :
  - Βρίσκουμε μέτρο στο δυαδικό.
  - Αν (+) αυτή είναι και η παράστασή του.
  - Αν (-) αντιστρέφουμε όλα τα δυαδικά ψηφία.

Συμπλήρωμα ως προς 1

- Κωδικοποίηση :
  - Βρίσκουμε μέτρο στο δυαδικό.
  - Αν (+) αυτή είναι και η παράστασή του.
  - Αν (-) αντιστρέφουμε όλα τα δυαδικά ψηφία.
- Αποκωδικοποίηση, Μέθοδος 1:
  - Ελέγχουμε το αριστερότερο δυαδικό ψηφίο.
  - Αν 0, πρόσημο = +, μέτρο από κανόνα πολυωνύμου.
  - Αν 1, πρόσημο = -, μέτρο από κανόνα πολυωνύμου, αφού πρώτα αντιστρέψουμε όλα τα δυαδικά ψηφία.

Συμπλήρωμα ως προς 1

- Κωδικοποίηση :
  - Βρίσκουμε μέτρο στο δυαδικό.
  - Αν (+) αυτή είναι και η παράστασή του.
  - Αν (-) αντιστρέφουμε όλα τα δυαδικά ψηφία.
- Αποκωδικοποίηση, Μέθοδος 1:
  - Ελέγχουμε το αριστερότερο δυαδικό ψηφίο.
  - Αν 0, πρόσημο = +, μέτρο από κανόνα πολυωνύμου.
  - Αν 1, πρόσημο = -, μέτρο από κανόνα πολυωνύμου, αφού πρώτα αντιστρέψουμε όλα τα δυαδικά ψηφία.
- Αποκωδικοποίηση, Μέθοδος 2:
  - Εστω κ, ο αριθμός των ψηφίων της παράστασης.
  - Κανόνας πολυωνύμου με βάρος $-2^{\kappa-1}+1$ στο αριστερότερο ψηφίο.

Συμπλήρωμα ως προς 1

- Κωδικοποίηση :
  - Βρίσκουμε μέτρο στο δυαδικό.
  - Αν (+) αυτή είναι και η παράστασή του.
  - Αν (-) αντιστρέφουμε όλα τα δυαδικά ψηφία.
- Αποκωδικοποίηση, Μέθοδος 1:
  - Ελέγχουμε το αριστερότερο δυαδικό ψηφίο.
  - Αν 0, πρόσημο = +, μέτρο από κανόνα πολυωνύμου.
  - Αν 1, πρόσημο = -, μέτρο από κανόνα πολυωνύμου, αφού πρώτα αντιστρέψουμε όλα τα δυαδικά ψηφία.
- Αποκωδικοποίηση, Μέθοδος 2:
  - Εστω κ, ο αριθμός των ψηφίων της παράστασης.
  - Κανόνας πολυωνύμου με βάρος $-2^{\kappa-1}+1$ στο αριστερότερο ψηφίο.
- Με κ ψηφία αναπαρίστανται οι ακέραιοι στο $[-(2^{\kappa-1}-1), 2^{\kappa-1}-1]$.
- Ο μικρότερος είναι ο $100\ldots00$ και ο μεγαλύτερος ο $011\ldots11$.
- Δύο αναπαραστάσεις του 0 : $000\ldots00$ και $111\ldots11$.

Πρόσθεση στο δυαδικό : Μια πρώιμη ματιά

- Η πρόσθεση στο δυαδικό μπορεί να γίνει όπως στο δεκαδικό και πιο απλά.

Πρόσθεση στο δυαδικό : Μια πρώιμη ματιά

- Η πρόσθεση στο δυαδικό μπορεί να γίνει όπως στο δεκαδικό και πιο απλά.
- Ξεκινάμε πρώτα από τις μονάδες και υπολογίζουμε το ψηφίο μονάδων του αποτελέσματος. Αν κατά τη πρόσθεση προκύψει άθροισμα μεγαλύτερο από 2 (βάση), μεταφέρουμε κρατούμενο (ψηφίο μεγαλύτερης σημαντικότητας στην επόμενη βαθμίδα).
- Η πρόσθεση σε κάθε στήλη δύο δυαδικών αριθμών, ακολουθεί το πίνακα :

Είσοδοι			Εξοδοι	
Ψηφίο $1^{ου}$ Προσθετέου	Ψηφίο $2^{ου}$ Προσθετέου	Κρατούμενο από δεξιά	Ψηφίο αθροίσματος	Κρατούμενο προς δεξιά
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Συμπλήρωμα ως προς 2

- Κωδικοποίηση :
  - Βρίσκουμε μέτρο στο δυαδικό.
  - Αν (+) αυτή είναι και η παράστασή του.
  - Αν (-) αντιστρέφουμε όλα τα δυαδικά ψηφία και προσθέτουμε 1.

Συμπλήρωμα ως προς 2

- Κωδικοποίηση :
  - Βρίσκουμε μέτρο στο δυαδικό.
  - Αν (+) αυτή είναι και η παράστασή του.
  - Αν (-) αντιστρέφουμε όλα τα δυαδικά ψηφία και προσθέτουμε 1.
- Αποκωδικοποίηση, Μέθοδος 1:
  - Ελέγχουμε το αριστερότερο δυαδικό ψηφίο.
  - Αν 0, πρόσημο = +, μέτρο από κανόνα πολυωνύμου.
  - Αν 1, πρόσημο = -, αντιστρέφουμε όλα τα δυαδικά ψηφία, προσθέτουμε 1. Μέτρο = μέτρο της νέας παράστασης.

Συμπλήρωμα ως προς 2

- Κωδικοποίηση :
  - Βρίσκουμε μέτρο στο δυαδικό.
  - Αν (+) αυτή είναι και η παράστασή του.
  - Αν (-) αντιστρέφουμε όλα τα δυαδικά ψηφία και προσθέτουμε 1.
- Αποκωδικοποίηση, Μέθοδος 1:
  - Ελέγχουμε το αριστερότερο δυαδικό ψηφίο.
  - Αν 0, πρόσημο = +, μέτρο από κανόνα πολυωνύμου.
  - Αν 1, πρόσημο = -, αντιστρέφουμε όλα τα δυαδικά ψηφία, προσθέτουμε 1. Μέτρο = μέτρο της νέας παράστασης.
- Αποκωδικοποίηση, Μέθοδος 2:
  - Εστω κ, ο αριθμός των ψηφίων της παράστασης.
  - Κανόνας πολυωνύμου με βάρος $-2^{\kappa-1}$ στο αριστερότερο ψηφίο.

Συμπλήρωμα ως προς 2

- Κωδικοποίηση :
  - Βρίσκουμε μέτρο στο δυαδικό.
  - Αν (+) αυτή είναι και η παράστασή του.
  - Αν (-) αντιστρέφουμε όλα τα δυαδικά ψηφία και προσθέτουμε 1.
- Αποκωδικοποίηση, Μέθοδος 1:
  - Ελέγχουμε το αριστερότερο δυαδικό ψηφίο.
  - Αν 0, πρόσημο = +, μέτρο από κανόνα πολυωνύμου.
  - Αν 1, πρόσημο = -, αντιστρέφουμε όλα τα δυαδικά ψηφία, προσθέτουμε 1. Μέτρο = μέτρο της νέας παράστασης.
- Αποκωδικοποίηση, Μέθοδος 2:
  - Εστω κ, ο αριθμός των ψηφίων της παράστασης.
  - Κανόνας πολυωνύμου με βάρος $-2^{\kappa-1}$ στο αριστερότερο ψηφίο.
- Με κ ψηφία αναπαρίστανται οι ακέραιοι στο $[-2^{\kappa-1}, 2^{\kappa-1}-1]$.
- Ο μικρότερος είναι ο $100\ldots00$ και ο μεγαλύτερος ο $011\ldots11$.
- Μόνο 1 αναπαράσταση του 0 : $000\ldots00$.

Επέκταση προσήμου

- Ας υποθέσουμε τον $b_{k-1}b_{k-2}...b_1b_0$ σε "συμπλήρωμα ως προς 2".
- Σύμφωνα με τον κανόνα του πολυωνύμου, είναι ίσος με $-2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + ... + 2^1 \times b_1 + 2^0 \times b_0$.

Επέκταση προσήμου

- Ας υποθέσουμε τον $b_{k-1}b_{k-2}\ldots b_1b_0$ σε "συμπλήρωμα ως προς 2".
- Σύμφωνα με τον κανόνα του πολυωνύμου, είναι ίσος με $-2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + \ldots + 2^1 \times b_1 + 2^0 \times b_0$.
- $-2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + 2^1 \times b_1 + 2^0 \times b_0$ = $-2^k \times b_{k-1} + 2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + \ldots + 2^1 \times b_1 + 2^0 \times b_0$.
- Όμως το δεύτερο ανάπτυγμα αντιστοιχεί στον $b_{k-1}b_{k-1}b_{k-2}\ldots b_1b_0$

Επέκταση προσήμου

- Ας υποθέσουμε τον $b_{k-1}b_{k-2}\ldots b_1b_0$ σε "συμπλήρωμα ως προς 2".
- Σύμφωνα με τον κανόνα του πολυωνύμου, είναι ίσος με $-2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + \ldots + 2^1 \times b_1 + 2^0 \times b_0$.
- $-2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + 2^1 \times b_1 + 2^0 \times b_0$ = $-2^k \times b_{k-1} + 2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + \ldots + 2^1 \times b_1 + 2^0 \times b_0$.
- Όμως το δεύτερο ανάπτυγμα αντιστοιχεί στον $b_{k-1}b_{k-1}b_{k-2}\ldots b_1b_0$
- Άρα σε μια παράσταση σε συμπλήρωμα ως προς 2 μπορούμε να επεκτείνουμε την παράσταση επαναλαμβάνοντας στα αριστερά το αριστερότερο bit, χωρίς να αλλάζει η αναπαριστούμενη ποσότητα.

Επέκταση προσήμου

- Ας υποθέσουμε τον $b_{k-1}b_{k-2}\ldots b_1b_0$ σε "συμπλήρωμα ως προς 2".
- Σύμφωνα με τον κανόνα του πολυωνύμου, είναι ίσος με $-2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + \ldots + 2^1 \times b_1 + 2^0 \times b_0$.
- $-2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + 2^1 \times b_1 + 2^0 \times b_0 =$
  $-2^k \times b_{k-1} + 2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + \ldots + 2^1 \times b_1 + 2^0 \times b_0$.
- Όμως το δεύτερο ανάπτυγμα αντιστοιχεί στον $b_{k-1}b_{k-1}b_{k-2}\ldots b_1b_0$
- Άρα σε μια παράσταση σε συμπλήρωμα ως προς 2 μπορούμε να επεκτείνουμε την παράσταση επαναλαμβάνοντας στα αριστερά το αριστερότερο bit, χωρίς να αλλάζει η αναπαριστούμενη ποσότητα.
- Για τον ίδιο λόγο μπορούμε να σβήνουμε bit με την ίδια τιμή στην αριστερή πλευρά της παράστασης.

Επέκταση προσήμου

- Ας υποθέσουμε τον $b_{k-1}b_{k-2}...b_1b_0$ σε "συμπλήρωμα ως προς 2".
- Σύμφωνα με τον κανόνα του πολυωνύμου, είναι ίσος με $-2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + ... + 2^1 \times b_1 + 2^0 \times b_0$.
- $-2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + 2^1 \times b_1 + 2^0 \times b_0$ = $-2^k \times b_{k-1} + 2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + ... + 2^1 \times b_1 + 2^0 \times b_0$.
- Όμως το δεύτερο ανάπτυγμα αντιστοιχεί στον $b_{k-1}b_{k-1}b_{k-2}...b_1b_0$
- Άρα σε μια παράσταση σε συμπλήρωμα ως προς 2 μπορούμε να επεκτείνουμε την παράσταση επαναλαμβάνοντας στα αριστερά το αριστερότερο bit, χωρίς να αλλάζει η αναπαριστούμενη ποσότητα.
- Για τον ίδιο λόγο μπορούμε να σβήνουμε bit με την ίδια τιμή στην αριστερή πλευρά της παράστασης.
- Π.χ. $100_{2s} = 1100_{2s} = 11100_{2s} = ...$

Επέκταση προσήμου

- Ας υποθέσουμε τον $b_{k-1}b_{k-2}...b_1b_0$ σε "συμπλήρωμα ως προς 2".
- Σύμφωνα με τον κανόνα του πολυωνύμου, είναι ίσος με $-2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + ... + 2^1 \times b_1 + 2^0 \times b_0$.
- $-2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + 2^1 \times b_1 + 2^0 \times b_0$ = $-2^k \times b_{k-1} + 2^{k-1} \times b_{k-1} + 2^{k-2} \times b_{k-2} + ... + 2^1 \times b_1 + 2^0 \times b_0$.
- Όμως το δεύτερο ανάπτυγμα αντιστοιχεί στον $b_{k-1}b_{k-1}b_{k-2}...b_1b_0$
- Άρα σε μια παράσταση σε συμπλήρωμα ως προς 2 μπορούμε να επεκτείνουμε την παράσταση επαναλαμβάνοντας στα αριστερά το αριστερότερο bit, χωρίς να αλλάζει η αναπαριστούμενη ποσότητα.
- Για τον ίδιο λόγο μπορούμε να σβήνουμε bit με την ίδια τιμή στην αριστερή πλευρά της παράστασης.
- Π.χ. $100_{2s} = 1100_{2s} = 11100_{2s} = ...$
- Η ίδια επέκταση προσήμου ισχύει και στο συμπλήρωμα ως προς 1.

Επέκταση των αριθμητικών συστημάτων για κλάσματα

- Εστω ο αριθμός $\beta_{\kappa-1}\beta_{\kappa-2}\ldots\beta_1\beta_0,\beta_{-1}\beta_{-2}\ldots\beta_{-(\lambda-1)}\beta_{-\lambda}\ _{\rho}$.

Επέκταση των αριθμητικών συστημάτων για κλάσματα

- Εστω ο αριθμός $\beta_{\kappa-1}\beta_{\kappa-2}\ldots\beta_1\beta_0,\beta_{-1}\beta_{-2}\ldots\beta_{-(\lambda-1)}\beta_{-\lambda}\ \rho$.
- Επεκτείνοντας τις ιδιότητες των αριθμητικών συστημάτων, και σε αρνητικές δυνάμεις θέσεως, μπορούμε να θεωρούμε ότι παριστά την ποσότητα $\Pi = \sum_{i=-\lambda}^{\kappa-1}(\beta_i\rho^i)$

Επέκταση των αριθμητικών συστημάτων για κλάσματα

- Εστω ο αριθμός $\beta_{\kappa-1}\beta_{\kappa-2}\ldots\beta_1\beta_0,\beta_{-1}\beta_{-2}\ldots\beta_{-(\lambda-1)}\beta_{-\lambda}\ \rho$.
- Επεκτείνοντας τις ιδιότητες των αριθμητικών συστημάτων, και σε αρνητικές δυνάμεις θέσεως, μπορούμε να θεωρούμε ότι παριστά την ποσότητα $\Pi = \sum_{i=-\lambda}^{\kappa-1}(\beta_i\rho^i)$
- Μπορούμε επίσης να δώσουμε διαδικασία μετατροπής του κλασματικού μέρους σε κλασματικό μέρος άλλου αριθμητικού συστήματος με διαδοχικούς πολλαπλασιασμούς.

Επέκταση των αριθμητικών συστημάτων για κλάσματα

- Εστω ο αριθμός $\beta_{\kappa-1}\beta_{\kappa-2}\ldots\beta_1\beta_0,\beta_{-1}\beta_{-2}\ldots\beta_{-(\lambda-1)}\beta_{-\lambda}\ {}_{\rho}$.
- Επεκτείνοντας τις ιδιότητες των αριθμητικών συστημάτων, και σε αρνητικές δυνάμεις θέσεως, μπορούμε να θεωρούμε ότι παριστά την ποσότητα $\Pi = \sum_{i=-\lambda}^{\kappa-1}(\beta_i\rho^i)$
- Μπορούμε επίσης να δώσουμε διαδικασία μετατροπής του κλασματικού μέρους σε κλασματικό μέρος άλλου αριθμητικού συστήματος με διαδοχικούς πολλαπλασιασμούς.
- $0,\lambda_{10} = 0,\beta_{-1}\beta_{-2}\ldots\beta_{-(\lambda-1)}\beta_{-\lambda}\ {}_{\rho} =$
  $\beta_{-1}\times\rho^{-1}+\beta_{-2}\times\rho^{-2}+\ldots+\beta_{-(\lambda-1)}\times\rho^{-(\lambda-1)}+\beta_{-\lambda}\times\rho^{-\lambda} \Leftrightarrow$
  $0,\lambda_{10}\times\rho = \beta_{-1}\times\rho^{0}+\left[\beta_{-2}\times\rho^{-1}+\ldots+\beta_{-(\lambda-1)}\times\rho^{-(\lambda-2)}+\beta_{-\lambda}\times\rho^{-(\lambda-1)}\right]$

Επέκταση των αριθμητικών συστημάτων για κλάσματα

- Εστω ο αριθμός $\beta_{\kappa-1}\beta_{\kappa-2}\ldots\beta_1\beta_0,\beta_{-1}\beta_{-2}\ldots\beta_{-(\lambda-1)}\beta_{-\lambda}\ {}_{\rho}$.
- Επεκτείνοντας τις ιδιότητες των αριθμητικών συστημάτων, και σε αρνητικές δυνάμεις θέσεως, μπορούμε να θεωρούμε ότι παριστά την ποσότητα $\Pi = \sum_{i=-\lambda}^{\kappa-1}(\beta_i\rho^i)$
- Μπορούμε επίσης να δώσουμε διαδικασία μετατροπής του κλασματικού μέρους σε κλασματικό μέρος άλλου αριθμητικού συστήματος με διαδοχικούς πολλαπλασιασμούς.
- $0,\lambda_{10} = 0,\beta_{-1}\beta_{-2}\ldots\beta_{-(\lambda-1)}\beta_{-\lambda}\ {}_{\rho} =$
  $\beta_{-1}\times\rho^{-1}+\beta_{-2}\times\rho^{-2}+\ldots+\beta_{-(\lambda-1)}\times\rho^{-(\lambda-1)}+\beta_{-\lambda}\times\rho^{-\lambda} \Leftrightarrow$
  $0,\lambda_{10}\times\rho = \beta_{-1}\times\rho^{0}+\left[\beta_{-2}\times\rho^{-1}+\ldots+\beta_{-(\lambda-1)}\times\rho^{-(\lambda-2)}+\beta_{-\lambda}\times\rho^{-(\lambda-1)}\right]$
- Μετατρέψαμε έτσι ένα πρόβλημα καθορισμού λ αγνώστων σε ένα καθορισμού $\lambda-1$ και συνεπώς μπορούμε να εφαρμόσουμε αυτή τη διαδικασία επαναληπτικά έως ότου μηδενιστεί το κλασματικό μέρος.

Παράδειγμα μετατροπής κλασματικού μέρους

- $0,14_8 = ?_2$

Παράδειγμα μετατροπής κλασματικού μέρους

- $0,14_8 = ?_2$
- $0,14_8 = 1 \times 8^{-1} + 4 \times 8^{-2} = 0,125 + 0,0625 = 0,1875_{10}$

Παράδειγμα μετατροπής κλασματικού μέρους

- $0,14_8 = ?_2$
- $0,14_8 = 1 \times 8^{-1} + 4 \times 8^{-2} = 0,125 + 0,0625 = 0,1875_{10}$
- $0,1875_{10} \times 2 = 0(MSD) + 0,375$

Παράδειγμα μετατροπής κλασματικού μέρους

- $0,14_8 = ?_2$
- $0,14_8 = 1 \times 8^{-1} + 4 \times 8^{-2} = 0,125 + 0,0625 = 0,1875_{10}$
- $0,1875_{10} \times 2 = 0(MSD) + 0,375$
- $0,375_{10} \times 2 = 0 + 0,75$

Παράδειγμα μετατροπής κλασματικού μέρους

- $0,14_8 = ?_2$
- $0,14_8 = 1 \times 8^{-1} + 4 \times 8^{-2} = 0,125 + 0,0625 = 0,1875_{10}$
- $0,1875_{10} \times 2 = 0(MSD) + 0,375$
- $0,375_{10} \times 2 = 0 + 0,75$
- $0,75_{10} \times 2 = 1 + 0,5$

Παράδειγμα μετατροπής κλασματικού μέρους

- $0,14_8 = ?_2$
- $0,14_8 = 1 \times 8^{-1} + 4 \times 8^{-2} = 0,125 + 0,0625 = 0,1875_{10}$
- $0,1875_{10} \times 2 = 0(MSD) + 0,375$
- $0,375_{10} \times 2 = 0 + 0,75$
- $0,75_{10} \times 2 = 1 + 0,5$
- $0,5_{10} \times 2 = 1 + 0$

Παράδειγμα μετατροπής κλασματικού μέρους

- $0,14_8 = ?_2$
- $0,14_8 = 1 \times 8^{-1} + 4 \times 8^{-2} = 0,125 + 0,0625 = 0,1875_{10}$
- $0,1875_{10} \times 2 = 0(MSD) + 0,375$
- $0,375_{10} \times 2 = 0 + 0,75$
- $0,75_{10} \times 2 = 1 + 0,5$
- $0,5_{10} \times 2 = 1 + 0$
- Συνεπώς $0,14_8 = 0,0011_2$

Παράδειγμα μετατροπής κλασματικού μέρους

- $0,14_8 = ?_2$
- $0,14_8 = 1 \times 8^{-1} + 4 \times 8^{-2} = 0,125 + 0,0625 = 0,1875_{10}$
- $0,1875_{10} \times 2 = 0(MSD) + 0,375$
- $0,375_{10} \times 2 = 0 + 0,75$
- $0,75_{10} \times 2 = 1 + 0,5$
- $0,5_{10} \times 2 = 1 + 0$
- Συνεπώς $0,14_8 = 0,0011_2$
- Κανένας πιο γρήγορος τρόπος ?

Μη πεπερασμένες αναπαραστάσεις & προβλήματα εφαρμογής

- $0,2_{10} = ?_2$

Μη πεπερασμένες αναπαραστάσεις & προβλήματα εφαρμογής

- $0,2_{10} = ?_2$
- $0,2_{10} \times 2 = 0 + 0,4$

Μη πεπερασμένες αναπαραστάσεις & προβλήματα εφαρμογής

- $0,2_{10} = ?_2$
- $0,2_{10} \times 2 = 0 + 0,4$
- $0,4_{10} \times 2 = 0 + 0,8$

Μη πεπερασμένες αναπαραστάσεις & προβλήματα εφαρμογής

- $0,2_{10} = ?_2$
- $0,2_{10} \times 2 = 0 + 0,4$
- $0,4_{10} \times 2 = 0 + 0,8$
- $0,8_{10} \times 2 = 1 + 0,6$

Μη πεπερασμένες αναπαραστάσεις & προβλήματα εφαρμογής

- $0,2_{10} = ?_2$
- $0,2_{10} \times 2 = 0 + 0,4$
- $0,4_{10} \times 2 = 0 + 0,8$
- $0,8_{10} \times 2 = 1 + 0,6$
- $0,6_{10} \times 2 = 1 + 0,2$!!!

Μη πεπερασμένες αναπαραστάσεις & προβλήματα εφαρμογής

- $0,2_{10} = ?_2$
- $0,2_{10} \times 2 = 0 + 0,4$
- $0,4_{10} \times 2 = 0 + 0,8$
- $0,8_{10} \times 2 = 1 + 0,6$
- $0,6_{10} \times 2 = 1 + 0,2$!!!
- Συνεπώς το $0,2_{10}$ δεν έχει πεπερασμένη αναπαράσταση!

Μη πεπερασμένες αναπαραστάσεις & προβλήματα εφαρμογής

- $0,2_{10} = ?_2$
- $0,2_{10} \times 2 = 0 + 0,4$
- $0,4_{10} \times 2 = 0 + 0,8$
- $0,8_{10} \times 2 = 1 + 0,6$
- $0,6_{10} \times 2 = 1 + 0,2$!!!
- Συνεπώς το $0,2_{10}$ δεν έχει πεπερασμένη αναπαράσταση!
- $0,1_3 = 0,3333\ldots\ _{10}$!

Μη πεπερασμένες αναπαραστάσεις & προβλήματα εφαρμογής

- $0,2_{10} = ?_2$
- $0,2_{10} \times 2 = 0 + 0,4$
- $0,4_{10} \times 2 = 0 + 0,8$
- $0,8_{10} \times 2 = 1 + 0,6$
- $0,6_{10} \times 2 = 1 + 0,2$!!!
- Συνεπώς το $0,2_{10}$ δεν έχει πεπερασμένη αναπαράσταση!
- $0,1_3 = 0,3333\ldots\ _{10}$!
- Η υποδιαστολή πως αποθηκεύεται ?

Μη πεπερασμένες αναπαραστάσεις & προβλήματα εφαρμογής

- $0,2_{10} = ?_2$
- $0,2_{10} \times 2 = 0 + 0,4$
- $0,4_{10} \times 2 = 0 + 0,8$
- $0,8_{10} \times 2 = 1 + 0,6$
- $0,6_{10} \times 2 = 1 + 0,2$!!!
- Συνεπώς το $0,2_{10}$ δεν έχει πεπερασμένη αναπαράσταση!
- $0,1_3 = 0,3333\ldots\ _{10}$!
- Η υποδιαστολή πως αποθηκεύεται ?
- Ας υποθέσουμε αρχικά ότι δεν αποθηκεύεται αλλά υπονοείται.

Σταθερή υποδιαστολή

- Εύρος αναπαράστασης = 16 δυαδικά ψηφία
- Προσέγγιση 1 : πρόσημο xxxxxxxxxxx (.) xxxx

Σταθερή υποδιαστολή

- Εύρος αναπαράστασης = 16 δυαδικά ψηφία
- Προσέγγιση 1 : πρόσημο xxxxxxxxxxx (.) xxxx
  - $0,2_{10}$ = 0 00000000000 0011
  - Απώλεια ακρίβειας $> 6\%$.
  - Αναπαριστά αριθμούς της τάξης 2^{11}.

Σταθερή υποδιαστολή

- Εύρος αναπαράστασης = 16 δυαδικά ψηφία
- Προσέγγιση 1 : πρόσημο xxxxxxxxxxx (.) xxxx
  - $0,2_{10}$ = 0 00000000000 0011
  - Απώλεια ακρίβειας $> 6\%$.
  - Αναπαριστά αριθμούς της τάξης 2^{11}.
- Προσέγγιση 2 : πρόσημο xxxx (.) xxxxxxxxxxx

Σταθερή υποδιαστολή

- Εύρος αναπαράστασης = 16 δυαδικά ψηφία
- Προσέγγιση 1 : πρόσημο xxxxxxxxxxx (.) xxxx
  - $0,2_{10}$ = 0 00000000000 0011
  - Απώλεια ακρίβειας $>$ 6%.
  - Αναπαριστά αριθμούς της τάξης 2^{11}.
- Προσέγγιση 2 : πρόσημο xxxx (.) xxxxxxxxxxx
  - $0,2_{10}$ = 0 0000 00110011001
  - Απώλεια ακρίβειας $<<$ 1%.
  - Αναπαριστά αριθμούς της τάξης 2^4.

Σταθερή υποδιαστολή

- Εύρος αναπαράστασης = 16 δυαδικά ψηφία
- Προσέγγιση 1 : πρόσημο xxxxxxxxxxx (.) xxxx
  - $0,2_{10}$ = 0 00000000000 0011
  - Απώλεια ακρίβειας $>$ 6%.
  - Αναπαριστά αριθμούς της τάξης 2^{11}.
- Προσέγγιση 2 : πρόσημο xxxx (.) xxxxxxxxxxx
  - $0,2_{10}$ = 0 0000 00110011001
  - Απώλεια ακρίβειας $<<$ 1%.
  - Αναπαριστά αριθμούς της τάξης 2^4.
- Προσυμφωνημένη θέση υποδιαστολής $\Rightarrow$ παζάρι ακρίβειας & εύρους.

Σταθερή υποδιαστολή

- Εύρος αναπαράστασης = 16 δυαδικά ψηφία
- Προσέγγιση 1 : πρόσημο xxxxxxxxxxx (.) xxxx
  - $0,2_{10}$ = 0 00000000000 0011
  - Απώλεια ακρίβειας $>$ 6%.
  - Αναπαριστά αριθμούς της τάξης 2^{11}.
- Προσέγγιση 2 : πρόσημο xxxx (.) xxxxxxxxxxx
  - $0,2_{10}$ = 0 0000 00110011001
  - Απώλεια ακρίβειας $<<$ 1%.
  - Αναπαριστά αριθμούς της τάξης 2^4.
- Προσυμφωνημένη θέση υποδιαστολής $\Rightarrow$ παζάρι ακρίβειας & εύρους.
- Στη ζωή μας όμως χρειαζόμαστε & πολύ μεγάλους αριθμούς & πολύ μεγάλη ακρίβεια.

Σταθερή υποδιαστολή

- Εύρος αναπαράστασης = 16 δυαδικά ψηφία
- Προσέγγιση 1 : πρόσημο xxxxxxxxxxx (.) xxxx
  - $0,2_{10}$ = 0 00000000000 0011
  - Απώλεια ακρίβειας $>$ 6%.
  - Αναπαριστά αριθμούς της τάξης 2^{11}.
- Προσέγγιση 2 : πρόσημο xxxx (.) xxxxxxxxxxx
  - $0,2_{10}$ = 0 0000 00110011001
  - Απώλεια ακρίβειας $<<$ 1%.
  - Αναπαριστά αριθμούς της τάξης 2^4.
- Προσυμφωνημένη θέση υποδιαστολής $\Rightarrow$ παζάρι ακρίβειας & εύρους.
- Στη ζωή μας όμως χρειαζόμαστε & πολύ μεγάλους αριθμούς & πολύ μεγάλη ακρίβεια.
- Για να τα πετύχω και τα 2 με σταθερή υποδιαστολή θα χρειαζόμουν 80+ ψηφία / πραγματικό αριθμό $\Rightarrow$ τεράστια ποσότητα μνήμης $\Rightarrow$ πολύ υψηλό κόστος.

Κινητή υποδιαστολή

- Scientific notation : $\Sigma \times B^E$.

Κινητή υποδιαστολή

- Scientific notation : $\Sigma \times B^E$.
- Σ = συντελεστής (significant). Πραγματικός.

Κινητή υποδιαστολή

- Scientific notation : $\Sigma \times B^E$.
- Σ = συντελεστής (significant). Πραγματικός.
- Β = βάση αριθμητικού συστήματος (base). Φυσικός. Συνήθως υπονοείται.

Κινητή υποδιαστολή

- Scientific notation : $\Sigma \times B^E$.
- Σ = συντελεστής (significant). Πραγματικός.
- Β = βάση αριθμητικού συστήματος (base). Φυσικός. Συνήθως υπονοείται.
- Ε = εκθέτης (exponent). Ακέραιος.

Κινητή υποδιαστολή

- Scientific notation : $\Sigma \times B^E$.
- Σ = συντελεστής (significant). Πραγματικός.
- Β = βάση αριθμητικού συστήματος (base). Φυσικός. Συνήθως υπονοείται.
- Ε = εκθέτης (exponent). Ακέραιος.
- Ενας αριθμός μπορεί να έχει πολλές αναπαραστάσεις. Π.χ. $375 \times 10^0 = 3,75 \times 10^2 = 0,375 \times 10^3$

Κινητή υποδιαστολή

- Scientific notation : $\Sigma \times B^E$.
- Σ = συντελεστής (significant). Πραγματικός.
- Β = βάση αριθμητικού συστήματος (base). Φυσικός. Συνήθως υπονοείται.
- Ε = εκθέτης (exponent). Ακέραιος.
- Ενας αριθμός μπορεί να έχει πολλές αναπαραστάσεις. Π.χ. $375 \times 10^0 = 3,75 \times 10^2 = 0,375 \times 10^3$
- Κανονική αναπαράσταση : η υποδιαστολή βρίσκεται αριστερά του αριστερότερου μη μηδενικού στοιχείου.

Παράδειγμα με αυθαίρετες υποθέσεις

- Καθένας μπορεί να κάνει αυθαίρετες επιλογές. Εστω ότι έχουμε 16 δυαδικά ψηφία και κάνουμε τις εξής υποθέσεις :

Παράδειγμα με αυθαίρετες υποθέσεις

- Καθένας μπορεί να κάνει αυθαίρετες επιλογές. Εστω ότι έχουμε 16 δυαδικά ψηφία και κάνουμε τις εξής υποθέσεις :
  - Βάση =8
  - Εκθέτης 3 δυαδικών ψηφίων σε πόλωση κατά 4.
  - Συντελεστής σε πρόσημο μέτρο. Για το μέτρο 4 οκταδικά ψηφία.
  - Κανονικοποιημένες μορφές μόνο.
  - Μορφή : Πρόσημο Συντελεστή-Εκθέτης-Συντελεστής.

Παράδειγμα με αυθαίρετες υποθέσεις

- Καθένας μπορεί να κάνει αυθαίρετες επιλογές. Εστω ότι έχουμε 16 δυαδικά ψηφία και κάνουμε τις εξής υποθέσεις :
  - Βάση =8
  - Εκθέτης 3 δυαδικών ψηφίων σε πόλωση κατά 4.
  - Συντελεστής σε πρόσημο μέτρο. Για το μέτρο 4 οκταδικά ψηφία.
  - Κανονικοποιημένες μορφές μόνο.
  - Μορφή : Πρόσημο Συντελεστή-Εκθέτης-Συντελεστής.
- Ας δούμε την αναπαράσταση του $-16,125_{10}$

Παράδειγμα με αυθαίρετες υποθέσεις

- Καθένας μπορεί να κάνει αυθαίρετες επιλογές. Εστω ότι έχουμε 16 δυαδικά ψηφία και κάνουμε τις εξής υποθέσεις :
  - Βάση =8
  - Εκθέτης 3 δυαδικών ψηφίων σε πόλωση κατά 4.
  - Συντελεστής σε πρόσημο μέτρο. Για το μέτρο 4 οκταδικά ψηφία.
  - Κανονικοποιημένες μορφές μόνο.
  - Μορφή : Πρόσημο Συντελεστή-Εκθέτης-Συντελεστής.
- Ας δούμε την αναπαράσταση του $-16,125_{10}$
- $-16,125_{10} = -20,1_8 = -20,1 \times 8^0 = -0,201 \times 8^2$

Παράδειγμα με αυθαίρετες υποθέσεις

- Καθένας μπορεί να κάνει αυθαίρετες επιλογές. Εστω ότι έχουμε 16 δυαδικά ψηφία και κάνουμε τις εξής υποθέσεις :
  - Βάση =8
  - Εκθέτης 3 δυαδικών ψηφίων σε πόλωση κατά 4.
  - Συντελεστής σε πρόσημο μέτρο. Για το μέτρο 4 οκταδικά ψηφία.
  - Κανονικοποιημένες μορφές μόνο.
  - Μορφή : Πρόσημο Συντελεστή-Εκθέτης-Συντελεστής.
- Ας δούμε την αναπαράσταση του $-16,125_{10}$
- $-16,125_{10} = -20,1_8 = -20,1 \times 8^0 = -0,201 \times 8^2$
- Πρόσημο Συντελεστή =1

Παράδειγμα με αυθαίρετες υποθέσεις

- Καθένας μπορεί να κάνει αυθαίρετες επιλογές. Εστω ότι έχουμε 16 δυαδικά ψηφία και κάνουμε τις εξής υποθέσεις :
  - Βάση =8
  - Εκθέτης 3 δυαδικών ψηφίων σε πόλωση κατά 4.
  - Συντελεστής σε πρόσημο μέτρο. Για το μέτρο 4 οκταδικά ψηφία.
  - Κανονικοποιημένες μορφές μόνο.
  - Μορφή : Πρόσημο Συντελεστή-Εκθέτης-Συντελεστής.
- Ας δούμε την αναπαράσταση του $-16,125_{10}$
- $-16,125_{10} = -20,1_8 = -20,1 \times 8^0 = -0,201 \times 8^2$
- Πρόσημο Συντελεστή =1
- Εκθέτης = 2 και σε πόλωση κατά 4, $6_{10} = 110_2$.

Παράδειγμα με αυθαίρετες υποθέσεις

- Καθένας μπορεί να κάνει αυθαίρετες επιλογές. Εστω ότι έχουμε 16 δυαδικά ψηφία και κάνουμε τις εξής υποθέσεις :
  - Βάση =8
  - Εκθέτης 3 δυαδικών ψηφίων σε πόλωση κατά 4.
  - Συντελεστής σε πρόσημο μέτρο. Για το μέτρο 4 οκταδικά ψηφία.
  - Κανονικοποιημένες μορφές μόνο.
  - Μορφή : Πρόσημο Συντελεστή-Εκθέτης-Συντελεστής.
- Ας δούμε την αναπαράσταση του $-16,125_{10}$
- $-16,125_{10} = -20,1_8 = -20,1 \times 8^0 = -0,201 \times 8^2$
- Πρόσημο Συντελεστή =1
- Εκθέτης = 2 και σε πόλωση κατά 4, $6_{10} = 110_2$.
- Συντελεστής = 2010_8 = $010\ 000\ 001\ 000_2$

Παράδειγμα με αυθαίρετες υποθέσεις

- Καθένας μπορεί να κάνει αυθαίρετες επιλογές. Εστω ότι έχουμε 16 δυαδικά ψηφία και κάνουμε τις εξής υποθέσεις :
  - Βάση =8
  - Εκθέτης 3 δυαδικών ψηφίων σε πόλωση κατά 4.
  - Συντελεστής σε πρόσημο μέτρο. Για το μέτρο 4 οκταδικά ψηφία.
  - Κανονικοποιημένες μορφές μόνο.
  - Μορφή : Πρόσημο Συντελεστή-Εκθέτης-Συντελεστής.
- Ας δούμε την αναπαράσταση του $-16,125_{10}$
- $-16,125_{10} = -20,1_8 = -20,1 \times 8^0 = -0,201 \times 8^2$
- Πρόσημο Συντελεστή =1
- Εκθέτης = 2 και σε πόλωση κατά 4, $6_{10} = 110_2$.
- Συντελεστής = 2010_8 = $010\ 000\ 001\ 000_2$
- Συνολική αναπαράσταση = 1 110 010 000 001 000.

Η ανάγκη για ένα πρότυπο (standard)

- Οταν ο καθένας κάνει τις δικές του αυθαίρετες επιλογές :
  - Δεν μπορούν να ανταλλάσσονται δεδομένα μεταξύ υπολογιστικών συστημάτων.
  - Πρέπει να αλλάζουν τα προγράμματα επεξεργασίας των αριθμητικών δεδομένων.

Η ανάγκη για ένα πρότυπο (standard)

- Οταν ο καθένας κάνει τις δικές του αυθαίρετες επιλογές :
  - Δεν μπορούν να ανταλλάσσονται δεδομένα μεταξύ υπολογιστικών συστημάτων.
  - Πρέπει να αλλάζουν τα προγράμματα επεξεργασίας των αριθμητικών δεδομένων.
- Συνεπώς γεννιέται η ανάγκη ενός προτύπου.

Η ανάγκη για ένα πρότυπο (standard)

- Οταν ο καθένας κάνει τις δικές του αυθαίρετες επιλογές :
  - Δεν μπορούν να ανταλλάσσονται δεδομένα μεταξύ υπολογιστικών συστημάτων.
  - Πρέπει να αλλάζουν τα προγράμματα επεξεργασίας των αριθμητικών δεδομένων.
- Συνεπώς γεννιέται η ανάγκη ενός προτύπου.
- Τα πρότυπα προτείνονται από κάποια επιτροπή αφού αυτή μελετήσει τις ικανότητες και τις αδυναμίες κάθε πρότασης.

Η ανάγκη για ένα πρότυπο (standard)

- Οταν ο καθένας κάνει τις δικές του αυθαίρετες επιλογές :
  - Δεν μπορούν να ανταλλάσσονται δεδομένα μεταξύ υπολογιστικών συστημάτων.
  - Πρέπει να αλλάζουν τα προγράμματα επεξεργασίας των αριθμητικών δεδομένων.
- Συνεπώς γεννιέται η ανάγκη ενός προτύπου.
- Τα πρότυπα προτείνονται από κάποια επιτροπή αφού αυτή μελετήσει τις ικανότητες και τις αδυναμίες κάθε πρότασης.
- Η πρόταση κάθε επιτροπής υποβάλλεται με αριθμό πρωτοκόλλου για έγκριση σε παγκόσμιους οργανισμούς πρωτοτυποποίησης (ISO, OSI, IEEE, ...)

Η ανάγκη για ένα πρότυπο (standard)

- Οταν ο καθένας κάνει τις δικές του αυθαίρετες επιλογές :
  - Δεν μπορούν να ανταλλάσσονται δεδομένα μεταξύ υπολογιστικών συστημάτων.
  - Πρέπει να αλλάζουν τα προγράμματα επεξεργασίας των αριθμητικών δεδομένων.
- Συνεπώς γεννιέται η ανάγκη ενός προτύπου.
- Τα πρότυπα προτείνονται από κάποια επιτροπή αφού αυτή μελετήσει τις ικανότητες και τις αδυναμίες κάθε πρότασης.
- Η πρόταση κάθε επιτροπής υποβάλλεται με αριθμό πρωτοκόλλου για έγκριση σε παγκόσμιους οργανισμούς πρωτοτυποποίησης (ISO, OSI, IEEE, ...)
- Το πρότυπο που προκύπτει συνήθως αναφέρεται με τον αριθμό πρωτοκόλλου.

Η ανάγκη για ένα πρότυπο (standard)

- Οταν ο καθένας κάνει τις δικές του αυθαίρετες επιλογές :
  - Δεν μπορούν να ανταλλάσσονται δεδομένα μεταξύ υπολογιστικών συστημάτων.
  - Πρέπει να αλλάζουν τα προγράμματα επεξεργασίας των αριθμητικών δεδομένων.
- Συνεπώς γεννιέται η ανάγκη ενός προτύπου.
- Τα πρότυπα προτείνονται από κάποια επιτροπή αφού αυτή μελετήσει τις ικανότητες και τις αδυναμίες κάθε πρότασης.
- Η πρόταση κάθε επιτροπής υποβάλλεται με αριθμό πρωτοκόλλου για έγκριση σε παγκόσμιους οργανισμούς πρωτοτυποποίησης (ISO, OSI, IEEE, ...)
- Το πρότυπο που προκύπτει συνήθως αναφέρεται με τον αριθμό πρωτοκόλλου.
- Πρότυπο αναπαράστασης κινητής υποδιαστολής : IEEE 754.

IEEE 754

- Υπάρχουν παραστάσεις απλής και διπλής ακρίβειας με 32 και 64 δυαδικά ψηφία αντίστοιχα.
- Βάση = 2.
- Συντελεστής σε πρόσημο μέτρο. (23 και 52 ψηφία αντίστοιχα + 1 προσήμου + 1 υποννοούμενο).
- Εκθέτης σε πόλωση (κατά 127 και κατά 1023 αντίστοιχα με 8 και 11 ψηφία αντίστοιχα).
- Μορφή : πρόσημο συντελεστή (κ), εκθέτης (E), συντελεστής (Σ).

IEEE 754

- Υπάρχουν παραστάσεις απλής και διπλής ακρίβειας με 32 και 64 δυαδικά ψηφία αντίστοιχα.
- Βάση = 2.
- Συντελεστής σε πρόσημο μέτρο. (23 και 52 ψηφία αντίστοιχα + 1 προσήμου + 1 υποννοούμενο).
- Εκθέτης σε πόλωση (κατά 127 και κατά 1023 αντίστοιχα με 8 και 11 ψηφία αντίστοιχα).
- Μορφή : πρόσημο συντελεστή (κ), εκθέτης (E), συντελεστής (Σ).
- Κανονικοποίηση με μορφή 1,xxxx... Μπορώ να μην αποθηκεύω το 1 !
- 000...00 και 111...11 στον εκθέτη υποδεικνύουν ειδικές περιπτώσεις.
- Μια παράσταση του IEEE 754 υποδεικνύει τη ποσότητα : $(-1)^{\kappa} \times (1 + \Sigma) \times 2^{(E - \text{πόλωση})}$

Παράδειγμα αναπαράστασης σε 754

- $-12,625_{10} = ?_{SP-754}$

Παράδειγμα αναπαράστασης σε 754

- $-12,625_{10} = ?_{SP-754}$
- $12_{10} = 1100_2$. $0,625_{10} = 0,101_2$. $-12,625_{10} = -1100,101_2$.

Παράδειγμα αναπαράστασης σε 754

- $-12,625_{10} = ?_{SP-754}$
- $12_{10} = 1100_2$. $0,625_{10} = 0,101_2$. $-12,625_{10} = -1100,101_2$.
- $-1100,101_2 = -1100,101_2 \times 2^0 = -1,100101_2 \times 2^3$

Παράδειγμα αναπαράστασης σε 754

- $-12,625_{10}$ = $?_{SP-754}$
- $12_{10} = 1100_2$. $0,625_{10} = 0,101_2$. $-12,625_{10} = -1100,101_2$.
- $-1100,101_2 = -1100,101_2 \times 2^0 = -1,100101_2 \times 2^3$
- Πρόσημο = 1. Εκθέτης=3 και σε πόλωση κατά 127 10000010_2. Συντελεστής χωρίς το hidden bit 1001010000...0

Παράδειγμα αναπαράστασης σε 754

- $-12,625_{10}$ = $?_{SP-754}$
- $12_{10} = 1100_2$. $0,625_{10} = 0,101_2$. $-12,625_{10} = -1100,101_2$.
- $-1100,101_2 = -1100,101_2 \times 2^0 = -1,100101_2 \times 2^3$
- Πρόσημο = 1. Εκθέτης=3 και σε πόλωση κατά 127 10000010_2. Συντελεστής χωρίς το hidden bit 1001010000...0
- Μορφή : 1 10000010 10010100000000000000000

Παράδειγμα αναπαράστασης σε 754

- $-12,625_{10}$ = $?_{SP-754}$
- $12_{10} = 1100_2$. $0,625_{10} = 0,101_2$. $-12,625_{10} = -1100,101_2$.
- $-1100,101_2 = -1100,101_2 \times 2^0 = -1,100101_2 \times 2^3$
- Πρόσημο = 1. Εκθέτης=3 και σε πόλωση κατά 127 10000010_2. Συντελεστής χωρίς το hidden bit 1001010000...0
- Μορφή : 1 10000010 10010100000000000000000
- Δεκαεξαδικά (Hex) : C14A0000

Παράδειγμα αναπαράστασης σε 754

- $-12,625_{10}$ = $?_{SP-754}$
- $12_{10} = 1100_2$. $0,625_{10} = 0,101_2$. $-12,625_{10} = -1100,101_2$.
- $-1100,101_2 = -1100,101_2 \times 2^0 = -1,100101_2 \times 2^3$
- Πρόσημο = 1. Εκθέτης=3 και σε πόλωση κατά 127 10000010_2. Συντελεστής χωρίς το hidden bit 1001010000...0
- Μορφή : 1 10000010 10010100000000000000000
- Δεκαεξαδικά (Hex) : C14A0000
- 0 01111110 11010...0 = $?_{10}$

Παράδειγμα αναπαράστασης σε 754

- $-12,625_{10}$ = $?_{SP-754}$
- $12_{10} = 1100_2$. $0,625_{10} = 0,101_2$. $-12,625_{10} = -1100,101_2$.
- $-1100,101_2 = -1100,101_2 \times 2^0 = -1,100101_2 \times 2^3$
- Πρόσημο = 1. Εκθέτης=3 και σε πόλωση κατά 127 10000010_2. Συντελεστής χωρίς το hidden bit 1001010000...0
- Μορφή : 1 10000010 10010100000000000000000
- Δεκαεξαδικά (Hex) : C14A0000
- 0 01111110 11010...0 = $?_{10}$
- Θετικός. Εκθέτης=126. Αφαιρώ πόλωση, Εκθέτης = -1. 1+Σ = 1,1101.
- Αρα πρόκειται για τον $+1,1101_2 \times 2^{-1} = 0,11101_2 \times 2^0 = 0,90625_{10}$.

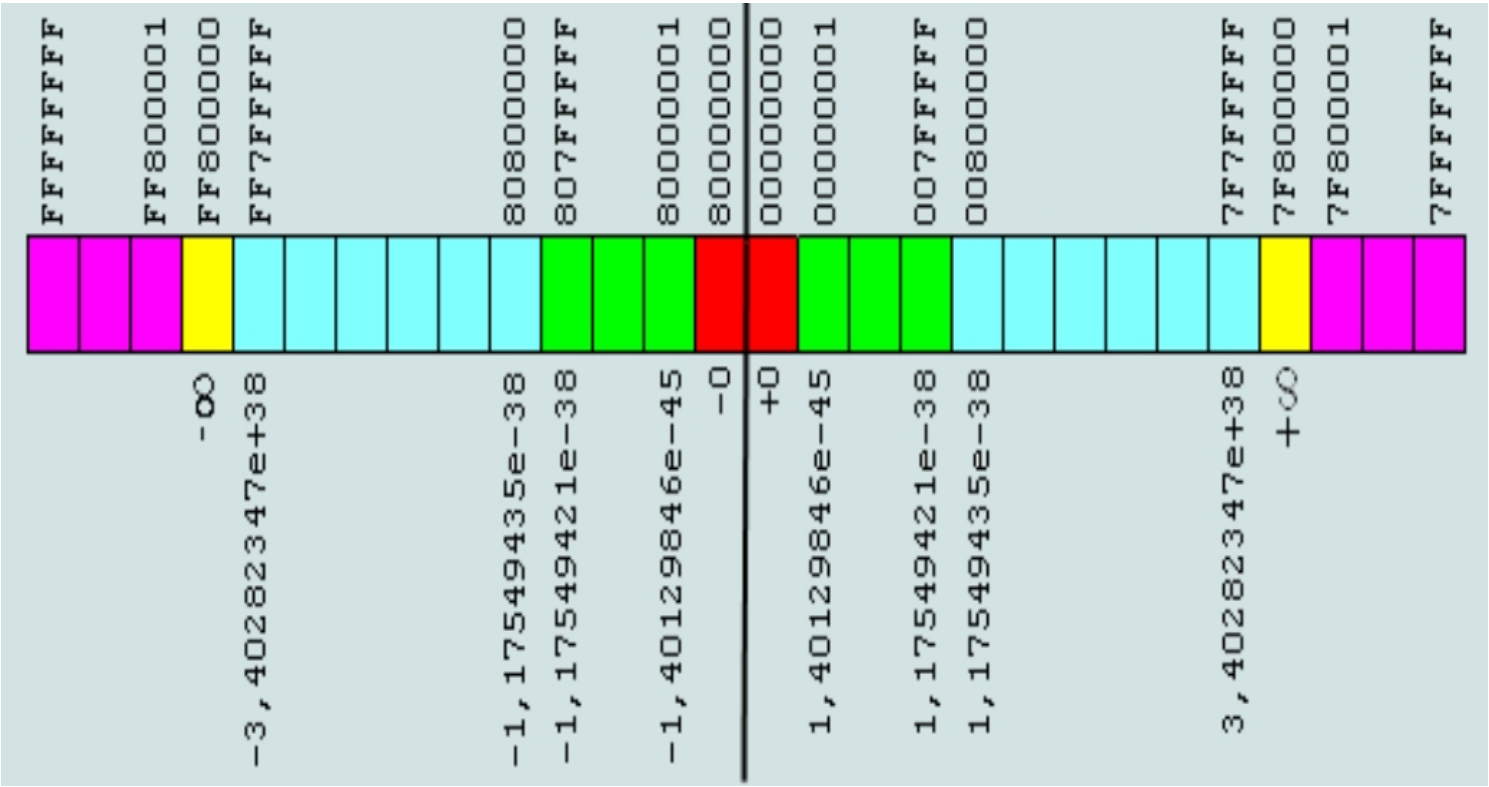
Ειδικές περιπτώσεις

Special Cases

Infinity	0 11111111 00000000000000000000000
Negative infinity	1 11111111 00000000000000000000000
Zero	0 00000000 00000000000000000000000
Negative zero	1 00000000 00000000000000000000000
Not a number*	x 11111111 xxxxxxxxxxxxxxxxxxxxxxx

*used for meaningless operations such as division by zero and square roots of negative numbers

SP IEEE 754 και η ευθεία των πραγματικών



Κώδικας BCD: B(inary)C(oded)D(ecimal)

- Κάθε ψηφίο του δεκαδικού συστήματος αντικαθίσταται από 4 δυαδικά ψηφία με βάρη.
- Οι δυαδικές αναπαραστάσεις δεν χρησιμοποιούν τα ελάχιστα δυνατά ψηφία.
- Μπορεί τα βάρη να επιτρέπουν διπλή αναπαράσταση του ίδιου ψηφίου, αλλά χρησιμοποιείται μόνο μία συγκεκριμένη.
- Οι αναπαραστάσεις των δυαδικών ψηφίων που δεν χρησιμοποιούνται θεωρούνται μη έγκυρες (invalid).
- + : Εύκολη μετατροπή από και προς το δεκαδικό, εύκολη απεικόνιση σε εκτυπωτές και οθόνες.
- - : Στις μαθηματικές πράξεις χρειάζεται να ελέγχουμε και να διορθώνουμε τα αποτελέσματα.

Κώδικες BCD

Decimal digit	BCD Code		
	8 4 2 1	4 2 2 1	5 4 2 1
0	0 0 0 0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1	0 0 1 1
4	0 1 0 0	1 0 0 0	0 1 0 0
5	0 1 0 1	0 0 1 1	1 0 0 1
6	0 1 1 0	1 1 0 0	1 0 1 0
7	0 1 1 1	1 1 0 1	
8	1 0 0 0	1 1 1 0	1 0 1 1
9	1 0 0 1	1 1 1 1	1 1 0 0

- Ο πιο διαδεδομένος είναι ο BCD-8421.
- $137_{10} \rightarrow 0001\ 0011\ 0111_{8421}$ Χρησιμοποιεί 12 ψηφία αντί 8 της ελάχιστης αναπαράστασης !
- $47_{10} + 83_{10} = 130_{10}$
- $(0100\ 0111 + 1000\ 0011)_{8421} = 1100\ 1010_{8421}$!!!
- Απαιτείται διόρθωση των μη έγκυρων αναπαραστάσεων με πρόσθεση του 6_{10} σε κάθε μη έγκυρο ψηφίο.
- $1100\ 1010 + 0110\ 0110 = 0001\ 0011\ 0000_{8421}$

Συμπληρούμενοι Κώδικες BCD

α/α	8421	2421	642–3
0	0000	0000	0000
1	0001	0001	0101
2	0010	0010	0010
3	0011	0011	1001
4	0100	0100	0100
5	0101	1011	1011
6	0110	1100	0110
7	0111	1101	1101
8	1000	1110	1010
9	1001	1111	1111

- Με προσεκτική επιλογή των αναπαραστάσεων σε κάποιους BCD κώδικες μπορώ να πετύχω η αναπαράσταση του συμπληρώματος ως προς 9 ενός αριθμού να προκύπτει συμπληρώνοντας όλα τα δυαδικά ψηφία της αναπαράστασής του.
- Αυτοί οι κώδικες ονομάζονται συμπληρούμενοι.
- Οι Aiken 2-4-2-1 και 6-4-2-(-3) είναι παραδείγματα τέτοιων κωδίκων.
- $47_{10} \rightarrow 0100\ 1101_{Aiken}$
- Ο -47 σε συμπλήρωμα ως προς 9 είναι ο 52_{9s}.
- $52_{9s} \rightarrow 1011\ 0010_{Aiken}$

Κώδικας Gray (Ανακλαστικός δυαδικός κώδικας)

- Στον κώδικα Gray κωδικοποιούμε διαδοχικούς αριθμούς με δυαδικές παραστάσεις που διαφέρουν σε 1 μόνο ψηφίο. Πχ. στον κώδικα με 4 bit: $3_{10} \rightarrow 0010$, $4_{10} \rightarrow 0110$.
- Μπορούμε να κατασκευάσουμε τον κώδικα Gray k ψηφίων κατοπτρίζοντας αυτόν των $k-1$ ψηφίων και επισυνάπτοντας στην αριστερότερη θέση 0 στο πάνω μισό και 1 στο κάτω.
- Ένας αριθμός που παρίσταται στο δυαδικό με $n > 1$ ψηφία ως $b_{n-1}b_{n-2}\ldots b_0$, κωδικοποιείται στον κώδικα Gray ως $b_{n-1}(b_{n-1} \bigoplus b_{n-2})(b_{n-2} \bigoplus b_{n-3})\ldots(b_1 \bigoplus b_0)$, όπου $\bigoplus$ συμβολίζει τη λογική συνάρτηση XOR που δίνει 1 αν τα δύο bit που λαμβάνει ως είσοδο είναι διαφορετικά και 0 αλλιώς.
- Ένας αριθμός που παρίσταται στον κώδικα Gray με $n > 1$ ψηφία ως $g_{n-1}g_{n-2}\ldots g_0$, στο δυαδικό είναι ο $b_{n-1}(g_{n-2} \bigoplus b_{n-1})(g_{n-3} \bigoplus b_{n-2})\ldots(g_0 \bigoplus b_1)$, με $b_{n-1} = g_{n-1}$.

Αναπαράσταση αλφαριθμητικών χαρακτήρων - ASCII

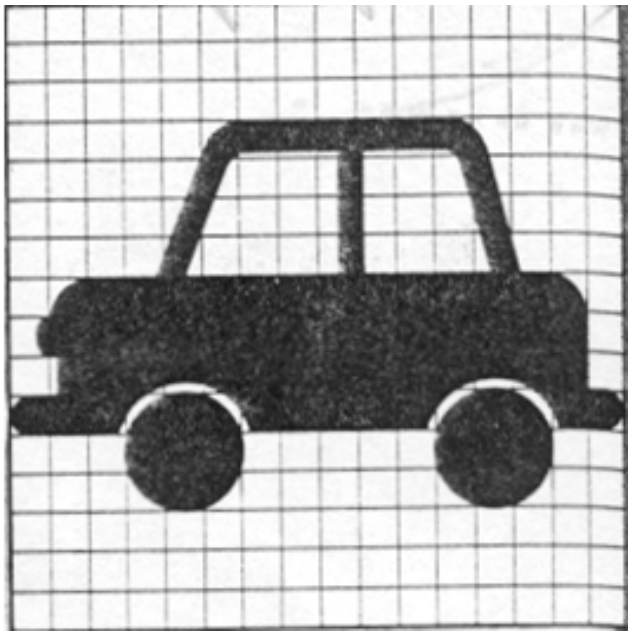
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

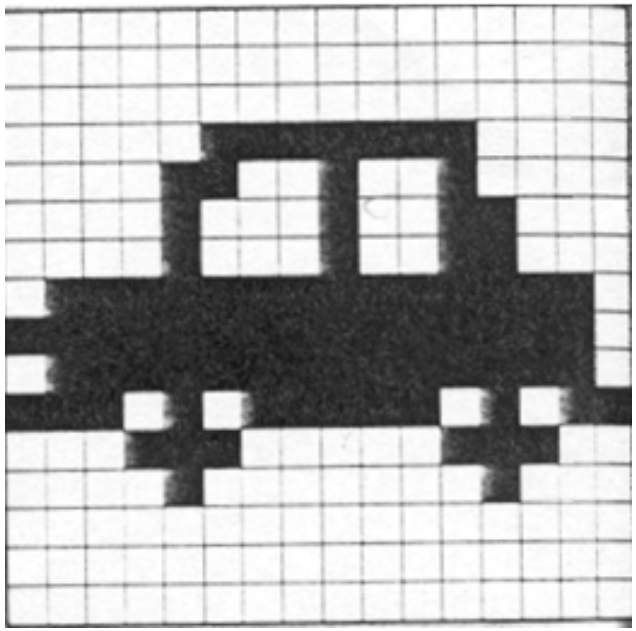
Επεκτείνοντας τον ASCII

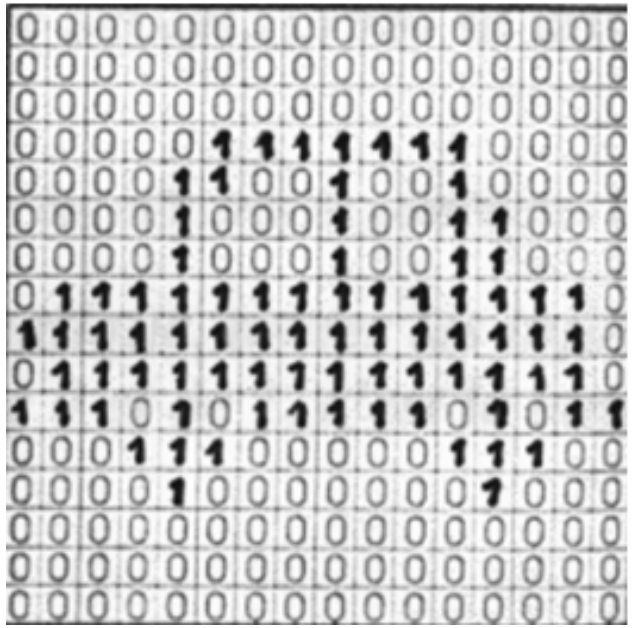
- Οι 128 αναπαραστάσεις του ASCII είναι εξαιρετικά λίγες.
- Αδύνατον να χωρέσουν 2 αλφάβητα και οι ειδικοί χαρακτήρες ελέγχου.
- EBCDIC : 8 δυαδικά ψηφία.
- Παγκόσμιο Standard : Unicode. Χρησιμοποιεί 16 δυαδικά ψηφία. Version 2.0 με 38.885 χαρακτήρες.
- Επόμενο Standard ήδη έτοιμο. 32-bit ISO 10646 Universal Character Set (UCS-4).

Αναπαράσταση εικόνας

- Οι εικόνες μπορεί να είναι διτονικές (μαύρο - άσπρο, πράσινο - μαύρο) ή συνεχούς τόνου. Στις συνεχούς τόνου ανήκουν οι εικόνες κλίμακας του γκρίζου και οι έγχρωμες.
- Κατάτμηση της εικόνας σε πολύ μικρές κουκίδες (pixels). Οσο περισσότερα τόσο πιο μεγάλη η ακρίβεια αναπαράστασης αλλά και τόσο μεγαλύτερος ο αποθηκευτικός χώρος.





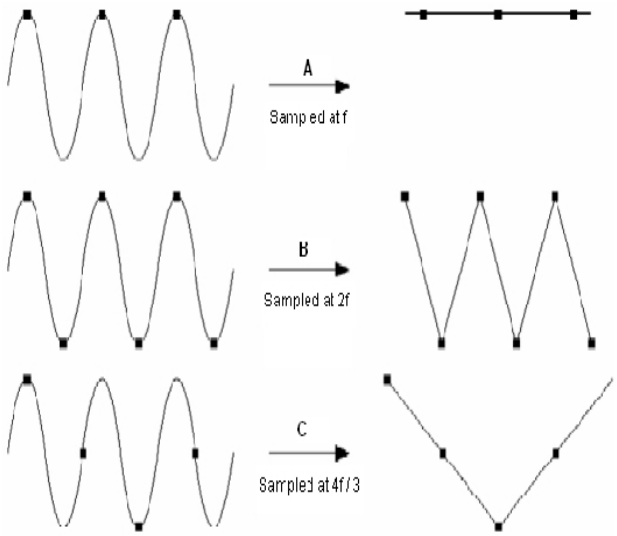


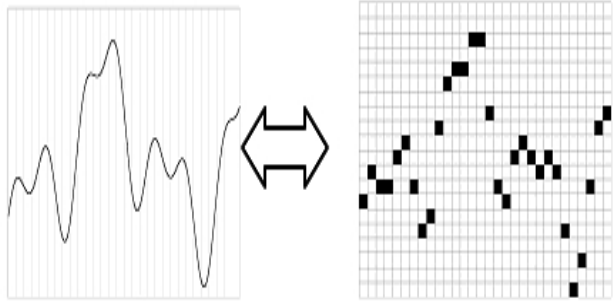
Βάλτε χρώμα στη ζωή σας !

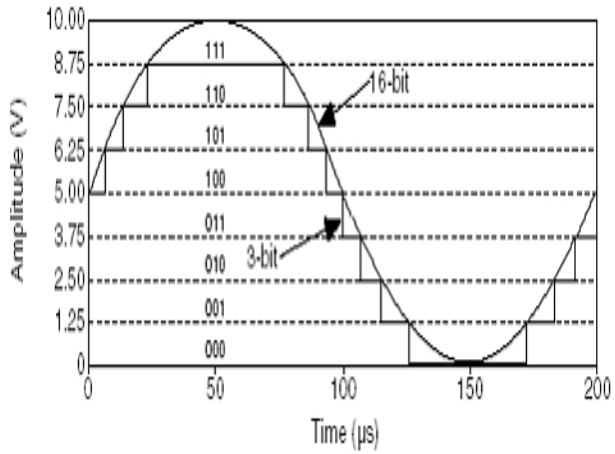
- Περισσότεροι τόνοι → περισσότερα δυαδικά ψηφία ανά pixel.
- Ολα τα χρώματα μπορούν να παραχθούν βάσει των τριών βασικών. Για μια έγχρωμη εικόνα αρκεί να καθορίσουμε την ένταση κάθε βασικής χρωματικής συνιστώσας !

Αναπαράσταση αναλογικού σήματος

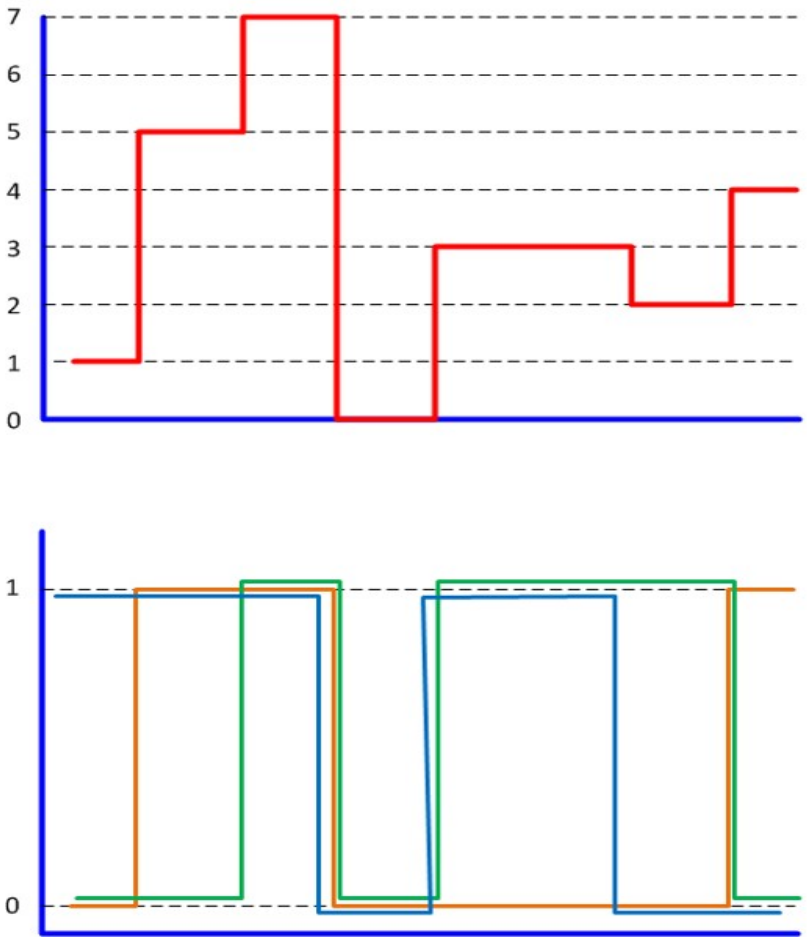
- Απαιτείται δειγματοληψία (sampling) και κβάντιση !
- Nyquist : Για σήμα συχνότητας f, η δειγματοληψία απαιτείται να γίνεται με συχνότητα $2 \times f$
- Με τη κβάντιση, διαιρούμε το συνεχές διάστημα τιμών σε διακριτά υποδιαστήματα και κωδικοποιούμε τα διαφορετικά διαστήματα στο δυαδικό.







Κωδικοποίση με δυαδικά σήματα



8 στάθμες => 3 δυαδικά σήματα
(x, y, z) για τη κωδικοποίηση.
Έστω η κωδικοποίηση :

0 -> 000
1 -> 001
2 -> 010
3 -> 011
4 -> 100
5 -> 101
6 -> 110
7 -> 111

EDC codes

- Τα ψηφιακά συστήματα είναι ευπαθή σε σφάλματα.
- Τα σφάλματα μπορεί να εκδηλώνονται με την αλλαγή στην τιμή ενός ή περισσοτέρων δυαδικών ψηφίων (bit flips).
- Προστατεύουμε τις πληροφορίες μας, με τους κώδικες EDC που πετυχαίνουν :
  - Εντοπισμό (detection) λαθών ή /και
  - Διόρθωση (correction) λαθών.
- Όλοι αυτοί οι κώδικες βασίζονται στην πρόσθεση στα αρχικά k ψηφία πληροφορίας (info), c επιπλέον ψηφίων ελέγχου (check).

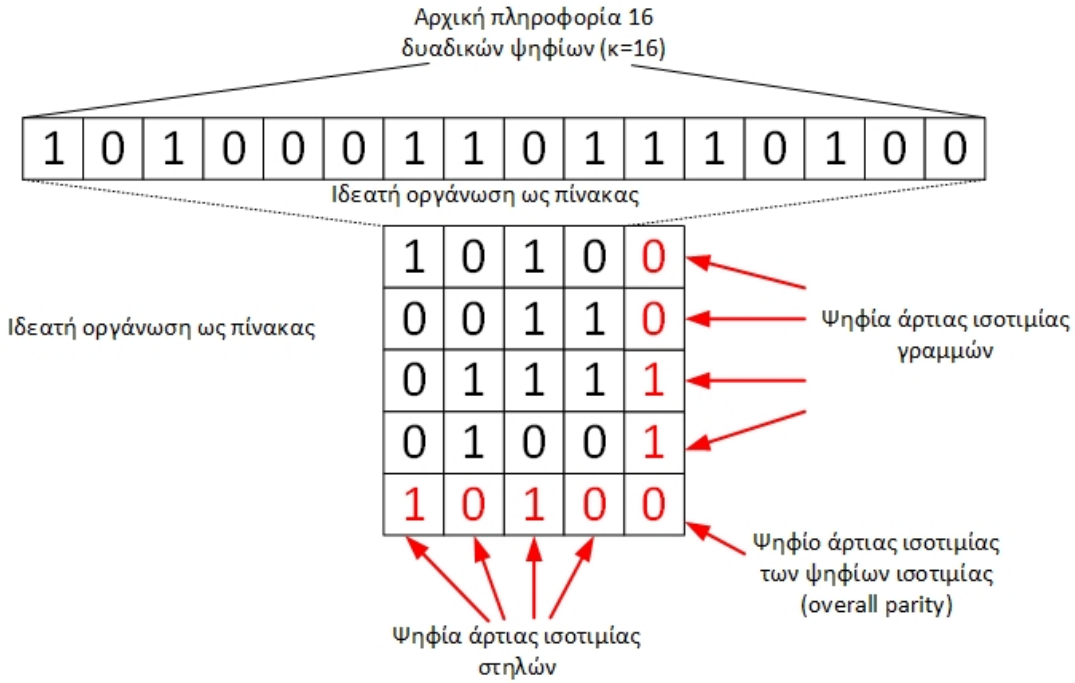
Κώδικας ισοτιμίας - Parity code - Ανίχνευση απλού λάθους

- Στην αρχική μου πληροφορία προσθέτω ένα επιπλέον ψηφίο (parity bit) ώστε η τελική λέξη να έχει είτε άρτιο, είτε περιττό αριθμό από 1.
- Αν επιλέξω άρτιο (περιττό) το ψηφίο που προσθέτω ονομάζεται ψηφίο άρτιας (περιττής) ισοτιμίας και ο κώδικας στον οποίο καταλήγω κώδικας άρτιας (περιττής) ισοτιμίας.
- Πριν χρησιμοποιηθούν τα ψηφία πληροφορίας η ισοτιμία της κωδικοποιημένης λέξης ελέγχεται:
  - Εάν είναι σωστή, θεωρούμε ότι δεν υπάρχει αλλοίωση.
  - Εάν είναι λανθασμένη, απορρίπτουμε αυτή την πληροφορία.
- Ο κώδικας ισοτιμίας, μας προσφέρει μόνο ανίχνευση περιττού αριθμού από bit flips.

Κώδικας ισοτιμίας στήλης - γραμμής

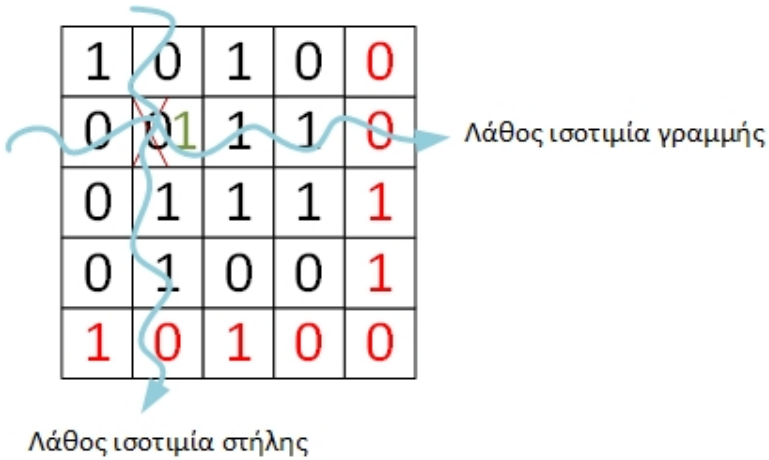
- Ο κώδικας ισοτιμίας δημιουργεί μια ομάδα ισοτιμίας (parity group) στην οποία ανήκει κάθε ψηφίο πληροφορίας.
- Αν θέλω να αποκτήσω ικανότητα διόρθωσης λάθους, θα πρέπει κάθε ψηφίο της πληροφορίας να ανήκει σε περισσότερα του ενός group.
- Η τομή των group με λανθασμένη ισοτιμία θα πρέπει να μου υποδεικνύουν μοναδικά το ψηφίο που έχει αντιστραφεί.
- Στον κώδικα ισοτιμίας στήλης - γραμμής, η πληροφορία οργανώνεται ιδεατά ως ένας πίνακας.
- Σε κάθε γραμμή και σε κάθε στήλη προστίθεται ένα ψηφίο ισοτιμίας δημιουργώντας οριζόντια και κάθετα groups ισοτιμίας.

Παράδειγμα κωδικοποίησης



- Για 16 ψηφία πληροφορίας προσθέσαμε 9 ψηφία ισοτιμίας και δημιουργήσαμε 9 groups ισοτιμίας.

Εντοπισμός & διόρθωση



- Το ψηφίο στη 2η γραμμή και στήλη άλλαξε τιμή λόγω σφάλματος.
- Αυτό επισημαίνεται από τη λανθασμένη ισοτιμία δύο group : της οριζόντιας για τη 2η γραμμή και της κάθετης της 2ης στήλης.
- Η τομή αυτών των group υποδεικνύει τη θέση του λάθους (εντοπισμός - location).
- Η διόρθωση (correction) του λάθους έγκειται στο να αλλάξουμε την τιμή του πάλι στην αρχική.
- Είναι άραγε αυτός ο αριθμός ψηφίων ελέγχου ο ελάχιστος για να αποκτήσουμε ικανότητα εντοπισμού του λάθους;

Απλός κώδικας Hamming

- Ο Αμερικανός μαθηματικός Richard Wesley Hamming (1915 – 1998) μεταξύ άλλων απέδειξε ότι για να αποκτήσω ικανότητα εντοπισμού λάθους σε αρχική πληροφορία k ψηφίων, πρέπει να προσθέσω τουλάχιστον c check bits έτσι ώστε $2^c \geq k + c + 1$.
- Για k=16, αρκούν μόλις 5 ψηφία έναντι 9 του κώδικα ισοτιμίας στήλης - γραμμής.
- Διαδικασία :
  - Δημιουργώ ένα μήνυμα $k + c$ ψηφίων.
  - Αριθμώ τις θέσεις των ψηφίων από 1 έως $k + c$.
  - Οι θέσεις που είναι δυνάμεις του 2 καταλαμβάνονται από τα c ψηφία ελέγχου και οι υπόλοιπες από τα ψηφία πληροφορίας του. Συμβολίζουμε με c_i το ψηφίο ελέγχου που τοποθετείται στη θέση i.
  - Γράφω τη θέση κάθε ψηφίου ως δυαδικό αριθμό. Τα 1 της δυαδικής γραφής μου αποκαλύπτουν σε ποια group ισοτιμίας θα ανήκει.
  - Π.χ. το ψηφίο στη θέση 7_{10}, επειδή $7_{10} = 000 \ldots 111$, θα ανήκει στα group ισοτιμίας των 3 τελευταίων ψηφίων ελέγχου, δηλαδή το c_1, το c_2 και το c_4.
  - Εναλλακτικά, μπορούμε να δούμε ότι το c_i είναι το ψηφίο ισοτιμίας που ελέγχει $i-$άδες ψηφίων που ξεκινούν από τη θέση του περιλαμβάνοντας την 1η, την 3η, την 5η, κοκ.

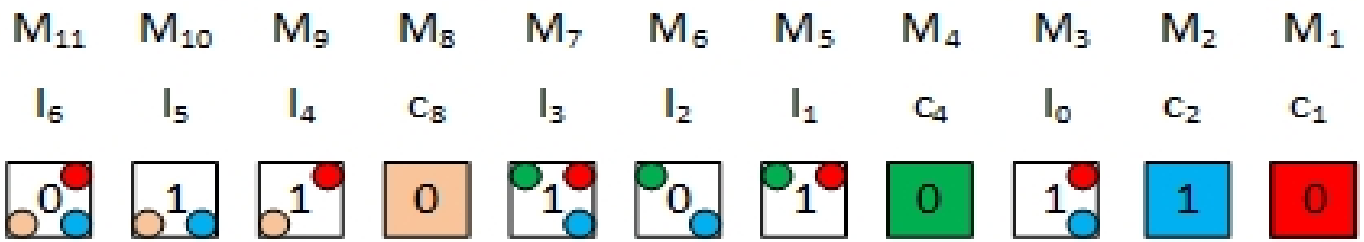
Κωδικοποίηση σε κώδικα Hamming (1/2)

- Ας υποθέσουμε $k = 7$, και ότι η πληροφορία μας είναι η $I_6 \ldots I_0 = 0111011$
- Λύνοντας την ανισότητα $2^c \geq k + c + 1$ βρίσκουμε ότι ο ελάχιστος c που την ικανοποιεί είναι $c = 4$.
- Συνεπώς κατασκευάζουμε το μήνυμα 11 ψηφίων $M_{11} \ldots M_1$:

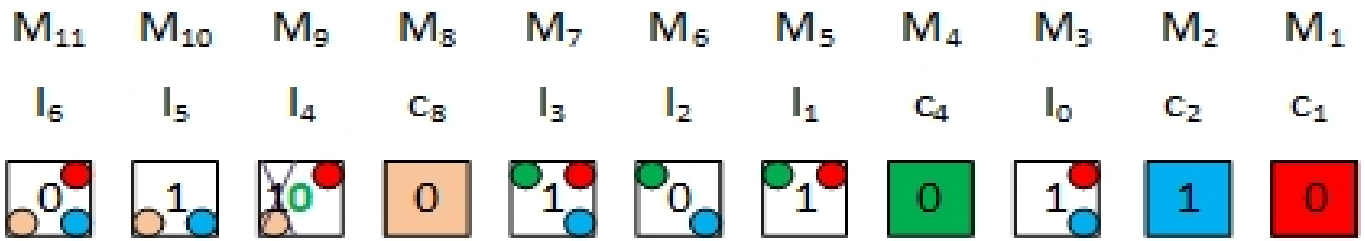
M_{11}	M_{10}	M_9	M_8	M_7	M_6	M_5	M_4	M_3	M_2	M_1
I_6	I_5	I_4	C_8	I_3	I_2	I_1	C_4	I_0	C_2	C_1
0	1	1		1	0	1		1		

Κωδικοποίηση σε κώδικα Hamming (2/2)

- Φτιάχνουμε τα groups ισοτιμίας ως εξής (δείτε και τον χρωματικό κώδικα):
  - $\{M_{11}, M_9, M_7, M_5, \mathrm{M}_3, \mathrm{M}_1\}$, δηλαδή απλά bits που ξεκινούν από το c_1, εναλλάξ (ένα συμπεριλαμβάνεται κι ένα όχι).
  - $\{M_{11}, M_{10}, M_7, M_6, \mathrm{M}_3, \mathrm{M}_2\}$, δηλαδή δυάδες bit που ξεκινούν από το c_2, εναλλάξ (μία δυάδα συμπεριλαμβάνεται και μία όχι).
  - $\{M_7, M_6, \mathrm{M}_5, \mathrm{M}_4\}$, δηλαδή τετράδες bit που ξεκινούν από το c_4, εναλλάξ.
  - $\{M_{11}, M_{10}, \mathrm{M}_9, \mathrm{M}_8\}$, δηλαδή οκτάδες bits που ξεκινούν από το c_8, εναλλάξ. Υποθέτουμε ότι η οκτάδα συμπληρώνεται με 0.
- Υπολογίζουμε τα c_i ως τα ψηφία άρτιας ισοτιμίας σε κάθε group.

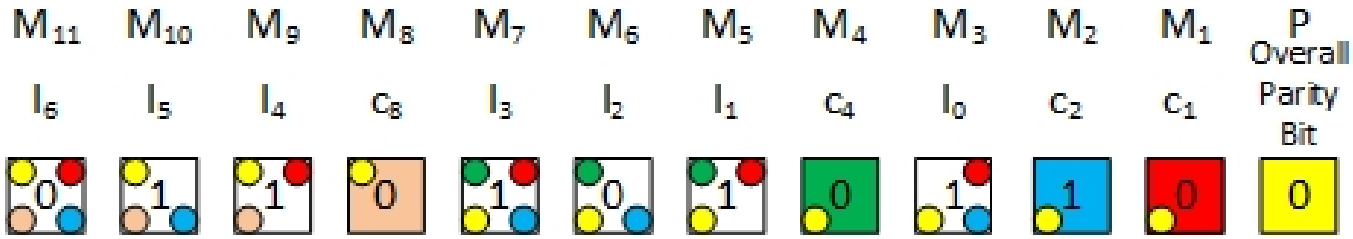


Εντοπισμός - διόρθωση



- Ας υποθέσουμε ότι λόγω σφάλματος το ψηφίο M_9 του μηνύματος αντιστρέφεται.
- Ελέγχουμε την ισοτιμία κάθε group και αθροίζουμε τα βάρη των c_i με λανθασμένη ισοτιμία, δημιουργώντας έτσι ένα σύνδρομο (syndrome).
- Σύνδρομο $0 \rightarrow$ δεν έχει γίνει απλό λάθος στο μήνυμά μας.
- Σύνδρομο $\neq 0 \rightarrow$ έχει γίνει απλό λάθος στη θέση του μηνύματος που υποδεικνύει το σύνδρομο, το οποίο και διορθώνουμε.
- Στο παράδειγμά μας, θα βρούμε λάθος στο c_8 και το c_1, άρα σύνδρομο 9 και θα πρέπει να αλλάξουμε την τιμή του M_9.
- Ο απλός κώδικας Hamming διορθώνει απλά λάθη. Αν χρησιμοποιηθεί σε περιβάλλον διπλών λαθών θα αποτύχει παταγωδώς.
- Για διπλά λάθη, χρησιμοποιούμε τον τροποποιημένο κώδικα Hamming.

Τροποποιημένος (modified) κώδικας Hamming



- Ο τροποποιημένος κώδικας Hamming μπορεί να διορθώσει απλό λάθος και να ανιχνεύσει διπλό.
- Κατασκευάζεται προσθέτοντας ένα ψηφίο ισοτιμίας στο κωδικοποιημένο μήνυμα του απλού κώδικα (overall parity).
- Για τον έλεγχο της ορθότητας του συνολικού μηνύματος, υπολογίζουμε αφενός τη συνολική ισοτιμία και αφετέρου το σύνδρομο χωρίς το overall parity και διακρίνουμε τις εξής περιπτώσεις:
  - Σωστή ισοτιμία και σύνδρομο $0 \rightarrow$ το μήνυμα είναι σωστό και μπορώ να εμπιστευθώ την πληροφορία του.
  - Σωστή ισοτιμία και σύνδρομο $\neq 0 \rightarrow$ έχουν συμβεί ζυγά και μεγαλύτερα του 0 λάθη και συνεπώς απορρίπτω την πληροφορία.
  - Λάθος ισοτιμία και σύνδρομο 0. Σε περιβάλλον απλών και διπλών λαθών, αυτή η περίπτωση δείχνει λάθος στο ψηφίο συνολικής ισοτιμίας.
  - Λάθος ισοτιμία και σύνδρομο $\neq 0 \rightarrow$ έχουν συμβεί περιττά και μεγαλύτερα του 0 λάθη. Σε περιβάλλον απλών και διπλών λαθών, θα διορθώσω το ψηφίο που υποδεικνύει το σύνδρομο.

Γενικά

- Θα ασχοληθούμε μόνο με τις βασικές πράξεις.
- Οι πιο σύνθετες πράξεις προκύπτουν ως σύνθεση των απλών.
- Η αριθμητική υπολογιστών είναι και θα συνεχίσει να είναι ενδιαφέρουσα.
- Το σύνολο των υπολογιστικών συστημάτων σήμερα χρησιμοποιούν για τους ακεραίους αναπαράσταση συμπληρώματος ως προς 2.
- Εξετάζουμε την αριθμητική συμπληρώματος ως προς 1 για ιστορικούς καθαρά λόγους.
- Πρόσθεση και αφαίρεση εξετάζονται σα μία πράξη, αφού $A - B = A + (-B)$.
- Σε μία πρόσθεση ή αφαίρεση τα τελούμενα όπως και το αποτέλεσμα έχουν πάντοτε το ίδιο μήκος.

Σημαίες Κατάστασης (flags / status bits)

- Πέραν του αποτελέσματος κάθε αριθμητική πράξη ”παράγει” και κάποια δυαδικά ψηφία σχετικά με το είδος του αποτελέσματος, γνωστά ως σημαίες κατάστασης (flags).
  - Σημαία κρατουμένου, Carry Flag - C: Τίθεται στο 1 αν και μόνο αν υπάρχει κρατούμενο εξόδου.
  - Σημαία αρνητικού αποτελέσματος, Negative Flag - N: Τίθεται στο 1 αν και μόνο αν το MSB του αποτελέσματος είναι 1.
  - Σημαία μηδενικού αποτελέσματος, Zero Flag - Z: Τίθεται στο 1 αν και μόνο αν το αποτέλεσμα είναι 0.
  - Σημαία υπερχείλισης, oVerflow Flag - V: Τίθεται στο 1 αν και μόνο αν το κρατούμενο προς τη πιο σημαντική θέση και το κρατούμενο εξόδου έχουν διαφορετικές τιμές.

Στο συμπλήρωμα ως προς 1

- End-around-carry - επανεισαγόμενο κρατούμενο : Το κρατούμενο εξόδου δε θα πρέπει να αγνοηθεί αλλά θα πρέπει να προστεθεί με 2η πρόσθεση στο πρώτο προσωρινό αποτέλεσμα.
- Βασίζεται στο γεγονός της μη $1 \leftrightarrow 1$ και επί συσχέτισης των ακεραίων με τις αναπαραστάσεις του κώδικα.

Αναπαραστάσεις	100	101	110	111 000	001	010	011
Ακέραιοι	-3	-2	-1	0	1	2	3

Παραδείγματα πρόσθεσης στο συμπλήρωμα ως προς 1

```
 01001011 ( +75)
 10001001 (-118) +
------------------
0 11010100 ( -43)
```

```
 11000101  ( -58)
 01110010 (+114) +
------------------
1 00110111
    |
    +---> 1   +
------------------
 00111000 (+56)
```

Υπερχείλιση

Πράξη που οδηγεί σε υπερχείλιση είναι πάντα λανθασμένη, ανεξάρτητα του επανεισαγόμενου κρατούμενου.

```
  01001011 ( +75)
  01000000 (+ 64)  +
-----------------------
0 10001011 ( -116)
```

```
  11000101  ( -58)
  10001101  (-114)  +
-----------------------
1 01010010
    |
    +-------> 1     +
-----------------------
  01010011 (+83) !!!
```

Προβλήματα του συμπληρώματος ως προς 1

- Δύο αναπαραστάσεις του 0 : ανάγκη διαχωρισμού τους.
- Δεύτερη πρόσθεση όταν το κρατούμενο εξόδου είναι 1.
- Και τα 2 πιο πάνω λύνονται με το συμπλήρωμα ως προς 2.

Προβλήματα του συμπληρώματος ως προς 1

- Δύο αναπαραστάσεις του 0 : ανάγκη διαχωρισμού τους.
- Δεύτερη πρόσθεση όταν το κρατούμενο εξόδου είναι 1.
- Και τα 2 πιο πάνω λύνονται με το συμπλήρωμα ως προς 2.
- Είναι η αριθμητική συμπληρώματος ως προς 1 άχρηστη ?

Προβλήματα του συμπληρώματος ως προς 1

- Δύο αναπαραστάσεις του 0 : ανάγκη διαχωρισμού τους.
- Δεύτερη πρόσθεση όταν το κρατούμενο εξόδου είναι 1.
- Και τα 2 πιο πάνω λύνονται με το συμπλήρωμα ως προς 2.
- Είναι η αριθμητική συμπληρώματος ως προς 1 άχρηστη ?
- Οχι. Είναι ισοδύναμη με αυτήν του υπολοίπου κατά $2^n - 1$ που χρησιμοποιείται ευρέως σε υπερυπολογιστές.

Στο συμπλήρωμα ως προς 2

- Απαλλασσόμαστε από το επανεισαγόμενο κρατούμενο

	00001010	(+10)	
	01010000	(+80)	+
0	01011010	(+90)	

	00000101	(+5)	
	11111110	(-2)	+
1	00000011	(+3)	

- Προφανώς δεν απαλλασσόμαστε από την υπερχείλιση

	00110010	(+50)	
	01010000	(+80)	+
0	10000010	(-126)	

	11010001	(-47)	
	10100110	(-90)	+
1	01110111	(+119)	

Μια κοντινή ματιά στην υπερχείλιση

- Δε μπορεί να προκύψει υπερχείλιση κατά τη πρόσθεση (αφαίρεση) ετεροσήμων (ομοσήμων).

Μια κοντινή ματιά στην υπερχείλιση

- Δε μπορεί να προκύψει υπερχείλιση κατά τη πρόσθεση (αφαίρεση) ετεροσήμων (ομοσήμων).
- Η υπερχείλιση εκδηλώνεται σαν αρνητικό αποτέλεσμα από πρόσθεση θετικών ή θετικό αποτέλεσμα από πρόσθεση αρνητικών.

Μια κοντινή ματιά στην υπερχείλιση

- Δε μπορεί να προκύψει υπερχείλιση κατά τη πρόσθεση (αφαίρεση) ετεροσήμων (ομοσήμων).
- Η υπερχείλιση εκδηλώνεται σαν αρνητικό αποτέλεσμα από πρόσθεση θετικών ή θετικό αποτέλεσμα από πρόσθεση αρνητικών.
- α_7, β_7, κ_7, σ_7, κ, τα MSB των α και β, το κρατούμενο εισόδου στη πιο σημαντική βαθμίδα, το πιο σημαντικό δυαδικό ψηφίο αθροίσματος και το κρατούμενο εξόδου αντίστοιχα.

Μια κοντινή ματιά στην υπερχείλιση

- Δε μπορεί να προκύψει υπερχείλιση κατά τη πρόσθεση (αφαίρεση) ετεροσήμων (ομοσήμων).
- Η υπερχείλιση εκδηλώνεται σαν αρνητικό αποτέλεσμα από πρόσθεση θετικών ή θετικό αποτέλεσμα από πρόσθεση αρνητικών.
- α_7, β_7, κ_7, σ_7, κ, τα MSB των α και β, το κρατούμενο εισόδου στη πιο σημαντική βαθμίδα, το πιο σημαντικό δυαδικό ψηφίο αθροίσματος και το κρατούμενο εξόδου αντίστοιχα.
- Υπερχείλιση όταν :

Μια κοντινή ματιά στην υπερχείλιση

- Δε μπορεί να προκύψει υπερχείλιση κατά τη πρόσθεση (αφαίρεση) ετεροσήμων (ομοσήμων).
- Η υπερχείλιση εκδηλώνεται σαν αρνητικό αποτέλεσμα από πρόσθεση θετικών ή θετικό αποτέλεσμα από πρόσθεση αρνητικών.
- α_7, β_7, κ_7, σ_7, κ, τα MSB των α και β, το κρατούμενο εισόδου στη πιο σημαντική βαθμίδα, το πιο σημαντικό δυαδικό ψηφίο αθροίσματος και το κρατούμενο εξόδου αντίστοιχα.
- Υπερχείλιση όταν :
  - $\alpha_7 = 0, \beta_7 = 0, \sigma_7 = 1 \Rightarrow \kappa_7 = 1, \kappa = 0.$

Μια κοντινή ματιά στην υπερχείλιση

- Δε μπορεί να προκύψει υπερχείλιση κατά τη πρόσθεση (αφαίρεση) ετεροσήμων (ομοσήμων).
- Η υπερχείλιση εκδηλώνεται σαν αρνητικό αποτέλεσμα από πρόσθεση θετικών ή θετικό αποτέλεσμα από πρόσθεση αρνητικών.
- α_7, β_7, κ_7, σ_7, κ, τα MSB των α και β, το κρατούμενο εισόδου στη πιο σημαντική βαθμίδα, το πιο σημαντικό δυαδικό ψηφίο αθροίσματος και το κρατούμενο εξόδου αντίστοιχα.
- Υπερχείλιση όταν :
  - $\alpha_7 = 0, \beta_7 = 0, \sigma_7 = 1 \Rightarrow \kappa_7 = 1, \kappa = 0.$
  - $\alpha_7 = 1, \beta_7 = 1, \sigma_7 = 0 \Rightarrow \kappa_7 = 0, \kappa = 1.$

Μια κοντινή ματιά στην υπερχείλιση

- Δε μπορεί να προκύψει υπερχείλιση κατά τη πρόσθεση (αφαίρεση) ετεροσήμων (ομοσήμων).
- Η υπερχείλιση εκδηλώνεται σαν αρνητικό αποτέλεσμα από πρόσθεση θετικών ή θετικό αποτέλεσμα από πρόσθεση αρνητικών.
- α_7, β_7, κ_7, σ_7, κ, τα MSB των α και β, το κρατούμενο εισόδου στη πιο σημαντική βαθμίδα, το πιο σημαντικό δυαδικό ψηφίο αθροίσματος και το κρατούμενο εξόδου αντίστοιχα.
- Υπερχείλιση όταν :
  - $\alpha_7 = 0, \beta_7 = 0, \sigma_7 = 1 \Rightarrow \kappa_7 = 1, \kappa = 0.$
  - $\alpha_7 = 1, \beta_7 = 1, \sigma_7 = 0 \Rightarrow \kappa_7 = 0, \kappa = 1.$
- Αρα υπερχείλιση όταν $\kappa_7 \neq \kappa \Leftrightarrow \kappa_7 \oplus \kappa = 1.$

Πρόσθεση - Αφαίρεση σε Πρόσημο - Μέτρο

- Αφαίρεση = Πρόσθεση με εναλλαγή του προσήμου του αφαιρέτη. Εξετάζουμε συνεπώς μόνο τη πρόσθεση.
- Εξετάζουμε πρόσημο αριθμού :
  - Ομόσημοι ⇒ πρόσημο αποτελέσματος το κοινό πρόσημο. Μέτρο, το άθροισμα των μέτρων.
  - Ετερόσημοι ⇒ σύγκριση των μέτρων. Πρόσημο αποτελέσματος το πρόσημο του μεγαλύτερου μέτρου. Μέτρο η διαφορά των μέτρων. Αφαίρεση σύμφωνα με τον πίνακα.

Είσοδοι			Εξοδοι	
Ψηφίο Αφαιρετέου	Ψηφίο Αφαιρέτη	Δανεικό από δεξιά	Ψηφίο υπολοίπου	Δανεικό προς δεξιά
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Πρόσθεση - Αφαίρεση σε Πλεονασμό κατά K

- Πρόσθεση =
  1. Δυαδική πρόσθεση. Το αποτέλεσμα είναι πολωμένο κατά 2K και απαιτεί την αφαίρεση της υπερβάλλουσας πόλωσης. Το ενδιάμεσο αποτέλεσμα χρειάζεται περισσότερα δυαδικά ψηφία αναπαράστασης, ή

Πρόσθεση - Αφαίρεση σε Πλεονασμό κατά K

- Πρόσθεση =
  1. Δυαδική πρόσθεση. Το αποτέλεσμα είναι πολωμένο κατά 2K και απαιτεί την αφαίρεση της υπερβάλλουσας πόλωσης. Το ενδιάμεσο αποτέλεσμα χρειάζεται περισσότερα δυαδικά ψηφία αναπαράστασης, ή
  2. Συγκρίνουμε κάθε έντελο με το K. Αν μεγαλύτερο ή ίσο $\Rightarrow$ θετικός, αλλιώς αρνητικός. Αφαιρούμε τη πόλωση (πόλωση - τελούμενο, αν αρνητικός) και βρίσκουμε το μέτρο. Εχουμε πλέον τα έντελα σε μορφή πρόσημο-μέτρο και ακολουθεί πρόσθεση για εκείνη την αναπαράσταση.

Πρόσθεση - Αφαίρεση σε Πλεονασμό κατά K

- Πρόσθεση =
  1. Δυαδική πρόσθεση. Το αποτέλεσμα είναι πολωμένο κατά 2K και απαιτεί την αφαίρεση της υπερβάλλουσας πόλωσης. Το ενδιάμεσο αποτέλεσμα χρειάζεται περισσότερα δυαδικά ψηφία αναπαράστασης, ή
  2. Συγκρίνουμε κάθε έντελο με το K. Αν μεγαλύτερο ή ίσο ⇒ θετικός, αλλιώς αρνητικός. Αφαιρούμε τη πόλωση (πόλωση - τελούμενο, αν αρνητικός) και βρίσκουμε το μέτρο. Εχουμε πλέον τα έντελα σε μορφή πρόσημο-μέτρο και ακολουθεί πρόσθεση για εκείνη την αναπαράσταση.
- Αφαίρεση =
  1. Θεωρούμε τους αριθμούς σα να είναι θετικοί σε παράσταση πρόσημο-μέτρο. Τους αφαιρούμε. Το αποτέλεσμα είναι σωστό, αλλά μη πολωμένο. Προσθέτουμε συνεπώς τη πόλωση, ή

Πρόσθεση - Αφαίρεση σε Πλεονασμό κατά K

- Πρόσθεση =
  1. Δυαδική πρόσθεση. Το αποτέλεσμα είναι πολωμένο κατά 2K και απαιτεί την αφαίρεση της υπερβάλλουσας πόλωσης. Το ενδιάμεσο αποτέλεσμα χρειάζεται περισσότερα δυαδικά ψηφία αναπαράστασης, ή
  2. Συγκρίνουμε κάθε έντελο με το K. Αν μεγαλύτερο ή ίσο ⇒ θετικός, αλλιώς αρνητικός. Αφαιρούμε τη πόλωση (πόλωση - τελούμενο, αν αρνητικός) και βρίσκουμε το μέτρο. Εχουμε πλέον τα έντελα σε μορφή πρόσημο-μέτρο και ακολουθεί πρόσθεση για εκείνη την αναπαράσταση.
- Αφαίρεση =
  1. Θεωρούμε τους αριθμούς σα να είναι θετικοί σε παράσταση πρόσημο-μέτρο. Τους αφαιρούμε. Το αποτέλεσμα είναι σωστό, αλλά μη πολωμένο. Προσθέτουμε συνεπώς τη πόλωση, ή
  2. Μετατρέπουμε τους αριθμούς σε πρόσημο μέτρο και ακολουθούμε πρόσθεση σε πρόσημο-μέτρο.

Αριθμητική των Ολισθήσεων

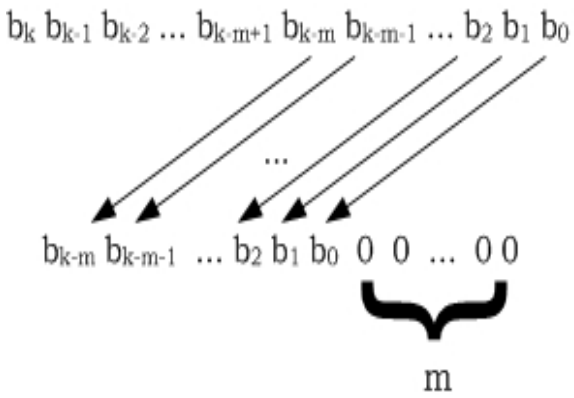
- Μπορούμε να ορίσουμε μοναδιαίους τελεστές που υλοποιούνται με λογικές πράξεις ολίσθησης των δυαδικών ψηφίων μιας ψηφιολέξης. Οι πράξεις αυτές έχουν αριθμητικές ιδιότητες.

Αριθμητική των Ολισθήσεων

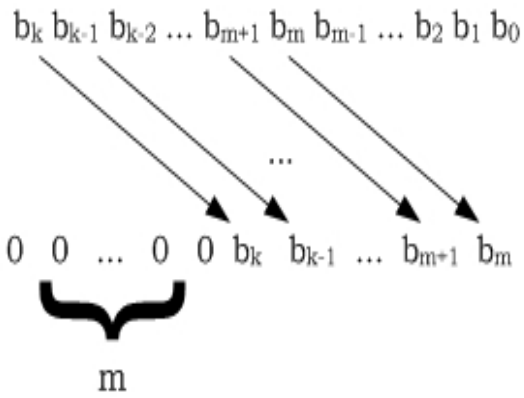
- Μπορούμε να ορίσουμε μοναδιαίους τελεστές που υλοποιούνται με λογικές πράξεις ολίσθησης των δυαδικών ψηφίων μιας ψηφιολέξης. Οι πράξεις αυτές έχουν αριθμητικές ιδιότητες.
- Λογική ολίσθηση προς τα αριστερά ή τα δεξιά. Ολίσθηση προς τα αριστερά ή τα δεξιά κατά κ θέσεις αντιστοιχεί σε πολλαπλασιασμό ή ακέραιη διαίρεση με 2^κ αντίστοιχα.
- Στην αριθμητική ολίσθηση, οι κενές θέσεις συμπληρώνονται με το αριστερότερο ψηφίο της αρχικής μας ψηφιολέξης. Επεκτείνεται η διαίρεση με 2^κ και σε αριθμούς σε συμπλήρωμα ως προς 2.
- Κυκλική ολίσθηση προς τα δεξιά ή τα αριστερά.
- Ολες αυτές οι διαδικασίες, υλοποιούνται με ένα λογικό κύκλωμα που ονομάζεται ολισθητής.

Σχήματα Ολισθήσεων

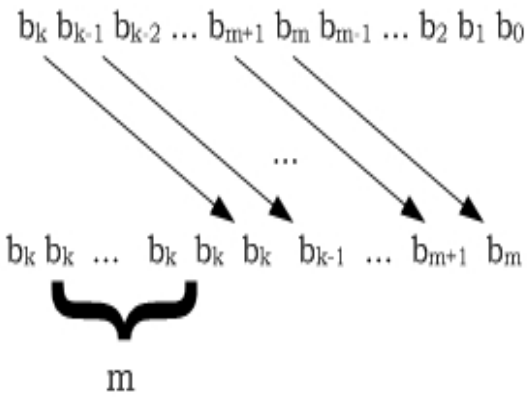
m θέσεων Λογική Ολίσθηση Αριστερά



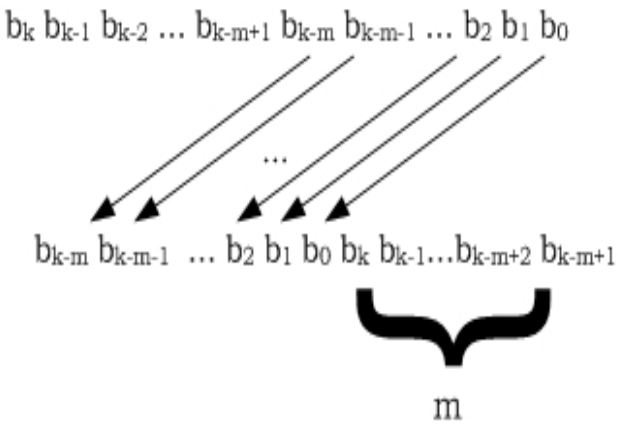
m θέσεων Λογική Ολίσθηση Δεξιά



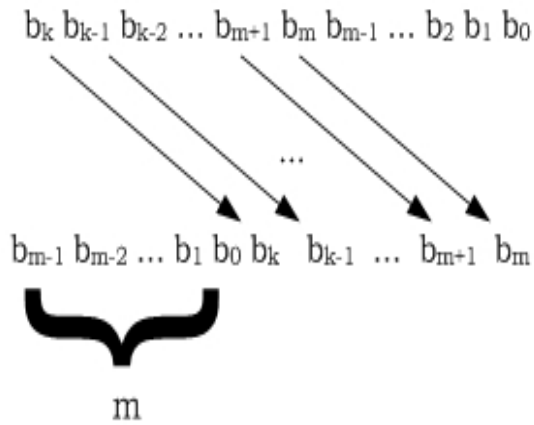
m θέσεων Αριθμητική Ολίσθηση Δεξιά



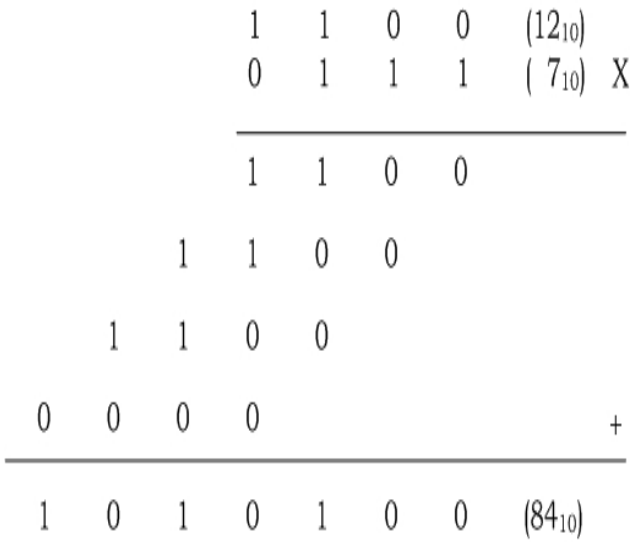
m θέσεων Κυκλική Ολίσθηση Αριστερά



m θέσεων Κυκλική Ολίσθηση Δεξιά



Πολλαπλασιασμός μη προσημασμένων αριθμών



- Κάθε μερικό γινόμενο είναι είτε 0, είτε ίσο με τον πολλαπλασιαστέο ολισθημένο αριστερά τόσες φορές όση η θέση του ψηφίου του πολλαπλασιαστή.
- Ο πολλαπλασιασμός είναι σημαντικά πιο πολύπλοκη πράξη από τη πρόσθεση.
- Το μήκος του αποτελέσματος απαιτεί $\kappa + \lambda$ δυαδικά ψηφία, όπου κ και λ τα μήκη πολλαπλασιαστέου και πολλαπλασιαστή.

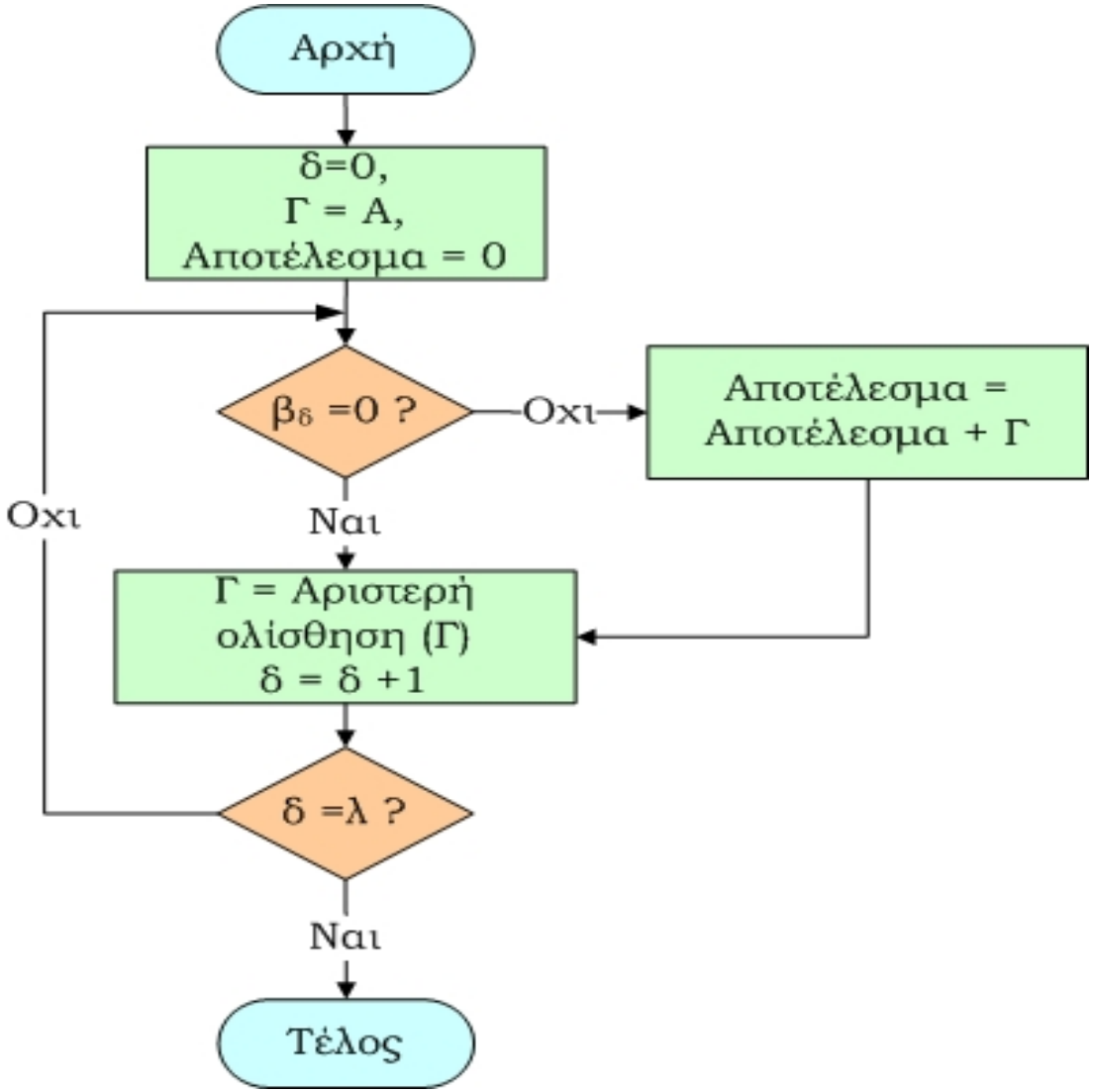
Αλγόριθμος μόνο με προσθέσεις & ολισθήσεις

Διαδικασία πολλαπλασιασμού του $A = \alpha_{\kappa-1}\alpha_{\kappa-2}\ldots\alpha_0$ με το $B = \beta_{\lambda-1}\beta_{\lambda-2}\ldots\beta_0$.

Αλγόριθμος : Η περιγραφή μιας διαδικασίας με απλά βήματα.

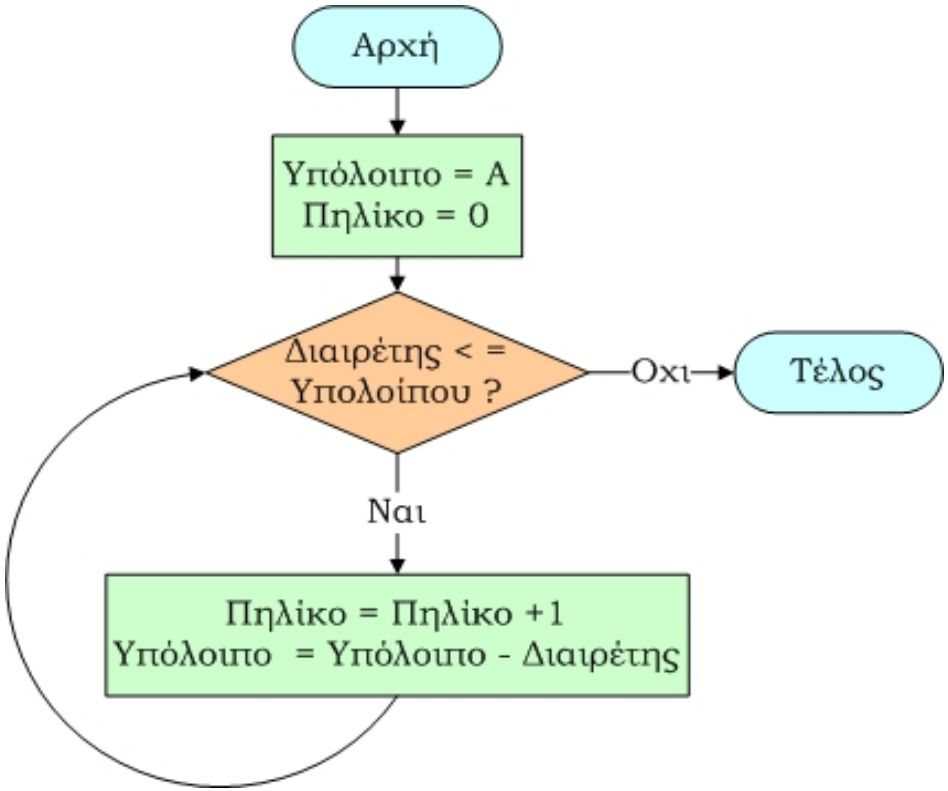
1. Θέσε $\delta = 0$ και $\Gamma = A$ και Αποτέλεσμα = 0.
2. Εξέτασε το δυαδικό ψηφίο β_δ.
3. Εάν είναι 0 πήγαινε στο βήμα 5.
4. Πρόσθεσε το Γ στο Αποτέλεσμα.
5. Θέσε $\Gamma =$ Αριστερή Ολίσθηση κατά μια θέση του Γ.
6. Αύξησε κατά 1 το δ.
7. Είναι $\delta = \lambda$; Αν ναι τότε ο πολλαπλασιασμός ολοκληρώθηκε. Αν όχι πήγαινε στο βήμα 2.

Περιγραφή ενός αλγορίθμου με διάγραμμα ροής



Αλγόριθμος διαδοχικών επιτυχών αφαιρέσεων

Αν ο διαιρετέος μας έχει μήκος $\kappa + \lambda$ δυαδικά ψηφία και ο διαιρέτης κ, το πηλίκο θα έχει λ και το υπόλοιπο το πολύ κ δυαδικά ψηφία.



Οταν το χαρτί και το μολύβι δε δουλεύει !

- Αν προσπαθήσουμε να ακολουθήσουμε τη κλασσική μέθοδο και για προσημασμένους, τότε θα διαπιστώναμε ότι δε δουλεύει:

			1	1	0	0	(-4_{10})
			0	1	0	1	$(+5_{10})$ X
			1	1	0	0	
		0	0	0	0		
	1	1	0	0			
0	0	0	0				+
0	1	1	1	1	0	0	$(+60_{10})$

Οταν το χαρτί και το μολύβι δε δουλεύει !

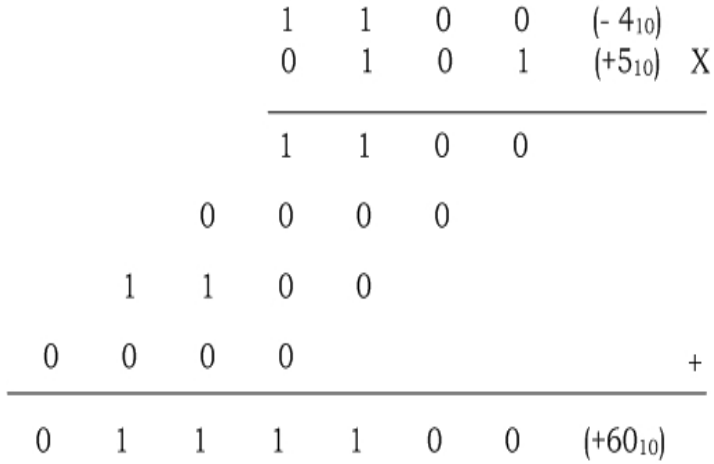
- Αν προσπαθήσουμε να ακολουθήσουμε τη κλασσική μέθοδο και για προσημασμένους, τότε θα διαπιστώναμε ότι δε δουλεύει:

			1	1	0	0	(-4_{10})
			0	1	0	1	$(+5_{10})$ X
			1	1	0	0	
		0	0	0	0		
	1	1	0	0			
0	0	0	0				+
0	1	1	1	1	0	0	$(+60_{10})$

- Το πρόβλημα δημιουργείται από το ότι το 1ο και το 3ο μερικό γινόμενο θεωρούνται θετικοί κατά τη πρόσθεσή τους.

Οταν το χαρτί και το μολύβι δε δουλεύει !

- Αν προσπαθήσουμε να ακολουθήσουμε τη κλασσική μέθοδο και για προσημασμένους, τότε θα διαπιστώναμε ότι δε δουλεύει:



- Το πρόβλημα δημιουργείται από το ότι το 1ο και το 3ο μερικό γινόμενο θεωρούνται θετικοί κατά τη πρόσθεσή τους.
- Λύση: Επέκταση κάθε εντέλου με τη διαδικασία επέκτασης προσήμου εξ' αρχής στην ακρίβεια του τελικού αποτελέσματος.

Οταν το χαρτί και το μολύβι δε δουλεύει !

- Αν προσπαθήσουμε να ακολουθήσουμε τη κλασσική μέθοδο και για προσημασμένους, τότε θα διαπιστώναμε ότι δε δουλεύει:

			1	1	0	0	(-4_{10})
			0	1	0	1	$(+5_{10})$ X
			1	1	0	0	
		0	0	0	0		
	1	1	0	0			
0	0	0	0				+
0	1	1	1	1	0	0	$(+60_{10})$

- Το πρόβλημα δημιουργείται από το ότι το 1ο και το 3ο μερικό γινόμενο θεωρούνται θετικοί κατά τη πρόσθεσή τους.
- Λύση: Επέκταση κάθε εντέλου με τη διαδικασία επέκτασης προσήμου εξ' αρχής στην ακρίβεια του τελικού αποτελέσματος.
- Για έντελα κ και λ ψηφίων, το αποτέλεσμα χρειάζεται κ+λ ψηφία.

Παράδειγμα σωστού πολλαπλασιασμού

1	1	1	1	1	1	0	0	(-4_{10})
0	0	0	0	0	1	0	1	$(+5_{10})$ X
1	1	1	1	1	1	0	0	
0	0	0	0	0	0	0		
1	1	1	1	0	0			
0	0	0	0	0				
0	0	0	0					
0	0	0						
0	0							
0								+
1	1	1	0	1	1	0	0	(-20_{10})

- Νέο πρόβλημα : Οταν ο πολλαπλασιαστής είναι μικρός αρνητικός αριθμός, η επέκταση προσήμου προκαλεί πάρα πολλά μη μηδενικά μερικά γινόμενα.
- Ο πολλαπλασιασμός αποκτά πολύ μεγάλη πολυπλοκότητα.

Τα βασικά μαθηματικά

- Εστω Α ο πολλαπλασιαστέος, Β ο πολλαπλασιαστής.

Τα βασικά μαθηματικά

- Εστω Α ο πολλαπλασιαστέος, Β ο πολλαπλασιαστής.
- $\beta_\kappa \beta_{\kappa-1} \ldots \beta_\lambda$ μια σειρά διαδοχικών ψηφίων του Β που είναι 1.

Τα βασικά μαθηματικά

- Εστω A ο πολλαπλασιαστέος, B ο πολλαπλασιαστής.
- $\beta_\kappa \beta_{\kappa-1} \ldots \beta_\lambda$ μια σειρά διαδοχικών ψηφίων του B που είναι 1.
- Θα μας δώσουν τα μερικά γινόμενα $A \times 2^\kappa, A \times 2^{\kappa-1}, \ldots, A \times 2^\lambda$.

Τα βασικά μαθηματικά

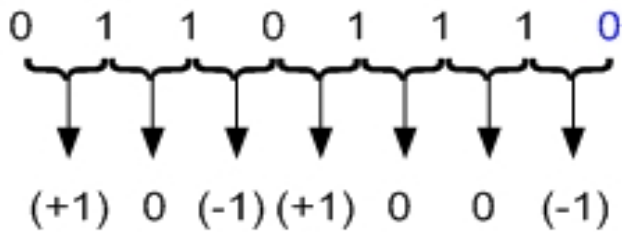
- Εστω A ο πολλαπλασιαστέος, B ο πολλαπλασιαστής.
- $\beta_\kappa \beta_{\kappa-1} \ldots \beta_\lambda$ μια σειρά διαδοχικών ψηφίων του B που είναι 1.
- Θα μας δώσουν τα μερικά γινόμενα $A \times 2^\kappa, A \times 2^{\kappa-1}, \ldots, A \times 2^\lambda$.
- Η πρόσθεση αυτών μας δίνει : $A \times 2^\lambda \times (2^{\kappa-\lambda} + 2^{\kappa-\lambda-1} + \ldots + 2^0) = A \times 2^\lambda \times (2^{\kappa-\lambda+1} - 1) = A \times (2^{\kappa+1} - 2^\lambda)$

Τα βασικά μαθηματικά

- Εστω Α ο πολλαπλασιαστέος, Β ο πολλαπλασιαστής.
- $\beta_\kappa \beta_{\kappa-1} \ldots \beta_\lambda$ μια σειρά διαδοχικών ψηφίων του Β που είναι 1.
- Θα μας δώσουν τα μερικά γινόμενα $A \times 2^\kappa, A \times 2^{\kappa-1}, \ldots, A \times 2^\lambda$.
- Η πρόσθεση αυτών μας δίνει : $A \times 2^\lambda \times (2^{\kappa-\lambda} + 2^{\kappa-\lambda-1} + \ldots + 2^0) = A \times 2^\lambda \times (2^{\kappa-\lambda+1} - 1) = A \times (2^{\kappa+1} - 2^\lambda)$
- Ο υπολογισμός του $A \times (2^{\kappa+1} - 2^\lambda)$ απαιτεί την αφαίρεση 2 μερικών γινομένων δηλαδή πολυπλοκότητα αντίστοιχη μιας μόνο πρόσθεσης !

Αρχικός αλγόριθμος Booth

- Σκοπός είναι να μετατρέψουμε τον πολλαπλασιαστή σε μια μορφή με ελάχιστα ψηφία διαφορετικά του 0.
- Στη νέα μορφή, κάθε ψηφίο του πολλαπλασιαστή μπορεί να είναι 0, +1 ή -1.
- Στο $\beta_\kappa \beta_{\kappa-1} \ldots \beta_1 \beta_0$ προσθέτουμε δεξιά ένα 0.
- Σαρώνουμε τον πολλαπλασιαστή από δεξιά προς αριστερά.
  - Τρέχων 1 και προηγούμενο 0 $\Rightarrow$ κωδικοποιημένο : -1
  - Τρέχων 0 και προηγούμενο 1 $\Rightarrow$ κωδικοποιημένο : +1
  - Για κάθε άλλο ζεύγος $\Rightarrow$ κωδικοποιημένο : 0



- 4 μη μηδενικά μερικά γινόμενα έναντι 5 αρχικών.
- Μείωση κατά 20% του αρχικού χρόνου εκτέλεσης.

Προβλήματα του αρχικού αλγορίθμου

- Ο αρχικός αλγόριθμος είναι εξαιρετικά χρήσιμος όταν ο πολλαπλασιαστής έχει μακριές αλυσίδες από συνεχόμενους 1.
- Δυστυχώς στις υπόλοιπες περιπτώσεις, ο αλγόριθμος αποτυγχάνει δραματικά.
- Για $B = 21_{10} = 010101_2$ θα είχαμε 3 μη μηδενικά μερικά γινόμενα.
- Αν τον κωδικοποιήσουμε παίρνει τη μορφή $(+1)(-1)(+1)(-1)(+1)(-1)$. Δημιουργούνται δηλαδή 6 μη μηδενικά μερικά γινόμενα.
- Δυστυχώς, η εφαρμογή του αλγορίθμου, αύξησε το χρόνο του πολλαπλασιασμού κατά 100%.

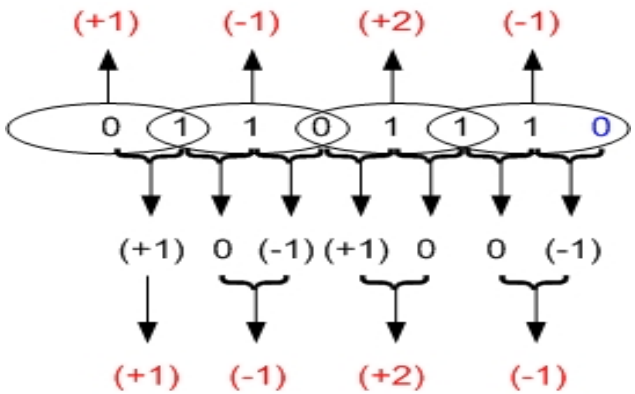
Τροποποιημένος αλγόριθμος Booth

- Στον τροποποιημένο αλγόριθμο δυνατά ψηφία είναι τα 0, (+1), (+2), (-1) και (-2).
- Η τροποποιημένη κωδικοποίηση μπορεί να προκύψει είτε εξετάζοντας ζεύγη ψηφίων της πρώτης ή ισοδύναμα επικαλυπτόμενες τριάδες της αρχικής μορφής του πολλαπλασιαστή (Πίνακας)
- Κάθε ψηφίο στη τροποποιημένη κωδικοποίηση έχει βάρος $2^{2\kappa}$.

Ζεύγος Ψηφίων 1ης Κωδικοποίησης	Επικαλυπτόμενη Τριάδα Πολλαπλασιαστή	Τελική Κωδικοποίηση
0 0	000 / 111	0
0 (+1)	001	(+1)
0 (-1)	110	(-1)
(+1) 0	011	(+2)
(+1)(-1)	010	(+1)
(-1) 0	100	(-2)
(-1) (+1)	101	(-1)

Παράδειγμα με τροποποιημένο αλγόριθμο Booth

- Πολλαπλασιαστής = $55_{10} = 0110111_{2s}$.
- 5 μη μηδενικά μερικά γινόμενα $\Rightarrow$ 4 προσθέσεις.
- Αρχική και τελική κωδικοποίηση κατά Booth σύμφωνα με το σχήμα



- Πολλαπλασιαστής = $(+1) \times 2^{2\times 3} + (-1) \times 2^{2\times 2} + (+2) \times 2^{2\times 1} + (-1) \times 2^{2\times 0} = (+1) \times 2^6 + (-1) \times 2^4 + (+2) \times 2^2 + (-1) \times 2^0 = 2^6 - 2^4 + 2^3 - 2^0 = 64 - 16 + 8 - 1 = 55_{10}$.
- $A \times B = A \times [2^6 - 2^4 + 2^3 - 2^0] = A000000 + (-A0000) + A000 + (-A)$

Κινητή Υποδιαστολή: Πρόσθεση & Αφαίρεση

- Για τη πρόσθεση / αφαίρεση ακολουθούνται τα εξής βήματα :
  1. Σύγκριση των εκθετών και ενδεχόμενα των σημαντικών μερών και εντοπισμός του μεγαλύτερου.
  2. Εκθέτης και πρόσημο αποτελέσματος = Εκθέτης, πρόσημο του μεγαλύτερου.
  3. Σε περίπτωση διαφορετικών εκθετών, ολίσθηση του μικρότερου δεξιά έως να εξισωθούν οι εκθέτες.
  4. Σημαντικό μέρος = Πρόσθεση ή αφαίρεση των σημαντικών μερών αφού πρώτα μετατραπούν σε συμπλήρωμα ως προς 2.
  5. Κανονικοποίηση του αποτελέσματος.

Κινητή Υποδιαστολή: Πολλαπλασιασμός & Διαίρεση

- Για τον/τη πολλαπλασιαμό/διαίρεση ακολουθούνται τα εξής βήματα :
  1. Πρόσημο αποτελέσματος = το $\oplus$ των προσήμων.
  2. Προσωρινός εκθέτης από πρόσθεση / αφαίρεση των εκθετών των εντέλων.
  3. Προσωρινό σημαντικό μέρος από πολλαπλασιασμό / διαίρεση των σημαντικών μερών των εντέλων.
  4. Κανονικοποίηση.

Πολύπλοκες πράξεις

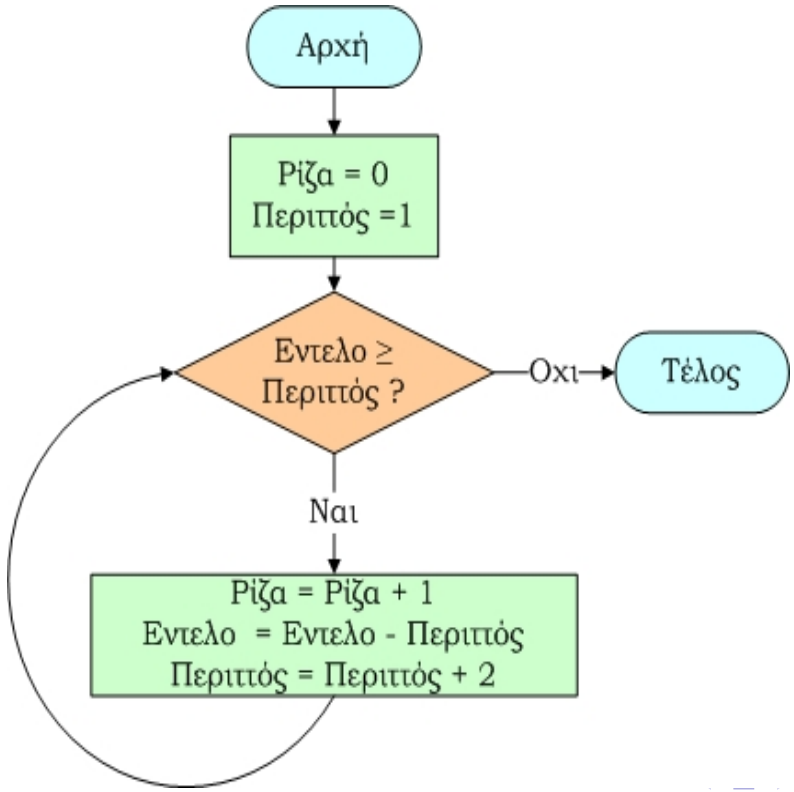
- Στους υπολογιστές μας υπάρχουν μόνο μονάδες άθροισης, πολλαπλασιασμού και ολισθήσεων.

Πολύπλοκες πράξεις

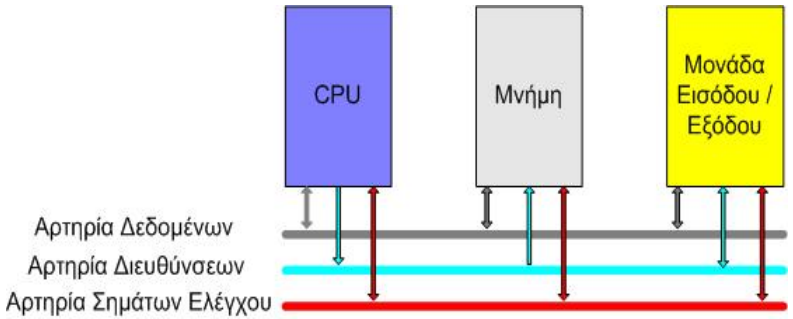
- Στους υπολογιστές μας υπάρχουν μόνο μονάδες άθροισης, πολλαπλασιασμού και ολισθήσεων.
- Ολες οι υπόλοιπες σύνθετες πράξεις εκτελούνται σα σύνθεση αυτών.

Πολύπλοκες πράξεις

- Στους υπολογιστές μας υπάρχουν μόνο μονάδες άθροισης, πολλαπλασιασμού και ολισθήσεων.
- Ολες οι υπόλοιπες σύνθετες πράξεις εκτελούνται σα σύνθεση αυτών.
- Το ακέραιο μέρος της ρίζας φυσικού, προσεγγίζεται από το flowchart :

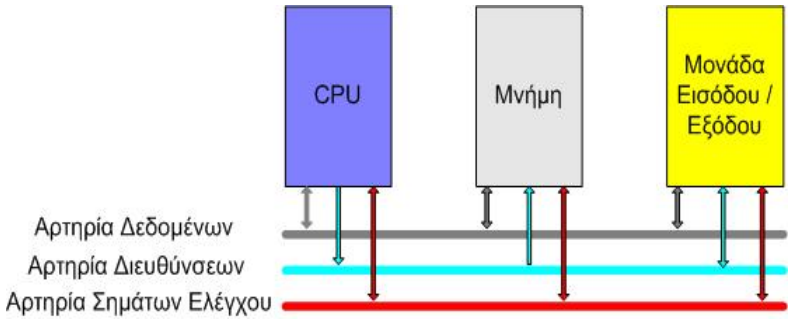


Η μεγάλη εικόνα ...



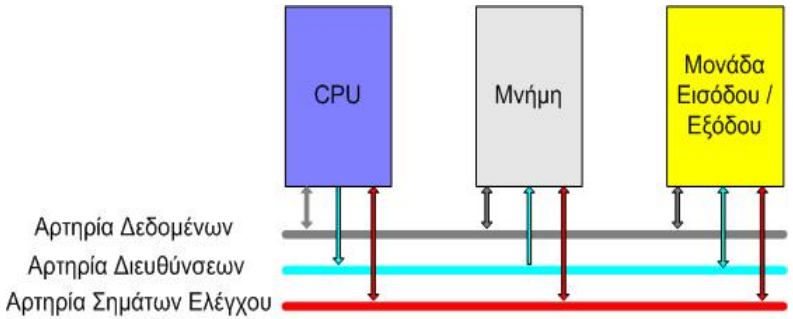
- Μάθαμε πως αναπαρίστανται οι πληροφορίες μας στον υπολογιστή.

Η μεγάλη εικόνα ...



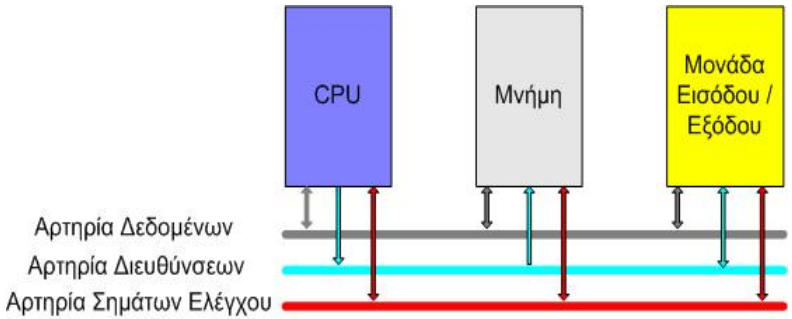
- Μάθαμε πως αναπαρίστανται οι πληροφορίες μας στον υπολογιστή.
- Είδαμε τρόπους με τους οποίους μπορούμε να κάνουμε αριθμητικές συναρτήσεις.

Η μεγάλη εικόνα ...



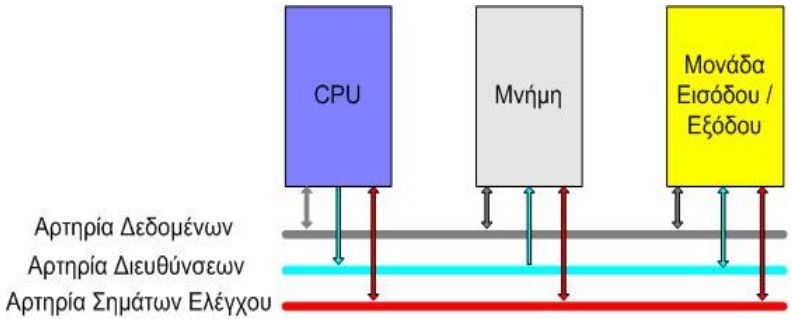
- Μάθαμε πως αναπαρίστανται οι πληροφορίες μας στον υπολογιστή.
- Είδαμε τρόπους με τους οποίους μπορούμε να κάνουμε αριθμητικές συναρτήσεις.
- Πιο σύνθετες συναρτήσεις φτιάχνονται από απλούστερες.

Η μεγάλη εικόνα ...



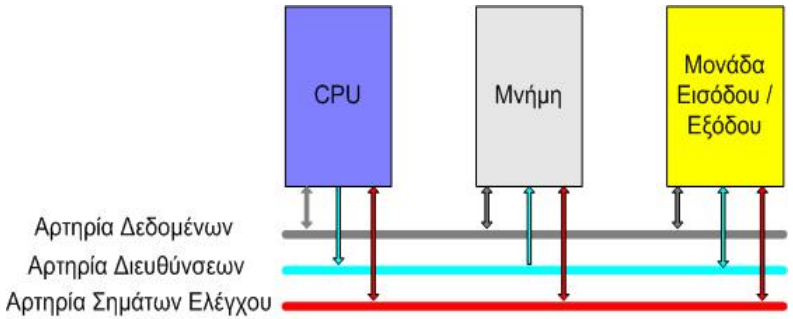
- Μάθαμε πως αναπαρίστανται οι πληροφορίες μας στον υπολογιστή.
- Είδαμε τρόπους με τους οποίους μπορούμε να κάνουμε αριθμητικές συναρτήσεις.
- Πιο σύνθετες συναρτήσεις φτιάχνονται από απλούστερες.
- Που και από ποιον εκτελούνται οι συναρτήσεις ?

Η μεγάλη εικόνα ...



- Μάθαμε πως αναπαρίστανται οι πληροφορίες μας στον υπολογιστή.
- Είδαμε τρόπους με τους οποίους μπορούμε να κάνουμε αριθμητικές συναρτήσεις.
- Πιο σύνθετες συναρτήσεις φτιάχνονται από απλούστερες.
- Που και από ποιον εκτελούνται οι συναρτήσεις ?
- Ακολουθεί ανάλυση της κάθε βασικής μονάδας και των συναρτήσεων που υλοποιεί.

Η μεγάλη εικόνα ...



- Μάθαμε πως αναπαρίστανται οι πληροφορίες μας στον υπολογιστή.
- Είδαμε τρόπους με τους οποίους μπορούμε να κάνουμε αριθμητικές συναρτήσεις.
- Πιο σύνθετες συναρτήσεις φτιάχνονται από απλούστερες.
- Που και από ποιον εκτελούνται οι συναρτήσεις ?
- Ακολουθεί ανάλυση της κάθε βασικής μονάδας και των συναρτήσεων που υλοποιεί.
- Στόχος : να περάσουμε στο επόμενο ιεραρχικό επίπεδο, αυτό των εντολών γλώσσας μηχανής.

Σύστημα Μνήμης: Γενικά

- Κυψελλίδα μνήμης (memory cell) : στοιχειώδης ποσότητα υλικού.
  - Μπορεί να βρίσκεται σε δύο καταστάσεις. Αρα μπορεί να αποθηκεύει ένα δυαδικό ψηφίο.
  - Ιστορικά φτιάχτηκε με διάφορες τεχνολογίες υλικών. Σήμερα από ημιαγωγούς.
  - Σμικρύνεται με τη πρόοδο της τεχνολογίας. Στους εμπορικά διαθέσιμους υπολογιστές, 1.000.000 κυψελίδες καταλαμβάνουν εμβαδόν ίσο με τη διατομή μιας ανθρώπινης τρίχας.

Σύστημα Μνήμης: Γενικά

- Κυψελλίδα μνήμης (memory cell) : στοιχειώδης ποσότητα υλικού.
  - Μπορεί να βρίσκεται σε δύο καταστάσεις. Αρα μπορεί να αποθηκεύει ένα δυαδικό ψηφίο.
  - Ιστορικά φτιάχτηκε με διάφορες τεχνολογίες υλικών. Σήμερα από ημιαγωγούς.
  - Σμικρύνεται με τη πρόοδο της τεχνολογίας. Στους εμπορικά διαθέσιμους υπολογιστές, 1.000.000 κυψελίδες καταλαμβάνουν εμβαδόν ίσο με τη διατομή μιας ανθρώπινης τρίχας.
- Το σύστημα μνήμης πρέπει :
  - Να δίνει επαρκή χωρητικότητα.
  - Στον ελάχιστο δυνατό φυσικό χώρο.
  - Με το ελάχιστο δυνατό κόστος.

Λέξη / Θέση Μνήμης

- Συνηθίζεται να αναπαριστούμε το σύστημα μνήμης με έναν δισδιάστατο πίνακα.
- Αυτό είναι ένα ικανοποιητικό επίπεδο αφαίρεσης για διδακτικούς λόγους. Αρκεί να μη λησμονούμε ότι στη πράξη δεν υπάρχει μήτε ένας πίνακας μήτε είναι δισδιάστατος.
- Το ελάχιστο ποσό πληροφορίας που μπορεί να προσπελασθεί από μια μνήμη ονομάζεται λέξη της μνήμης και αποτελεί την οριζόντια διάσταση του πίνακα.
- Προσοχή! Άλλο λέξη μνήμης και άλλο λέξη του επεξεργαστή!
- Ο αριθμός των διαφορετικών λέξεων που υπάρχουν στη δεδομένη χωρητικότητα της μνήμης είναι η άλλη διάσταση του πίνακα.
- Π.χ. Αν το σύστημα μνήμης αποτελείται από 512 κυψελίδες και η λέξη είναι των 32 δυαδικών ψηφίων, ο πίνακας μου θα έχει 512/32 = 16 διαφορετικές λέξεις (έστω Λέξη$_0$ έως Λέξη$_{15}$).

Η ανάγκη για διευθύνσεις

- Ας υποθέσουμε τον υπολογισμό $(5 + 3) \times (8 - 6)$.

Η ανάγκη για διευθύνσεις

- Ας υποθέσουμε τον υπολογισμό $(5+3) \times (8-6)$.
- Θυμηθείτε ότι με το συμπλήρωμα ως προς 2 αφαίρεση & πρόσθεση υλοποιούνται από το ίδιο υλικό. Αρα η πράξη (5+3) θα εκτελεστεί από το ίδιο υλικό με το (8-6), σε διαδοχικούς κύκλους.

Η ανάγκη για διευθύνσεις

- Ας υποθέσουμε τον υπολογισμό $(5+3) \times (8-6)$.
- Θυμηθείτε ότι με το συμπλήρωμα ως προς 2 αφαίρεση & πρόσθεση υλοποιούνται από το ίδιο υλικό. Αρα η πράξη (5+3) θα εκτελεστεί από το ίδιο υλικό με το (8-6), σε διαδοχικούς κύκλους.
- Αυτό γεννά την ανάγκη αποθήκευσης των ενδιάμεσων αποτελεσμάτων, άρα ανάγκη μνήμης.

Η ανάγκη για διευθύνσεις

- Ας υποθέσουμε τον υπολογισμό $(5+3) \times (8-6)$.
- Θυμηθείτε ότι με το συμπλήρωμα ως προς 2 αφαίρεση & πρόσθεση υλοποιούνται από το ίδιο υλικό. Αρα η πράξη (5+3) θα εκτελεστεί από το ίδιο υλικό με το (8-6), σε διαδοχικούς κύκλους.
- Αυτό γεννά την ανάγκη αποθήκευσης των ενδιάμεσων αποτελεσμάτων, άρα ανάγκη μνήμης.
- Αν υποθέσουμε τυχαίο τρόπο αποθήκευσης θα μπορούσε το $8 = (5+3)$, να αποθηκευτεί στη θέση Λέξη_4.

Η ανάγκη για διευθύνσεις

- Ας υποθέσουμε τον υπολογισμό $(5+3) \times (8-6)$.
- Θυμηθείτε ότι με το συμπλήρωμα ως προς 2 αφαίρεση & πρόσθεση υλοποιούνται από το ίδιο υλικό. Αρα η πράξη (5+3) θα εκτελεστεί από το ίδιο υλικό με το (8-6), σε διαδοχικούς κύκλους.
- Αυτό γεννά την ανάγκη αποθήκευσης των ενδιάμεσων αποτελεσμάτων, άρα ανάγκη μνήμης.
- Αν υποθέσουμε τυχαίο τρόπο αποθήκευσης θα μπορούσε το $8 = (5+3)$, να αποθηκευτεί στη θέση Λέξη_4.
- Δυστυχώς ο τυχαίος τρόπος δεν αποκλείει και το $2 = (8-6)$ να διαλέξει να αποθηκευτεί στη θέση Λέξη_4.

Η ανάγκη για διευθύνσεις

- Ας υποθέσουμε τον υπολογισμό $(5+3) \times (8-6)$.
- Θυμηθείτε ότι με το συμπλήρωμα ως προς 2 αφαίρεση & πρόσθεση υλοποιούνται από το ίδιο υλικό. Αρα η πράξη (5+3) θα εκτελεστεί από το ίδιο υλικό με το (8-6), σε διαδοχικούς κύκλους.
- Αυτό γεννά την ανάγκη αποθήκευσης των ενδιάμεσων αποτελεσμάτων, άρα ανάγκη μνήμης.
- Αν υποθέσουμε τυχαίο τρόπο αποθήκευσης θα μπορούσε το $8 = (5+3)$, να αποθηκευτεί στη θέση Λέξη_4.
- Δυστυχώς ο τυχαίος τρόπος δεν αποκλείει και το $2 = (8-6)$ να διαλέξει να αποθηκευτεί στη θέση Λέξη_4.
- Η θέση Λέξη_4 μπορεί δυστυχώς να συγκρατήσει μόνο μία πληροφορία ανά χρονική στιγμή. Αρα το πρώτο ενδιάμεσο αποτέλεσμα δε θα υπάρχει όταν χρειαστεί για τον πολλαπλασιασμό.

Η ανάγκη για διευθύνσεις

- Ας υποθέσουμε τον υπολογισμό $(5+3) \times (8-6)$.
- Θυμηθείτε ότι με το συμπλήρωμα ως προς 2 αφαίρεση & πρόσθεση υλοποιούνται από το ίδιο υλικό. Αρα η πράξη (5+3) θα εκτελεστεί από το ίδιο υλικό με το (8-6), σε διαδοχικούς κύκλους.
- Αυτό γεννά την ανάγκη αποθήκευσης των ενδιάμεσων αποτελεσμάτων, άρα ανάγκη μνήμης.
- Αν υποθέσουμε τυχαίο τρόπο αποθήκευσης θα μπορούσε το $8 = (5+3)$, να αποθηκευτεί στη θέση Λέξη_4.
- Δυστυχώς ο τυχαίος τρόπος δεν αποκλείει και το $2 = (8-6)$ να διαλέξει να αποθηκευτεί στη θέση Λέξη_4.
- Η θέση Λέξη_4 μπορεί δυστυχώς να συγκρατήσει μόνο μία πληροφορία ανά χρονική στιγμή. Αρα το πρώτο ενδιάμεσο αποτέλεσμα δε θα υπάρχει όταν χρειαστεί για τον πολλαπλασιασμό.
- Για την αποφυγή της τυχαιότητας, κάθε λέξη έχει μία διεύθυνση.

Ο πραγματικός υπολογισμός

- Ο υπολογισμός $(5+3) \times (8-6)$ θα μεταφραστεί στην εξής ακολουθία :
  - Εκτέλεσε το $5+3$. Αποθήκευσε το αποτέλεσμα στη θέση Λέξη$_x$. x διεύθυνση που δεν έχει εκείνη τη στιγμή κάποια χρήσιμη πληροφορία.
  - Εκτέλεσε το $8-6$. Αποθήκευσε το αποτέλεσμα στη θέση Λέξη$_y$. y διεύθυνση που δεν έχει εκείνη τη στιγμή κάποια χρήσιμη πληροφορία. Προφανώς $x \neq y$.
  - Ανακάλεσε τα περιεχόμενα των Λέξη$_x$ και Λέξη$_y$. Εστω [Λέξη$_x$] και [Λέξη$_y$] αντίστοιχα.
  - Εκτέλεσε [Λέξη$_x$] $\times$ [Λέξη$_y$]. Ο υπολογισμός ολοκληρώθηκε.

Ο πραγματικός υπολογισμός

- Ο υπολογισμός $(5+3) \times (8-6)$ θα μεταφραστεί στην εξής ακολουθία :
  - Εκτέλεσε το $5+3$. Αποθήκευσε το αποτέλεσμα στη θέση Λέξη$_x$. x διεύθυνση που δεν έχει εκείνη τη στιγμή κάποια χρήσιμη πληροφορία.
  - Εκτέλεσε το $8-6$. Αποθήκευσε το αποτέλεσμα στη θέση Λέξη$_y$. y διεύθυνση που δεν έχει εκείνη τη στιγμή κάποια χρήσιμη πληροφορία. Προφανώς $x \neq y$.
  - Ανακάλεσε τα περιεχόμενα των Λέξη$_x$ και Λέξη$_y$. Εστω [Λέξη$_x$] και [Λέξη$_y$] αντίστοιχα.
  - Εκτέλεσε [Λέξη$_x$] $\times$ [Λέξη$_y$]. Ο υπολογισμός ολοκληρώθηκε.
- Κάθε συναλλαγή με τη μνήμη γίνεται με μία θέση της που επιλέγεται βάσει της διεύθυνσης, την οποία υποχρεούται να παράσχει αυτός που θέλει τη συναλλαγή.

Ο πραγματικός υπολογισμός

- Ο υπολογισμός $(5+3) \times (8-6)$ θα μεταφραστεί στην εξής ακολουθία :
  - Εκτέλεσε το $5+3$. Αποθήκευσε το αποτέλεσμα στη θέση Λέξη$_x$. x διεύθυνση που δεν έχει εκείνη τη στιγμή κάποια χρήσιμη πληροφορία.
  - Εκτέλεσε το $8-6$. Αποθήκευσε το αποτέλεσμα στη θέση Λέξη$_y$. y διεύθυνση που δεν έχει εκείνη τη στιγμή κάποια χρήσιμη πληροφορία. Προφανώς $x \neq y$.
  - Ανακάλεσε τα περιεχόμενα των Λέξη$_x$ και Λέξη$_y$. Εστω [Λέξη$_x$] και [Λέξη$_y$] αντίστοιχα.
  - Εκτέλεσε [Λέξη$_x$] $\times$ [Λέξη$_y$]. Ο υπολογισμός ολοκληρώθηκε.
- Κάθε συναλλαγή με τη μνήμη γίνεται με μία θέση της που επιλέγεται βάσει της διεύθυνσης, την οποία υποχρεούται να παράσχει αυτός που θέλει τη συναλλαγή.
- Για ένα απλό υπολογισμό χρειαστήκαμε πολλές συναλαγές. Η γρήγορη μνήμη είναι προϋπόθεση για γρήγορα υπολογιστικά συστήματα.

Συναρτήσεις του συστήματος μνήμης

- Η μνήμη είναι ένα “χαζό” (παθητικό) στοιχείο του υπολογιστικού συστήματος. Υποστηρίζει δύο μόνο συναρτήσεις (προσπελάσεις μνήμης - memory accesses) :

Συναρτήσεις του συστήματος μνήμης

- Η μνήμη είναι ένα “χαζό” (παθητικό) στοιχείο του υπολογιστικού συστήματος. Υποστηρίζει δύο μόνο συναρτήσεις (προσπελάσεις μνήμης - memory accesses) :
  - Εγγραφή (write) : Η μνήμη αποθηκεύει στην υποδεικνυόμενη διεύθυνσή της, την υποδεικνυόμενη πληροφορία. Τυχόν προηγούμενα αποθηκευμένη πληροφορία εκεί, καταστρέφεται.
  - Ανάγνωση (read) : Η μνήμη παρέχει την πληροφορία που διατηρεί στην υποδεικνυόμενη διεύθυνσή της. Προφανώς συνεχίζει να αποθηκεύει την ίδια πληροφορία.

Μια διεύθυνση - Πολλαπλάσια πληροφορία

- Εστω ότι ενώ κάθε θέση μνήμης είναι του ενός byte, θέλω να αποθηκεύσω ακεραίους των 4 bytes.
- Μπορώ να αποφύγω 4 διαδοχικές πλήρεις προσπελάσεις ? Να στείλω μόνο 1 διεύθυνση ?

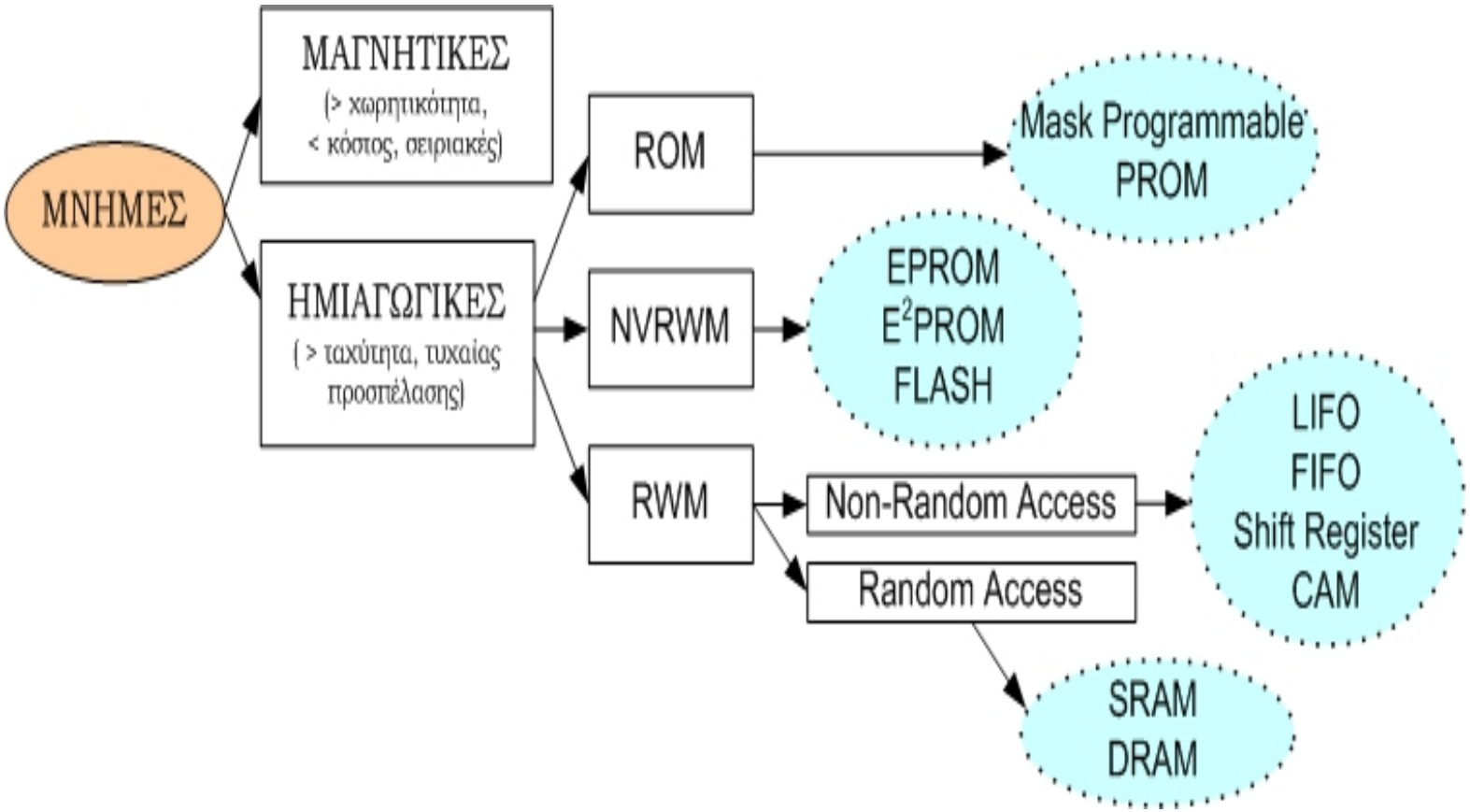
Μια διεύθυνση - Πολλαπλάσια πληροφορία

- Εστω ότι ενώ κάθε θέση μνήμης είναι του ενός byte, θέλω να αποθηκεύσω ακεραίους των 4 bytes.
- Μπορώ να αποφύγω 4 διαδοχικές πλήρεις προσπελάσεις ? Να στείλω μόνο 1 διεύθυνση ?
- Ναι, αλλά πέρα από το ότι πρέπει να ορίσω επιπλέον συναρτήσεις (εύκολο) :
  - Στη μικρότερη διεύθυνση θα αποθηκευτεί το LS Byte ή το MS Byte ? (Little Endian vs Big Endian).

Μια διεύθυνση - Πολλαπλάσια πληροφορία

- Εστω ότι ενώ κάθε θέση μνήμης είναι του ενός byte, θέλω να αποθηκεύσω ακεραίους των 4 bytes.
- Μπορώ να αποφύγω 4 διαδοχικές πλήρεις προσπελάσεις ? Να στείλω μόνο 1 διεύθυνση ?
- Ναι, αλλά πέρα από το ότι πρέπει να ορίσω επιπλέον συναρτήσεις (εύκολο) :
  - Στη μικρότερη διεύθυνση θα αποθηκευτεί το LS Byte ή το MS Byte ? (Little Endian vs Big Endian).
  - Θα υιοθετήσω κανόνες παράταξης (alignment) ή όχι ?

Κατηγοριοποίηση Μνημών



Ανάγκη για ιεραρχία μνήμης

- Θέλουμε :
  - Γρήγορη μνήμη
  - Μεγάλη μνήμη
  - Φθηνή μνήμη

Ανάγκη για ιεραρχία μνήμης

- Θέλουμε :
  - Γρήγορη μνήμη
  - Μεγάλη μνήμη
  - Φθηνή μνήμη
- Ομως :
  - Γρήγορη μνήμη ⇒ Ακριβή μνήμη
  - Μεγάλη μνήμη ⇒ Αργή μνήμη

Ανάγκη για ιεραρχία μνήμης

- Θέλουμε :
  - Γρήγορη μνήμη
  - Μεγάλη μνήμη
  - Φθηνή μνήμη
- Ομως :
  - Γρήγορη μνήμη ⇒ Ακριβή μνήμη
  - Μεγάλη μνήμη ⇒ Αργή μνήμη
- Είναι αδύνατο με μία μόνο διάταξη να πετύχω αυτά που θέλω.

Ανάγκη για ιεραρχία μνήμης

- Θέλουμε :
  - Γρήγορη μνήμη
  - Μεγάλη μνήμη
  - Φθηνή μνήμη
- Ομως :
  - Γρήγορη μνήμη ⇒ Ακριβή μνήμη
  - Μεγάλη μνήμη ⇒ Αργή μνήμη
- Είναι αδύνατο με μία μόνο διάταξη να πετύχω αυτά που θέλω.
- Μήπως εναλλακτικά μπορώ με πολλές διαφορετικές μνήμες ? Γιατί ? Και πως πρέπει να διαταχτούν αυτές ?

Γειτονικότητα των αναφορών

- Είμαστε τυχεροί καθώς τα προγράμματά μας έχουν τη τάση να επαναχρησιμοποιούν εντολές και δεδομένα που χρησιμοποίησαν πρόσφατα.

Γειτονικότητα των αναφορών

- Είμαστε τυχεροί καθώς τα προγράμματά μας έχουν τη τάση να επαναχρησιμοποιούν εντολές και δεδομένα που χρησιμοποίησαν πρόσφατα.
- Αν αυτό δεν ίσχυε, ο προσωπικός μας υπολογιστής θα έτρεχε τουλάχιστον 1000 φορές πιο αργά ή για την ίδια ταχύτητα θα χρειαζόταν να πληρώσουμε 1000 φορές περισσότερο.

Γειτονικότητα των αναφορών

- Είμαστε τυχεροί καθώς τα προγράμματά μας έχουν τη τάση να επαναχρησιμοποιούν εντολές και δεδομένα που χρησιμοποίησαν πρόσφατα.
- Αν αυτό δεν ίσχυε, ο προσωπικός μας υπολογιστής θα έτρεχε τουλάχιστον 1000 φορές πιο αργά ή για την ίδια ταχύτητα θα χρειαζόταν να πληρώσουμε 1000 φορές περισσότερο.
- "A program spends 90% of its time in 10% of its code"
- Συνέπεια της γειτονικότητας είναι ότι μπορούμε να προβλέψουμε με αρκετή ακρίβεια τι δεδομένα θα χρειαστεί από το σύστημα μνήμης ο υπολογισμός μας στο άμεσο μέλλον. Η πρόβλεψη στηρίζεται στα όσα έλαβαν χώρα στο παρελθόν.

Γειτονικότητα των αναφορών

- Είμαστε τυχεροί καθώς τα προγράμματά μας έχουν τη τάση να επαναχρησιμοποιούν εντολές και δεδομένα που χρησιμοποίησαν πρόσφατα.
- Αν αυτό δεν ίσχυε, ο προσωπικός μας υπολογιστής θα έτρεχε τουλάχιστον 1000 φορές πιο αργά ή για την ίδια ταχύτητα θα χρειαζόταν να πληρώσουμε 1000 φορές περισσότερο.
- "A program spends 90% of its time in 10% of its code"
- Συνέπεια της γειτονικότητας είναι ότι μπορούμε να προβλέψουμε με αρκετή ακρίβεια τι δεδομένα θα χρειαστεί από το σύστημα μνήμης ο υπολογισμός μας στο άμεσο μέλλον. Η πρόβλεψη στηρίζεται στα όσα έλαβαν χώρα στο παρελθόν.
- Η γειτονικότητα έχει 2 συνιστώσες :
  - Χρονική : Δεδομένα που πρόσφατα χρησιμοποιήθηκαν θα χρησιμοποιηθούν στο άμεσο μέλλον με μεγάλη πιθανότητα.
  - Τοπική : Δεδομένα με γειτονικές διευθύνσεις είναι πολύ πιθανόν να χρησιμοποιηθούν πολύ κοντά χρονικά.

Διάταξη των επιπέδων

- Η μνήμη μας αποτελείται από πολλά επίπεδα. Οσο πιο γρήγορα είναι τόσο και πιο μικρά ώστε το κόστος να παραμένει λογικό. Οσο πιο αργά τόσο πιο μεγάλα.
- Κάθε μικρότερο επίπεδο περιέχει αντίγραφο μέρους της πληροφορίας του αμέσως μεγαλύτερου.
- Κάθε προσπέλαση μνήμης είτε εξυπηρετείται από το πιο γρήγορο επίπεδο, είτε αυτό λόγω του μεγέθους του δε περιέχει τη ζητούμενη πληροφορία.
- Για την εκμετάλλευση των δύο συνιστωσών της γειτονικότητας, η πληροφορία που έλειπε και οι γειτονικές της μεταφέρονται στο πιο γρήγορο επίπεδο, αφού είναι οι πλέον πιθανές να χρησιμοποιηθούν στο μέλλον.

Επίπεδα Ιεραρχίας Μνήμης

Ονομα	Μέγεθος	Χρόνος Προσπέλασης	Υλοποίηση με
Καταχωρητές	64 - 16Kb	$< 0{,}5$ns	SRAM
Cache 1ου επιπέδου	1KB - 1MB	1-4ns	SRAM
Cache 2ου επιπέδου	512KB - 2MB	5-15ns	SRAM
Κύρια Μνήμη	64MB - 4GB	6-30ns	DRAM
Μαγνητικοί Δίσκοι	200GB - 8TB	$> 5\mu$s	Μαγνητικό υλικό
Μονάδα Backup	$>$ 1TB	$> 500\mu$s	Οπτικό υλικό

Ιδεατή μνήμη

- Παρότι η κύρια μνήμη των σημερινών υπολογιστών έχει αυξηθεί, υπάρχουν προγράμματα που σε μέγεθος ξεπερνούν ακόμη και αυτή τη χωρητικότητα.
- Επίσης πολλές φορές χρησιμοποιούμε παράλληλα πολλές εφαρμογές που συνολικά μπορεί να περιορίζουν τη διαθέσιμη μνήμη για κάποιες άλλες εφαρμογές.
- Προκύπτει συνεπώς το ερώτημα, αν αυτές οι εφαρμογές μπορούν να εκτελεστούν σε ένα σύστημα χαμηλής μνήμης.
- Η ιδεατή μνήμη (virtual memory) επιτρέπει στο χρήστη να βλέπει την κύρια μνήμη και μέρος του χώρου στο δίσκο σα μια ενιαία πολύ μεγάλη μνήμη.

Λογικές Διευθύνσεις

- Η ΚΜΕ σε αυτή την περίπτωση παράγει λογικές διευθύνσεις (logical / virtual addresses).
- Αυτές μεταφράζονται σε πραγματικές διευθύνσεις από τη Μονάδα Διαχείρισης Μνήμης (Memory Management Unit - MMU).
- Η MMU συνεργάζεται με το λειτουργικό σύστημα έτσι ώστε όταν ένα κομμάτι κώδικα δεν υπάρχει στην κύρια μνήμη, να μεταφερθεί σε αυτήν.
- Για την εκμετάλλευση της γειτονικότητας προφανώς θα μετφερθεί ένα μεγάλο κομμάτι που θα περιλαμβάνει και την πληροφρορία που ζητείται.
- Το κομμάτι αυτό ονόμάζεται στην περίπτωση δίσκου - κύριας μνήμης σελίδα.

Overlays

- Η ιδεατή μνήμη στις μέρες μας θεωρείται δεδομένη.
- Ως προγραμματιστές ή χρήστες εφαρμογών δε θα καταλάβουμε ποτέ την ύπαρξή της.
- Στους πρώτους υπολογιστές ωστόσο η ευθύνη μεταφοράς κομματιών ενός προγράμματος από και προς το δίσκο (υπήρχε και η περίπτωση του self modifying code) ήταν αποκλειστικά θέμα του προγραμματιστή.
- Ο προγραμματισμός γινόταν σε επικαλυπτόμενα μέρη overlays τα οποία ο προγραμματιστής κινούσε μεταξύ κύριας μνήμης και δίσκου.
- Ο προγραμματισμός σε υψηλά επίπεδα τότε ήταν πολύ πιο επίπονος απ' ότι σήμερα.
- Ιστορικά το λειτουργικό σύστημα (software) ανέλαβε στη συνέχεια όλη τη διαδικασία για να φτάσουμε στις μέρες μας που επιστρατεύουμε την MMU (υλικό) για να πετύχουμε καλύτερη απόδοση.

Δομή μιας ΚΜΕ

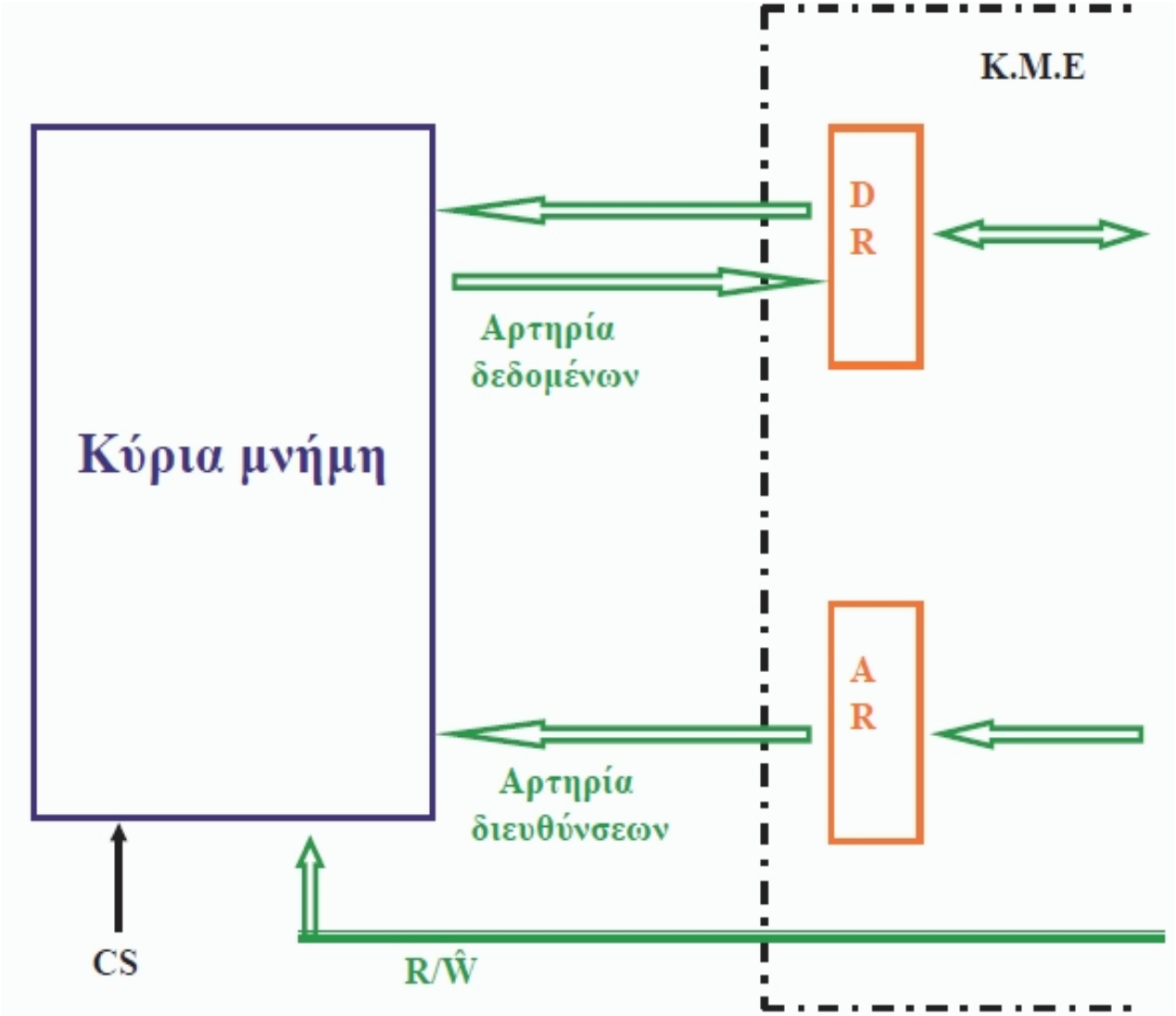
Καταχωρητές της ΚΜΕ

- Οι καταχωρητές είναι η πιο μικρή, αλλά και η πιο γρήγορη μνήμη του συστήματός μας.
- Υπάρχουν καταχωρητές
  - γενικού σκοπού που χρησιμεύουν για τη καταχώρηση ενδιάμεσων αποτελεσμάτων
  - ειδικού σκοπού που εξυπηρετούν την ακολουθία εκτέλεσης.

Καταχωρητές Ειδικού Σκοπού

- Μετρητής προγράμματος (Program Counter - PC). Κρατά τη διεύθυνση μνήμης της επόμενης προς εκτέλεση εντολής.
- Καταχωρητής εντολής (Instruction Register - IR). Αποθηκεύει τον κωδικό λειτουργίας της εντολής που εκτελείται.
- Καταχωρητής κατάστασης ((Program) Status Register - PSR / SR). Αποθηκεύει ενδείκτες κατάστασης από την εκτέλεση της προηγούμενης εντολής.
- Καταχωρητής διεύθυνσης μνήμης (Memory Address Register - MAR). Αποθηκεύει τη διεύθυνση μνήμης που θα προσπελάσει η ΚΜΕ.
- Καταχωρητής δεδομένων μνήμης (Memory Data Register - MDR). Αποθηκεύει τα δεδομένα που θα γραφούν στη μνήμη ή τα δεδομένα που διαβάστηκαν από αυτή.
- Δείκτης Στοίβας (Stack Pointer - SP). Δείχνει (ανάλογα με την υλοποίηση είτε το κορυφαίο, είτε την επόμενη κενή θέση της στοίβας.

Επικοινωνία ΚΜΕ - Μνήμης



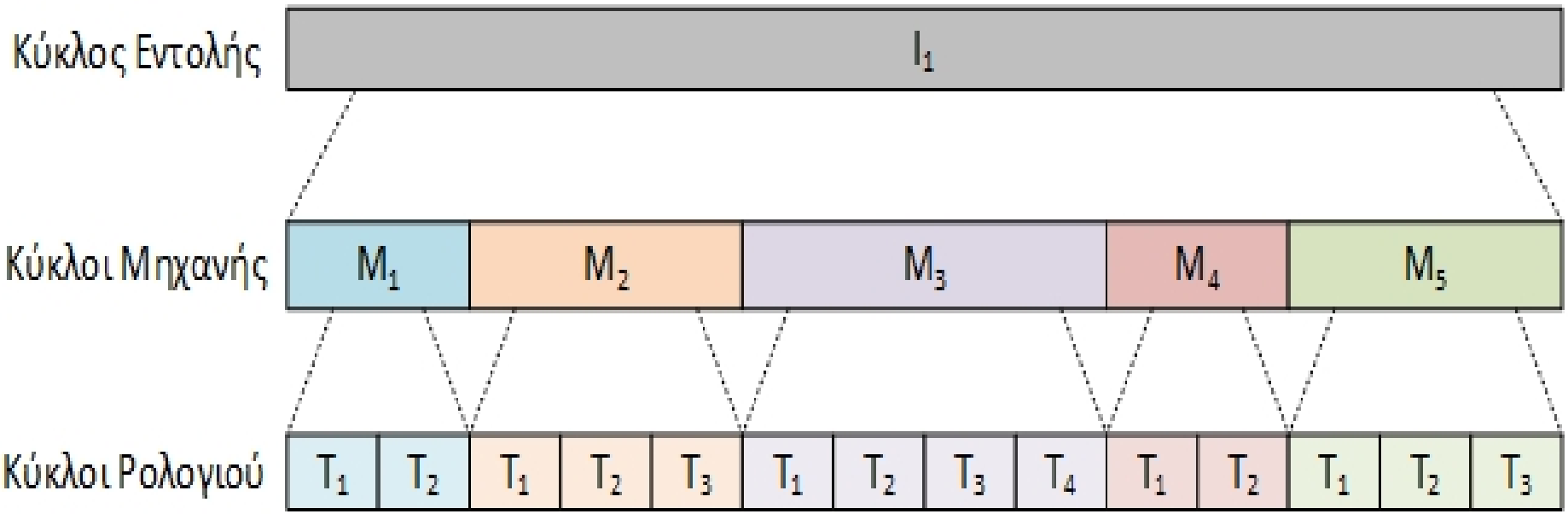
Μονάδα ελέγχου

- Συγχρονίζει την εκτέλεση μιας εντολής, παράγοντας σήματα για την απρόσκοπτη επικοινωνία των υπομονάδων που απαιτούνται για την εκτέλεση της εντολής.
- Π.χ. Για την εγγραφή σε μνήμη πρέπει
  - Στοχευόμενη διεύθυνση → καταχωρητής διευθύνσεων μνήμης.
  - Δεδομένα που θα γραφούν → καταχωρητής δεδομένων μνήμης.
  - Δημιουργία του σήματος εγγραφής στη μνήμη.
- Μικρολειτουργίες που ωστόσο απαιτούν αυστηρό χρονισμό, άρα έχουν μια ακολουθία.
- Αφού μια ακολουθία μικροεντολών απαιτείται για τη διεκπεραίωση μιας πιο σύνθετης λειτουργίας μπορούμε σε ένα πιο αφαιρετικό επίπεδο να θεωρούμε ότι η μονάδα ελέγχου εκτελεί μικροπρογράμματα (ακολουθίες μικρολειτουργιών) που διερμηνεύουν εντολές του πιο υψηλού ιεραρχικά επιπέδου.

Χρονισμός μιας εντολής

- Παρότι οι εντολές γλώσσας μηχανής που θα δούμε στη συνέχεια είναι εξαιρετικά κοντά στο αλφάβητο της μηχανής, κι αυτές δεν εκτελούνται ατομικά. Διαιρούνται περαιτέρω σε:
  - Κύκλους μηχανής (machine cycles). Σε κάθε κύκλο μηχανής ολοκληρώνεται μια μικρολειτουργία (π.χ. ανάγνωση από τη μνήμη, πρόσθεση, ...).
  - Κύκλους ρολογιού (clock cycles). Είναι η στοιχειώδης μονάδα καθυστέρησης ενός υπολογιστικού συστήματος. Κάθε κύκλος μηχανής μπορεί να απαιτεί από 1 έως (συνήθως) 5 κύκλους ρολογιού για να ολοκληρωθεί.

Χρονισμός μιας εντολής



Opcodes & Γλώσσα μηχανής

- Ο υπολογιστής έχει συγκεκριμένο σύνολο συναρτήσεων υλοποιημένο σε υλικό $\Rightarrow$ εκτελεί ένα πεπερασμένο σύνολο διαφορετικών λειτουργιών.

Opcodes & Γλώσσα μηχανής

- Ο υπολογιστής έχει συγκεκριμένο σύνολο συναρτήσεων υλοποιημένο σε υλικό $\Rightarrow$ εκτελεί ένα πεπερασμένο σύνολο διαφορετικών λειτουργιών.
- Αν σε κάθε λειτουργία αναθέσουμε ένα δυαδικό κωδικό (operation code - opcode) μπορούμε να διαχειριζόμαστε αυτές τις εντολές όπως όλες τις υπόλοιπες δυαδικές πληροφορίες.

Opcodes & Γλώσσα μηχανής

- Ο υπολογιστής έχει συγκεκριμένο σύνολο συναρτήσεων υλοποιημένο σε υλικό $\Rightarrow$ εκτελεί ένα πεπερασμένο σύνολο διαφορετικών λειτουργιών.
- Αν σε κάθε λειτουργία αναθέσουμε ένα δυαδικό κωδικό (operation code - opcode) μπορούμε να διαχειριζόμαστε αυτές τις εντολές όπως όλες τις υπόλοιπες δυαδικές πληροφορίες.
- Κάθε λειτουργία είναι στη πραγματικότητα μια κατηγορία λειτουργιών. Π.χ. Η λειτουργία πρόσθεσης μπορεί να περιλαμβάνει πρόσθεση λέξεων ή μεγαλύτερων ποσοτήτων.

Opcodes & Γλώσσα μηχανής

- Ο υπολογιστής έχει συγκεκριμένο σύνολο συναρτήσεων υλοποιημένο σε υλικό ⇒ εκτελεί ένα πεπερασμένο σύνολο διαφορετικών λειτουργιών.
- Αν σε κάθε λειτουργία αναθέσουμε ένα δυαδικό κωδικό (operation code - opcode) μπορούμε να διαχειριζόμαστε αυτές τις εντολές όπως όλες τις υπόλοιπες δυαδικές πληροφορίες.
- Κάθε λειτουργία είναι στη πραγματικότητα μια κατηγορία λειτουργιών. Π.χ. Η λειτουργία πρόσθεσης μπορεί να περιλαμβάνει πρόσθεση λέξεων ή μεγαλύτερων ποσοτήτων.
- Διαφοροποιήσεις μέσα σε μια κατηγορία απαιτούν την ανάθεση διαφορετικών κωδικών λειτουργίας, έτσι ώστε η μονάδα ελέγχου να μπορεί να εκτελεί τις διαφορετικές ακολουθίες μικρολειτουργιών που χρειάζονται για την σωστή εκτέλεση κάθε μίας.

Opcodes & Γλώσσα μηχανής

- Ο υπολογιστής έχει συγκεκριμένο σύνολο συναρτήσεων υλοποιημένο σε υλικό ⇒ εκτελεί ένα πεπερασμένο σύνολο διαφορετικών λειτουργιών.
- Αν σε κάθε λειτουργία αναθέσουμε ένα δυαδικό κωδικό (operation code - opcode) μπορούμε να διαχειριζόμαστε αυτές τις εντολές όπως όλες τις υπόλοιπες δυαδικές πληροφορίες.
- Κάθε λειτουργία είναι στη πραγματικότητα μια κατηγορία λειτουργιών. Π.χ. Η λειτουργία πρόσθεσης μπορεί να περιλαμβάνει πρόσθεση λέξεων ή μεγαλύτερων ποσοτήτων.
- Διαφοροποιήσεις μέσα σε μια κατηγορία απαιτούν την ανάθεση διαφορετικών κωδικών λειτουργίας, έτσι ώστε η μονάδα ελέγχου να μπορεί να εκτελεί τις διαφορετικές ακολουθίες μικρολειτουργιών που χρειάζονται για την σωστή εκτέλεση κάθε μίας.
- Γλώσσα μηχανής : Κωδικοί λειτουργίας και δεδομένων με 0 και 1.

Opcodes & Γλώσσα μηχανής

- Ο υπολογιστής έχει συγκεκριμένο σύνολο συναρτήσεων υλοποιημένο σε υλικό ⇒ εκτελεί ένα πεπερασμένο σύνολο διαφορετικών λειτουργιών.
- Αν σε κάθε λειτουργία αναθέσουμε ένα δυαδικό κωδικό (operation code - opcode) μπορούμε να διαχειριζόμαστε αυτές τις εντολές όπως όλες τις υπόλοιπες δυαδικές πληροφορίες.
- Κάθε λειτουργία είναι στη πραγματικότητα μια κατηγορία λειτουργιών. Π.χ. Η λειτουργία πρόσθεσης μπορεί να περιλαμβάνει πρόσθεση λέξεων ή μεγαλύτερων ποσοτήτων.
- Διαφοροποιήσεις μέσα σε μια κατηγορία απαιτούν την ανάθεση διαφορετικών κωδικών λειτουργίας, έτσι ώστε η μονάδα ελέγχου να μπορεί να εκτελεί τις διαφορετικές ακολουθίες μικρολειτουργιών που χρειάζονται για την σωστή εκτέλεση κάθε μίας.
- Γλώσσα μηχανής : Κωδικοί λειτουργίας και δεδομένων με 0 και 1.
- Παρά τον απόλυτο έλεγχο του υλικού, ένα πρόγραμμα απαιτεί σημαντική προσπάθεια συγγραφής σε γλώσσα μηχανής.

Εκτέλεση εντολής : Μηχανή πεπερασμένων καταστάσεων

- Προσκόμμιση - Αποκωδικοποίηση opcode :
  - MAR ← PC
  - IR ← MDR
  - Αποκωδικοποίηση
- Προσαγωγή εντέλων
  - MAR ← Διεύθυνση εντέλου
  - Καταχωρητή ← MDR
  - Επανάληψη αν χρειάζεται
- Εκτέλεση (ενεργοποίηση κατάλληλης μονάδας υλικού)
- Αποθήκευση αποτελέσματος
- Προετοιμασία για επόμενη εντολή
  - Συνήθως PC ← PC+k

Τι χρειαζόμαστε μια συμβολική γλώσσα ?

- Χρήση γλώσσας μηχανής ⇒
  - Πλήρης έλεγχος της μηχανής
  - >> χρόνο συγγραφής προγραμμάτων
  - >> κόπο συγγραφής προγραμμάτων
  - >> δυσκολία αποσφαλμάτωσης

Τι χρειαζόμαστε μια συμβολική γλώσσα ?

- Χρήση γλώσσας μηχανής ⇒
  - Πλήρης έλεγχος της μηχανής
  - >> χρόνο συγγραφής προγραμμάτων
  - >> κόπο συγγραφής προγραμμάτων
  - >> δυσκολία αποσφαλμάτωσης
- Χρήση υψηλής γλώσσας ⇒
  - Ολα εύκολα
  - καθόλου έλεγχο στο τι πραγματικά συμβαίνει στη μηχανή

Τι χρειαζόμαστε μια συμβολική γλώσσα ?

- Χρήση γλώσσας μηχανής ⇒
  - Πλήρης έλεγχος της μηχανής
  - >> χρόνο συγγραφής προγραμμάτων
  - >> κόπο συγγραφής προγραμμάτων
  - >> δυσκολία αποσφαλμάτωσης
- Χρήση υψηλής γλώσσας ⇒
  - Ολα εύκολα
  - καθόλου έλεγχο στο τι πραγματικά συμβαίνει στη μηχανή
- Συμβολική γλώσσα = χρυσή τομή. Χρήση της ⇒
  - Πλήρη έλεγχο της μηχανής
  - Πολύ λιγότερο χρόνο, κόπο και δυσκολία αποσφαλμάτωσης ενός προγράμματος

Τι χρειαζόμαστε μια συμβολική γλώσσα ?

- Χρήση γλώσσας μηχανής ⇒
  - Πλήρης έλεγχος της μηχανής
  - >> χρόνο συγγραφής προγραμμάτων
  - >> κόπο συγγραφής προγραμμάτων
  - >> δυσκολία αποσφαλμάτωσης
- Χρήση υψηλής γλώσσας ⇒
  - Ολα εύκολα
  - καθόλου έλεγχο στο τι πραγματικά συμβαίνει στη μηχανή
- Συμβολική γλώσσα = χρυσή τομή. Χρήση της ⇒
  - Πλήρη έλεγχο της μηχανής
  - Πολύ λιγότερο χρόνο, κόπο και δυσκολία αποσφαλμάτωσης ενός προγράμματος
- Οσο θα υπάρχουν κρίσιμα (σε χρόνο ή μέγεθος) κομμάτια κώδικα θα συνεχίσουν να υπάρχουν προγραμματιστές σε συμβολική γλώσσα και να αμείβονται πολύ καλά.

ISA & Assembly

- Κάθε ολοκληρωμένο ΚΜΕ έχει το δικό του σύνολο συναρτήσεων (ρεπερτόριο εντολών).

ISA & Assembly

- Κάθε ολοκληρωμένο ΚΜΕ έχει το δικό του σύνολο συναρτήσεων (ρεπερτόριο εντολών).
- Το ρεπερτόριο ορίζει ένα αρχιτεκτονικό επίπεδο, το οποίο ονομάζεται αρχιτεκτονική συνόλου εντολών (Instruction Set Architecture - ISA).

ISA & Assembly

- Κάθε ολοκληρωμένο ΚΜΕ έχει το δικό του σύνολο συναρτήσεων (ρεπερτόριο εντολών).
- Το ρεπερτόριο ορίζει ένα αρχιτεκτονικό επίπεδο, το οποίο ονομάζεται αρχιτεκτονική συνόλου εντολών (Instruction Set Architecture - ISA).
- Υπάρχει περίπτωση δύο μηχανές να παρέχουν την ίδια ISA χωρίς η υλοποίηση των κατωτέρων επιπέδων να είναι ίδια. Αυτές οι μηχανές θα είναι συμβατές σε επίπεδο αρχιτεκτονικής συνόλου εντολών (π.χ. AMD vs Intel processors).

ISA & Assembly

- Κάθε ολοκληρωμένο ΚΜΕ έχει το δικό του σύνολο συναρτήσεων (ρεπερτόριο εντολών).
- Το ρεπερτόριο ορίζει ένα αρχιτεκτονικό επίπεδο, το οποίο ονομάζεται αρχιτεκτονική συνόλου εντολών (Instruction Set Architecture - ISA).
- Υπάρχει περίπτωση δύο μηχανές να παρέχουν την ίδια ISA χωρίς η υλοποίηση των κατωτέρων επιπέδων να είναι ίδια. Αυτές οι μηχανές θα είναι συμβατές σε επίπεδο αρχιτεκτονικής συνόλου εντολών (π.χ. AMD vs Intel processors).
- Παρότι θα μιλήσουμε πολύ γενικά για τη συμβολική γλώσσα, όλα τα παραδείγματά μας θα είναι προσαρμοσμένα στον 8085 και στο AT91, το σύστημα που ενδεχόμενα να διεξαχθούν τα εργαστήριά σας.

ISA & Assembly

- Κάθε ολοκληρωμένο ΚΜΕ έχει το δικό του σύνολο συναρτήσεων (ρεπερτόριο εντολών).
- Το ρεπερτόριο ορίζει ένα αρχιτεκτονικό επίπεδο, το οποίο ονομάζεται αρχιτεκτονική συνόλου εντολών (Instruction Set Architecture - ISA).
- Υπάρχει περίπτωση δύο μηχανές να παρέχουν την ίδια ISA χωρίς η υλοποίηση των κατωτέρων επιπέδων να είναι ίδια. Αυτές οι μηχανές θα είναι συμβατές σε επίπεδο αρχιτεκτονικής συνόλου εντολών (π.χ. AMD vs Intel processors).
- Παρότι θα μιλήσουμε πολύ γενικά για τη συμβολική γλώσσα, όλα τα παραδείγματά μας θα είναι προσαρμοσμένα στον 8085 και στο AT91, το σύστημα που ενδεχόμενα να διεξαχθούν τα εργαστήριά σας.
- Το AT91 είναι ένα πρωτότυπο σύστημα φτιαγμένο από εσάς για εσάς βασισμένο στον ARM 7 TDMI.

Διαδικασία εκτέλεσης κώδικα Assembly

- Χρήση κειμενογράφου (editor) για αναγραφή πηγαίου κώδικα.

Διαδικασία εκτέλεσης κώδικα Assembly

- Χρήση κειμενογράφου (editor) για αναγραφή πηγαίου κώδικα.
- Χρήση Assembler (συμβολομεταφραστή - compiler) για μετάφραση σε (μεταθετό) κώδικα μηχανής.

Διαδικασία εκτέλεσης κώδικα Assembly

- Χρήση κειμενογράφου (editor) για αναγραφή πηγαίου κώδικα.
- Χρήση Assembler (συμβολομεταφραστή - compiler) για μετάφραση σε (μεταθετό) κώδικα μηχανής.
- Χρήση Loader (φορτωτή) για τοποθέτηση κώδικα σε κάποια διαθέσιμη περιοχή μνήμης.

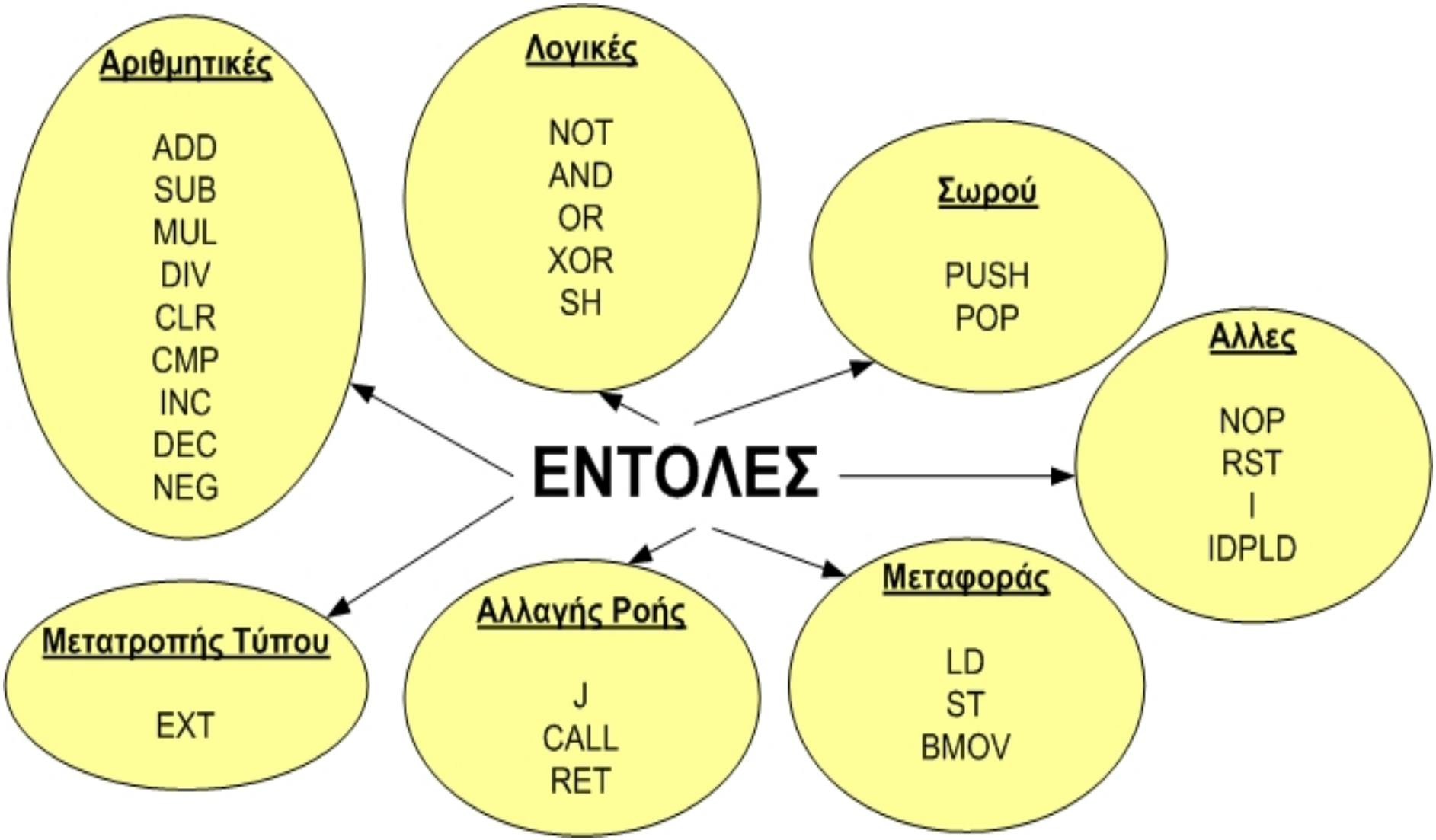
Διαδικασία εκτέλεσης κώδικα Assembly

- Χρήση κειμενογράφου (editor) για αναγραφή πηγαίου κώδικα.
- Χρήση Assembler (συμβολομεταφραστή - compiler) για μετάφραση σε (μεταθετό) κώδικα μηχανής.
- Χρήση Loader (φορτωτή) για τοποθέτηση κώδικα σε κάποια διαθέσιμη περιοχή μνήμης.
- Αρχικοποίηση του PC με τη διεύθυνση της μνήμης στην οποία βρίσκεται η αρχή του κώδικά μας.

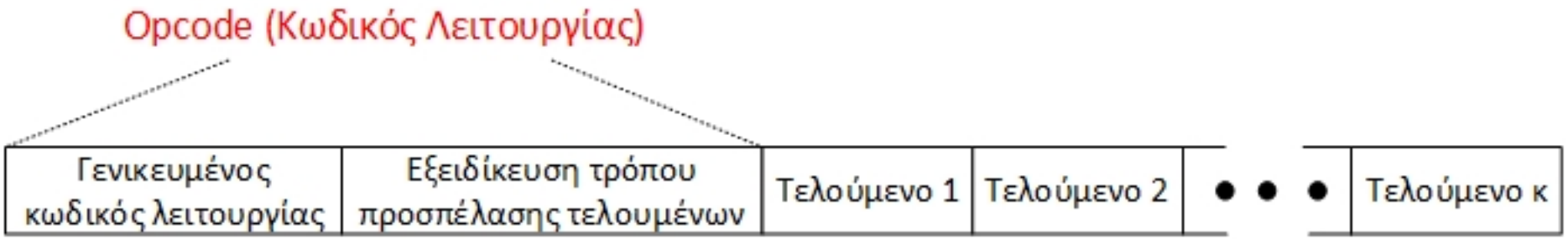
Διαδικασία εκτέλεσης κώδικα Assembly

- Χρήση κειμενογράφου (editor) για αναγραφή πηγαίου κώδικα.
- Χρήση Assembler (συμβολομεταφραστή - compiler) για μετάφραση σε (μεταθετό) κώδικα μηχανής.
- Χρήση Loader (φορτωτή) για τοποθέτηση κώδικα σε κάποια διαθέσιμη περιοχή μνήμης.
- Αρχικοποίηση του PC με τη διεύθυνση της μνήμης στην οποία βρίσκεται η αρχή του κώδικά μας.
- Program running ...

Κατηγοριοποίηση εντολών - Γενικοί κωδικοί



Μορφή μιας εντολής



ADD R_1, R_2, R_3
ADD #25
ADD R_1, [R_2]
ADD R_1, [R_2, #4]

Πινακίδες (Labels)

- Η συνολική μορφή των εντολών είναι : $\{[LABEL]\}$ OP-CODE $\{OPERAND_1\}$ $\{,OPERAND_2\}$ $\{,OPERAND_3\}$. Οτι είναι εντός των { } είναι προαιρετικό ή προαιρετικό ανάλογα με την εντολή.

Πινακίδες (Labels)

- Η συνολική μορφή των εντολών είναι : $\{[LABEL]\}$ OP-CODE $\{OPERAND_1\}$ $\{,OPERAND_2\}$ $\{,OPERAND_3\}$. Οτι είναι εντός των { } είναι προαιρετικό ή προαιρετικό ανάλογα με την εντολή.
- Η επισύναψη πινακίδας χρειάζεται για τη σηματοδότηση κάποιων σημείων του κώδικα, ως στόχων εντολών αλλαγής ροής.

Πινακίδες (Labels)

- Η συνολική μορφή των εντολών είναι : $\{[LABEL]\}$ OP-CODE $\{OPERAND_1\}$ $\{,OPERAND_2\}$ $\{,OPERAND_3\}$. Οτι είναι εντός των { } είναι προαιρετικό ή προαιρετικό ανάλογα με την εντολή.
- Η επισύναψη πινακίδας χρειάζεται για τη σηματοδότηση κάποιων σημείων του κώδικα, ως στόχων εντολών αλλαγής ροής.
- Οταν γράφουμε τον κώδικα Assembly δε γνωρίζουμε τη θέση μνήμης που θα βρίσκεται κάθε εντολή του κώδικά μας. Αλλωστε αυτό δεν ανήκει στα καθήκοντα του προγραμματιστή.

Πινακίδες (Labels)

- Η συνολική μορφή των εντολών είναι : $\{[LABEL]\}$ OP-CODE $\{OPERAND_1\}$ $\{,OPERAND_2\}$ $\{,OPERAND_3\}$. Οτι είναι εντός των { } είναι προαιρετικό ή προαιρετικό ανάλογα με την εντολή.
- Η επισύναψη πινακίδας χρειάζεται για τη σηματοδότηση κάποιων σημείων του κώδικα, ως στόχων εντολών αλλαγής ροής.
- Οταν γράφουμε τον κώδικα Assembly δε γνωρίζουμε τη θέση μνήμης που θα βρίσκεται κάθε εντολή του κώδικά μας. Αλλωστε αυτό δεν ανήκει στα καθήκοντα του προγραμματιστή.
- Οι διευθύνσεις αυτές θα γίνουν γνωστές όταν ο κώδικάς μας θα είναι έτοιμος να τρέξει.

Πινακίδες (Labels)

- Η συνολική μορφή των εντολών είναι : $\{[LABEL]\}$ OP-CODE $\{OPERAND_1\}$ $\{,OPERAND_2\}$ $\{,OPERAND_3\}$. Οτι είναι εντός των { } είναι προαιρετικό ή προαιρετικό ανάλογα με την εντολή.
- Η επισύναψη πινακίδας χρειάζεται για τη σηματοδότηση κάποιων σημείων του κώδικα, ως στόχων εντολών αλλαγής ροής.
- Οταν γράφουμε τον κώδικα Assembly δε γνωρίζουμε τη θέση μνήμης που θα βρίσκεται κάθε εντολή του κώδικά μας. Αλλωστε αυτό δεν ανήκει στα καθήκοντα του προγραμματιστή.
- Οι διευθύνσεις αυτές θα γίνουν γνωστές όταν ο κώδικάς μας θα είναι έτοιμος να τρέξει.
- Οι πινακίδες κάνουν το κώδικά μας μεταθετό.

Διευθυνσιοδότηση

- Πέραν του γενικευμένου κωδικού λειτουργίας το opcode μιας εντολής περιέχει και την εξειδίκευση του τρόπου προσπέλασης των εντέλων.

Διευθυνσιοδότηση

- Πέραν του γενικευμένου κωδικού λειτουργίας το opcode μιας εντολής περιέχει και την εξειδίκευση του τρόπου προσπέλασης των εντέλων.
- Συνήθως δε γνωρίζουμε τις τιμές των εντέλων που θα χρησιμοποιηθούν σε κάποια εντολή, καθώς στις περισσότερες περιπτώσεις έχουν προκύψει ως ενδιάμεσα αποτελέσματα προηγούμενων υπολογισμών.

Διευθυνσιοδότηση

- Πέραν του γενικευμένου κωδικού λειτουργίας το opcode μιας εντολής περιέχει και την εξειδίκευση του τρόπου προσπέλασης των εντέλων.
- Συνήθως δε γνωρίζουμε τις τιμές των εντέλων που θα χρησιμοποιηθούν σε κάποια εντολή, καθώς στις περισσότερες περιπτώσεις έχουν προκύψει ως ενδιάμεσα αποτελέσματα προηγούμενων υπολογισμών.
- Κάποιες φορές γνωρίζουμε τις διευθύνσεις τους.

Διευθυνσιοδότηση

- Πέραν του γενικευμένου κωδικού λειτουργίας το opcode μιας εντολής περιέχει και την εξειδίκευση του τρόπου προσπέλασης των εντέλων.
- Συνήθως δε γνωρίζουμε τις τιμές των εντέλων που θα χρησιμοποιηθούν σε κάποια εντολή, καθώς στις περισσότερες περιπτώσεις έχουν προκύψει ως ενδιάμεσα αποτελέσματα προηγούμενων υπολογισμών.
- Κάποιες φορές γνωρίζουμε τις διευθύνσεις τους.
- Αρκετές φορές όμως δε γνωρίζουμε ούτε τις διευθύνσεις τους, αλλά μόνο κάποιο τρόπο υπολογισμού τους.

Διευθυνσιοδότηση

- Πέραν του γενικευμένου κωδικού λειτουργίας το opcode μιας εντολής περιέχει και την εξειδίκευση του τρόπου προσπέλασης των εντέλων.
- Συνήθως δε γνωρίζουμε τις τιμές των εντέλων που θα χρησιμοποιηθούν σε κάποια εντολή, καθώς στις περισσότερες περιπτώσεις έχουν προκύψει ως ενδιάμεσα αποτελέσματα προηγούμενων υπολογισμών.
- Κάποιες φορές γνωρίζουμε τις διευθύνσεις τους.
- Αρκετές φορές όμως δε γνωρίζουμε ούτε τις διευθύνσεις τους, αλλά μόνο κάποιο τρόπο υπολογισμού τους.
- Για παράδειγμα, το κ στοιχείο ενός μονοδιάστατου πίνακα θα είναι στη διεύθυνση (*Διεύθυνση αρχής πίνακα* $+(\kappa - 1)$).

Διευθυνσιοδότηση

- Πέραν του γενικευμένου κωδικού λειτουργίας το opcode μιας εντολής περιέχει και την εξειδίκευση του τρόπου προσπέλασης των εντέλων.
- Συνήθως δε γνωρίζουμε τις τιμές των εντέλων που θα χρησιμοποιηθούν σε κάποια εντολή, καθώς στις περισσότερες περιπτώσεις έχουν προκύψει ως ενδιάμεσα αποτελέσματα προηγούμενων υπολογισμών.
- Κάποιες φορές γνωρίζουμε τις διευθύνσεις τους.
- Αρκετές φορές όμως δε γνωρίζουμε ούτε τις διευθύνσεις τους, αλλά μόνο κάποιο τρόπο υπολογισμού τους.
- Για παράδειγμα, το κ στοιχείο ενός μονοδιάστατου πίνακα θα είναι στη διεύθυνση (*Διεύθυνση αρχής πίνακα* $+(\kappa - 1)$).
- Τα υπολογιστικά συστήματα προσφέρουν πολλούς εναλλακτικούς τρόπους προσδιορισμού της τελικής διεύθυνσης του εντέλου που θα χρησιμοποιηθεί. Αυτοί ονομάζονται τρόποι διευθυνσιοδότησης ή τρόποι αναφοράς στη μνήμη.

Διευθυνσιοδότηση

- Πέραν του γενικευμένου κωδικού λειτουργίας το opcode μιας εντολής περιέχει και την εξειδίκευση του τρόπου προσπέλασης των εντέλων.
- Συνήθως δε γνωρίζουμε τις τιμές των εντέλων που θα χρησιμοποιηθούν σε κάποια εντολή, καθώς στις περισσότερες περιπτώσεις έχουν προκύψει ως ενδιάμεσα αποτελέσματα προηγούμενων υπολογισμών.
- Κάποιες φορές γνωρίζουμε τις διευθύνσεις τους.
- Αρκετές φορές όμως δε γνωρίζουμε ούτε τις διευθύνσεις τους, αλλά μόνο κάποιο τρόπο υπολογισμού τους.
- Για παράδειγμα, το κ στοιχείο ενός μονοδιάστατου πίνακα θα είναι στη διεύθυνση (*Διεύθυνση αρχής πίνακα* $+(\kappa - 1)$).
- Τα υπολογιστικά συστήματα προσφέρουν πολλούς εναλλακτικούς τρόπους προσδιορισμού της τελικής διεύθυνσης του εντέλου που θα χρησιμοποιηθεί. Αυτοί ονομάζονται τρόποι διευθυνσιοδότησης ή τρόποι αναφοράς στη μνήμη.
- Σε κάθε εντολή μόνο ένα έντελο μπορεί να έχει “περίεργο” τρόπο διευθυνσιοδότησης. Ολα τα υπόλοιπα έντελα είναι διευθύνσεις καταχωρητών. Το αποτέλεσμα κάποιας εντολής αποθηκεύεται πάντα σε καταχωρητή (πλήν μιας εντολής αποθήκευσης στη μνήμη - ST).

Εναλλακτικοί τρόποι διευθυνσιοδότησης

- Άμεσος (Immediate)

Εναλλακτικοί τρόποι διευθυνσιοδότησης

- Άμεσος (Immediate)
- Κατ' ευθείαν (Direct)
  - Με διεύθυνση καταχωρητή
  - Με διεύθυνση μνήμης

Εναλλακτικοί τρόποι διευθυνσιοδότησης

- Άμεσος (Immediate)
- Κατ' ευθείαν (Direct)
  - Με διεύθυνση καταχωρητή
  - Με διεύθυνση μνήμης
- Έμμεσος (Indirect)
  - Μέσω καταχωρητή
  - Μέσω μνήμης

Εναλλακτικοί τρόποι διευθυνσιοδότησης

- Άμεσος (Immediate)
- Κατ' ευθείαν (Direct)
  - Με διεύθυνση καταχωρητή
  - Με διεύθυνση μνήμης
- Έμμεσος (Indirect)
  - Μέσω καταχωρητή
  - Μέσω μνήμης
- Σχετικός (Relative)

Αμεση διευθυνσιοδότηση - Immediate addressing

- Μορφή :

Opcode	Δνση καταχωρητή αποθήκευσης	Δεδομένο

Αμεση διευθυνσιοδότηση - Immediate addressing

- Μορφή :

- Το δεδομένο παρέχεται στην εντολή (συνήθως σε επόμενα bytes από τον κωδικό λειτουργίας).

Αμεση διευθυνσιοδότηση - Immediate addressing

- Μορφή :

- Το δεδομένο παρέχεται στην εντολή (συνήθως σε επόμενα bytes από τον κωδικό λειτουργίας).
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.

Αμεση διευθυνσιοδότηση - Immediate addressing

- Μορφή :

- Το δεδομένο παρέχεται στην εντολή (συνήθως σε επόμενα bytes από τον κωδικό λειτουργίας).
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.
- Δεδομένο αρκετές φορές σε 16-δική αναπαράσταση συμπληρώματος ως προς 2. Π.χ. #80 = 1000 0000 = -128_{10}
- 8085 : ADI F2 (Acc = Acc - 14_{10})

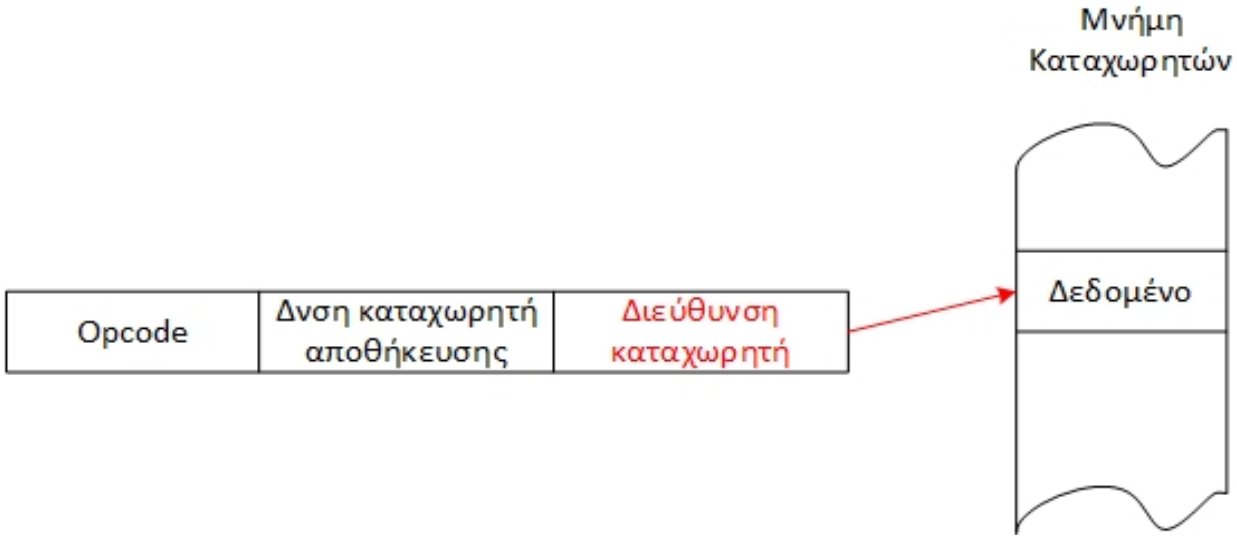
Αμεση διευθυνσιοδότηση - Immediate addressing

- Μορφή :

- Το δεδομένο παρέχεται στην εντολή (συνήθως σε επόμενα bytes από τον κωδικό λειτουργίας).
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.
- Δεδομένο αρκετές φορές σε 16-δική αναπαράσταση συμπληρώματος ως προς 2. Π.χ. #80 = 1000 0000 = -128_{10}
- 8085 : ADI F2 (Acc = Acc - 14_{10})
- AT91 :
  - MOV R3, #23 (R3 = 23_{10})
  - MOV R3, #0x23 (R3 = 35_{10})

Κατ' ευθείαν δ/νσιοδότηση με καταχωρητή - Register Direct addressing

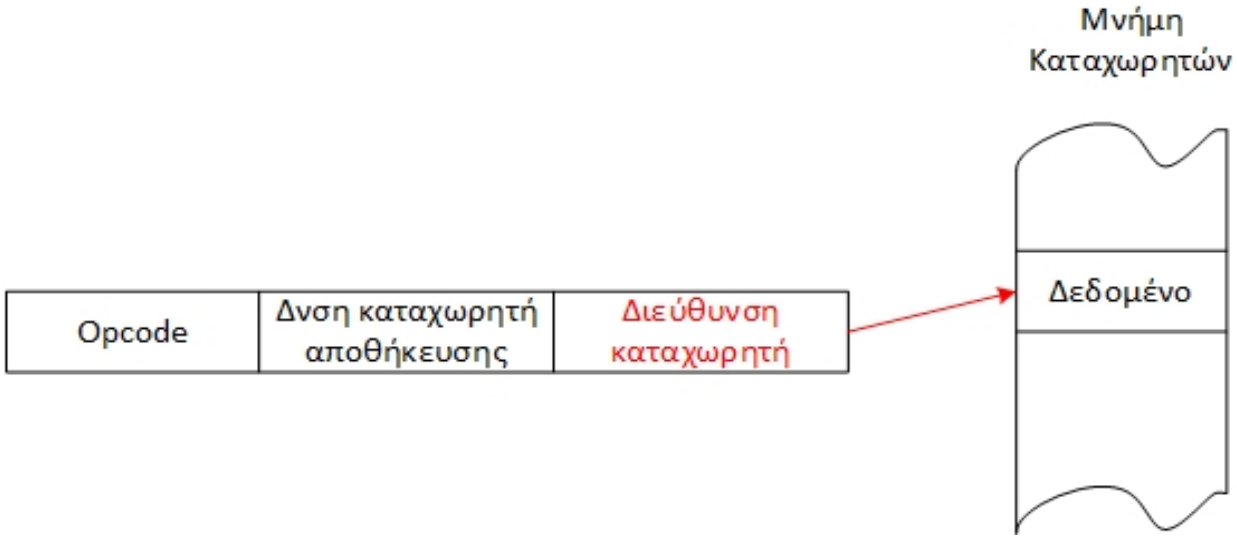
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση καταχωρητή από την οποία θα αντλήσουμε το δεδομένο.

Κατ' ευθείαν δ/νσιοδότηση με καταχωρητή - Register Direct addressing

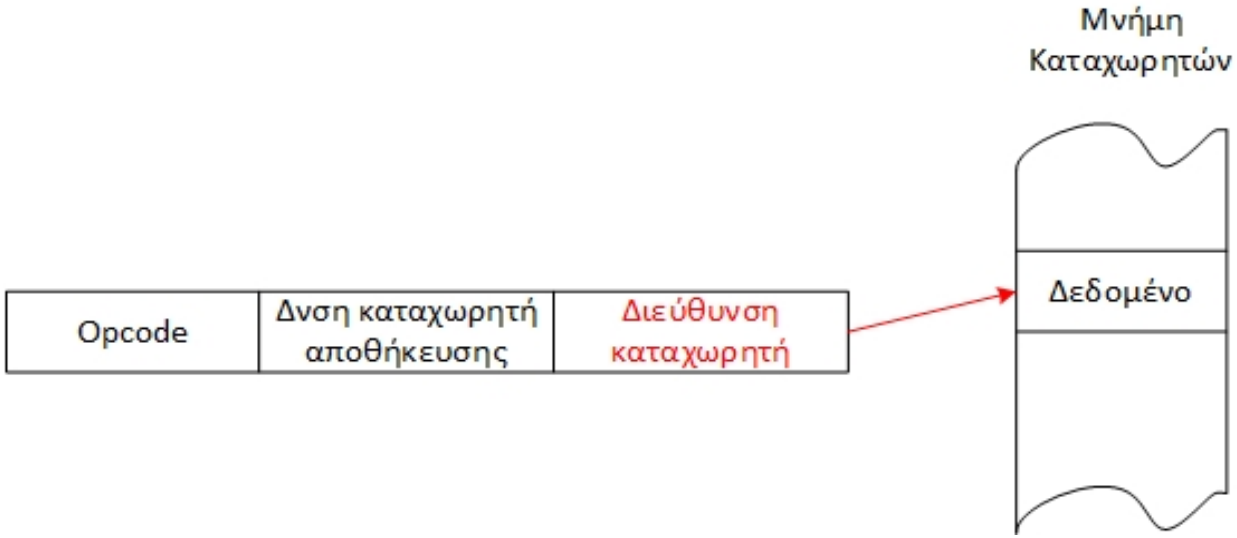
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση καταχωρητή από την οποία θα αντλήσουμε το δεδομένο.
- Καθώς η διεύθυνση καταχωρητή είναι σύντομη, (έως 6 bits) συνήθως μια τέτοια εντολή καταλαμβάνει 1 θέση μνήμης.

Κατ' ευθείαν δ/νσιοδότηση με καταχωρητή - Register Direct addressing

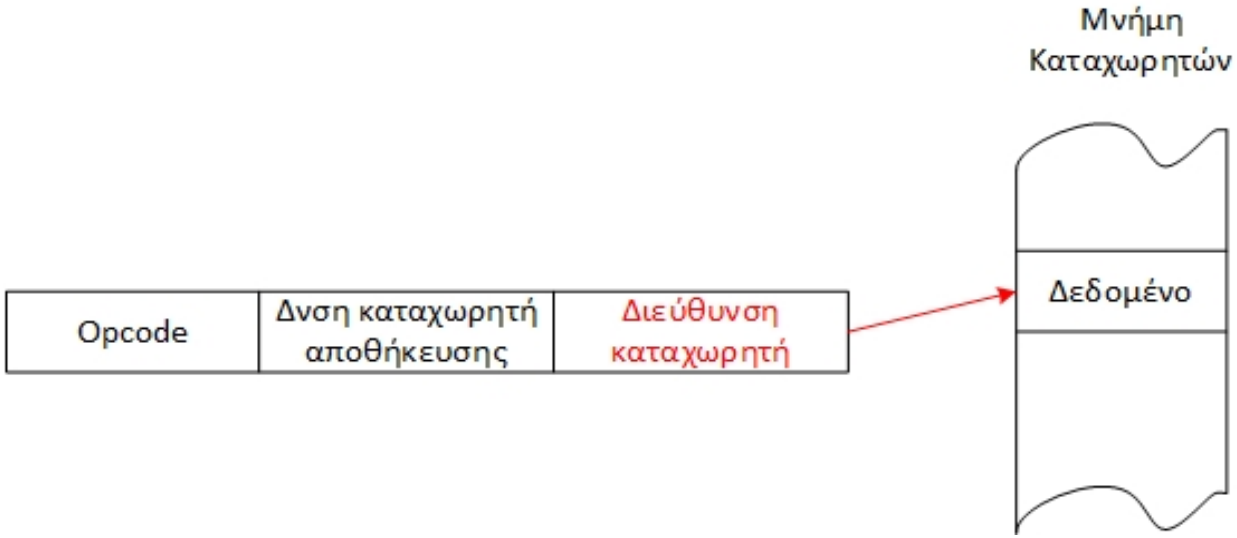
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση καταχωρητή από την οποία θα αντλήσουμε το δεδομένο.
- Καθώς η διεύθυνση καταχωρητή είναι σύντομη, (έως 6 bits) συνήθως μια τέτοια εντολή καταλαμβάνει 1 θέση μνήμης.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.

Κατ' ευθείαν δ/νσιοδότηση με καταχωρητή - Register Direct addressing

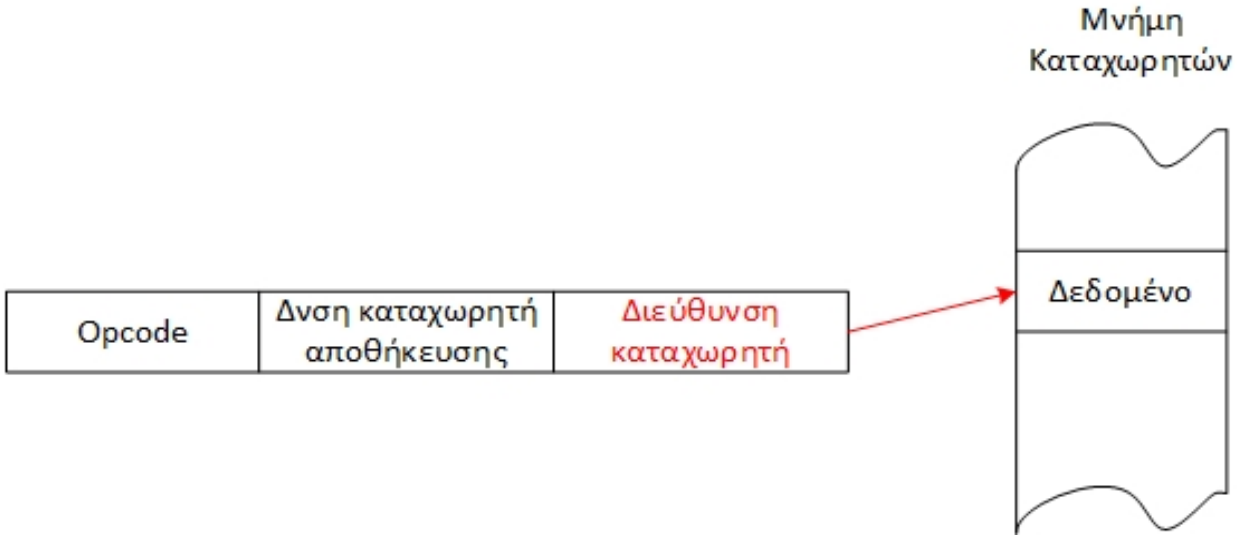
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση καταχωρητή από την οποία θα αντλήσουμε το δεδομένο.
- Καθώς η διεύθυνση καταχωρητή είναι σύντομη, (έως 6 bits) συνήθως μια τέτοια εντολή καταλαμβάνει 1 θέση μνήμης.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.
  - 8085 : ADD B (Acc = Acc + B)

Κατ' ευθείαν δ/νσιοδότηση με καταχωρητή - Register Direct addressing

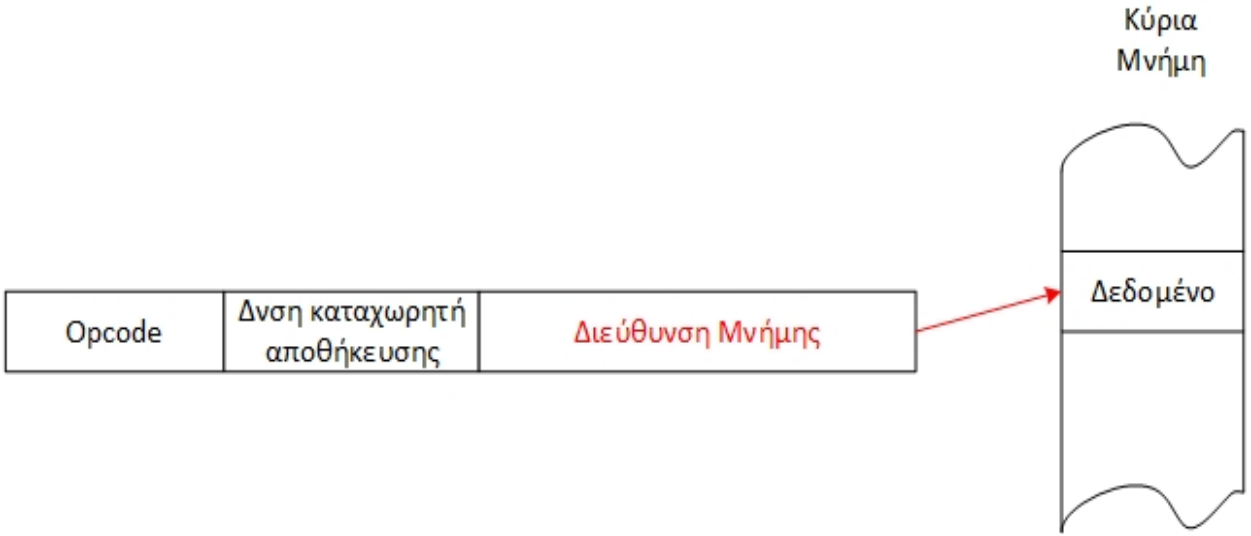
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση καταχωρητή από την οποία θα αντλήσουμε το δεδομένο.
- Καθώς η διεύθυνση καταχωρητή είναι σύντομη, (έως 6 bits) συνήθως μια τέτοια εντολή καταλαμβάνει 1 θέση μνήμης.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.
  - 8085 : ADD B (Acc = Acc + B)
  - AT91 : MOV R8, R4 (R8 = R4)

Κατ' ευθείαν δ/νσιοδότηση με κύρια μνήμη - Memory Direct addressing

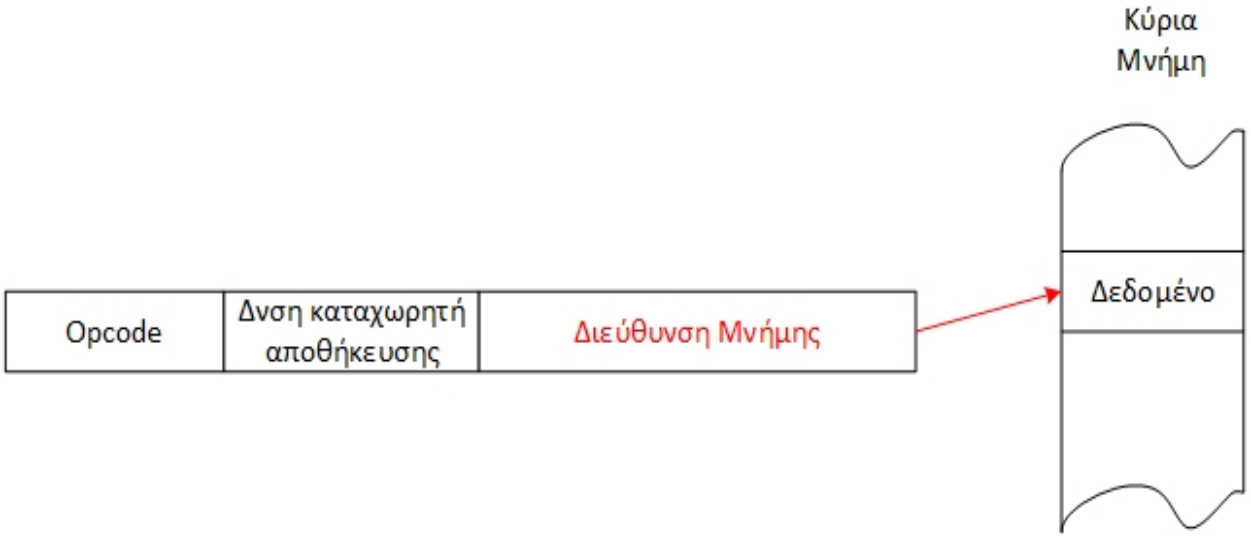
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση κύριας μνήμης από την οποία θα αντλήσουμε το δεδομένο.

Κατ' ευθείαν δ/νσιοδότηση με κύρια μνήμη - Memory Direct addressing

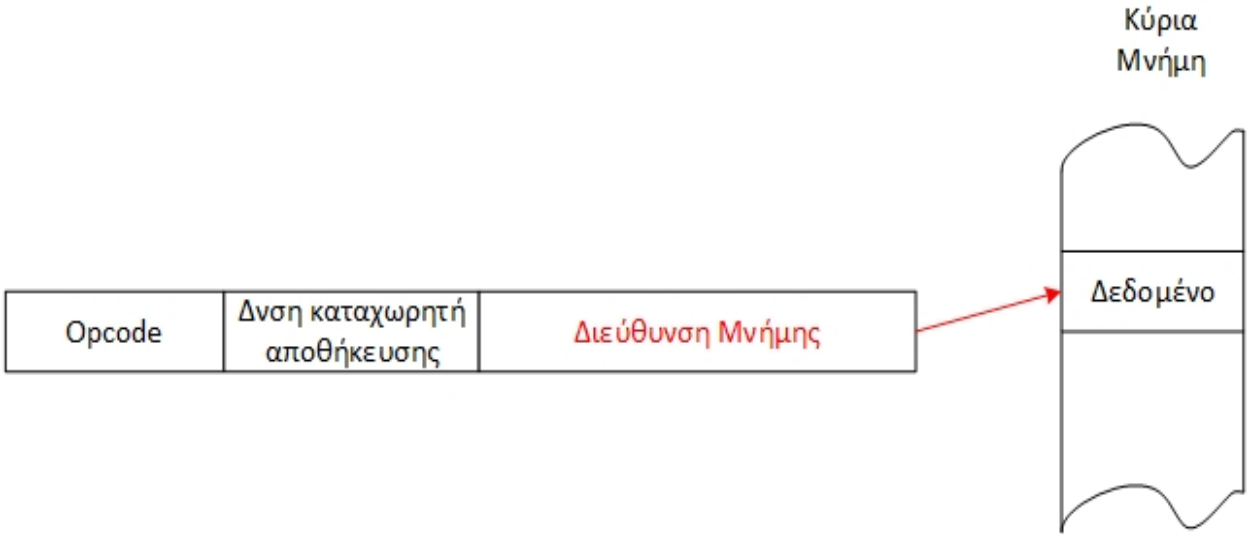
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση κύριας μνήμης από την οποία θα αντλήσουμε το δεδομένο.
- Καθώς η διεύθυνση μνήμης είναι πλατιά, (32 ή 64 bits) συνήθως μια τέτοια εντολή καταλαμβάνει πολλές θέσεις μνήμης.

Κατ' ευθείαν δ/νσιοδότηση με κύρια μνήμη - Memory Direct addressing

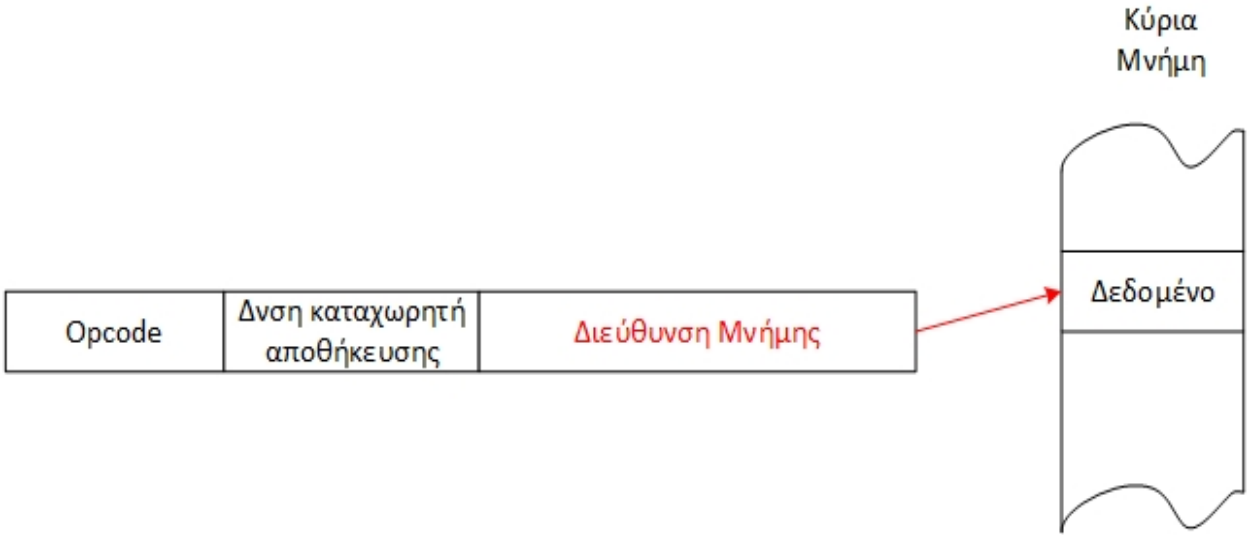
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση κύριας μνήμης από την οποία θα αντλήσουμε το δεδομένο.
- Καθώς η διεύθυνση μνήμης είναι πλατιά, (32 ή 64 bits) συνήθως μια τέτοια εντολή καταλαμβάνει πολλές θέσεις μνήμης.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.

Κατ' ευθείαν δ/νσιοδότηση με κύρια μνήμη - Memory Direct addressing

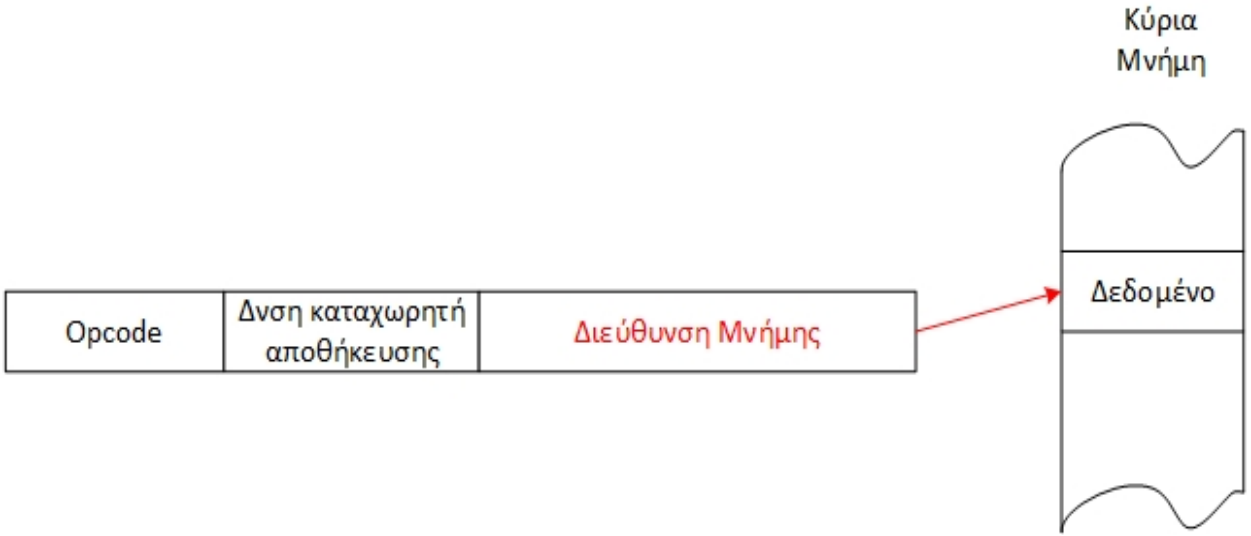
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση κύριας μνήμης από την οποία θα αντλήσουμε το δεδομένο.
- Καθώς η διεύθυνση μνήμης είναι πλατιά, (32 ή 64 bits) συνήθως μια τέτοια εντολή καταλαμβάνει πολλές θέσεις μνήμης.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.
  - 8085 : LDA 3FF0 (Acc = mem(3FF0))

Κατ' ευθείαν δ/νσιοδότηση με κύρια μνήμη - Memory Direct addressing

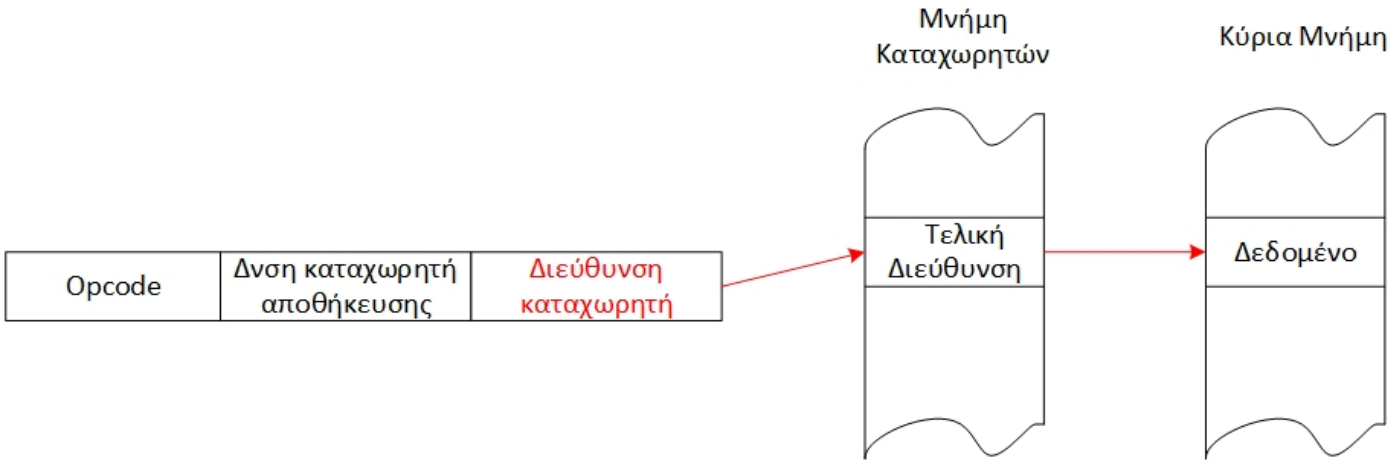
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση κύριας μνήμης από την οποία θα αντλήσουμε το δεδομένο.
- Καθώς η διεύθυνση μνήμης είναι πλατιά, (32 ή 64 bits) συνήθως μια τέτοια εντολή καταλαμβάνει πολλές θέσεις μνήμης.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.
  - 8085 : LDA 3FF0 (Acc = mem(3FF0))
  - AT91 : LDR R4, 0x32F4BABC (R4 = mem(32F4BABC))

Εμμεση δ/νσιοδότηση με καταχωρητή - Register indirect addressing

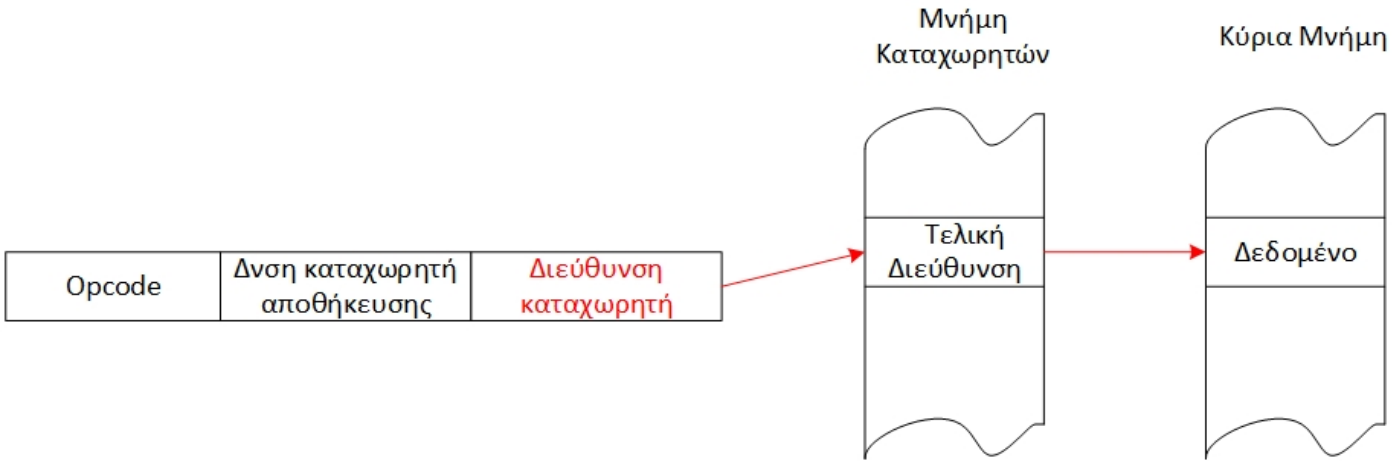
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση ενός καταχωρητή στην οποία θα βρούμε τη τελική (effective) διεύθυνση από την οποία θα πάρουμε το δεδομένο.

Εμμεση δ/νσιοδότηση με καταχωρητή - Register indirect addressing

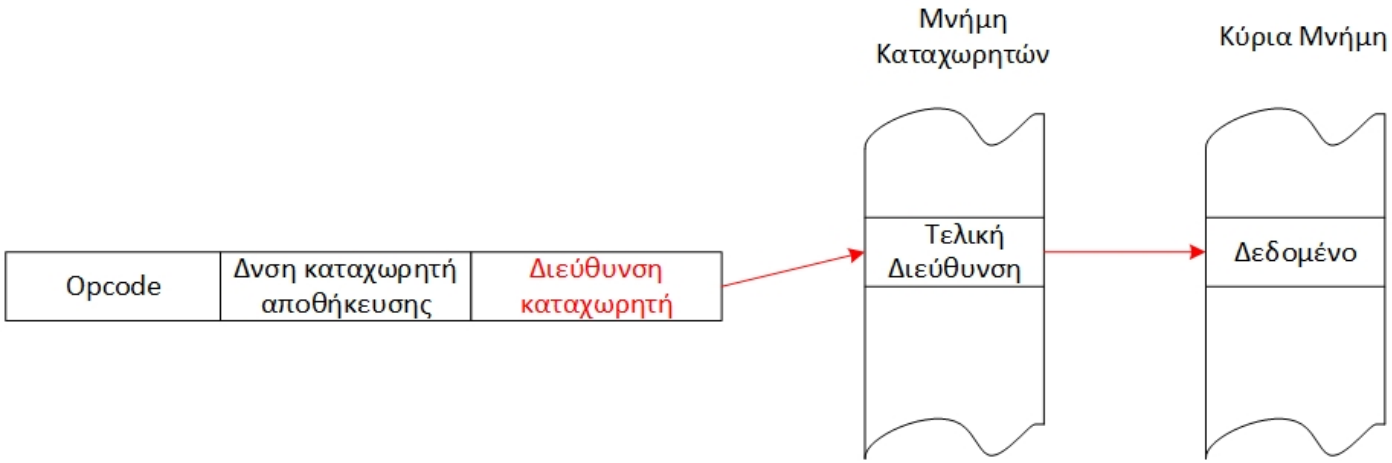
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση ενός καταχωρητή στην οποία θα βρούμε τη τελική (effective) διεύθυνση από την οποία θα πάρουμε το δεδομένο.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.

Εμμεση δ/νσιοδότηση με καταχωρητή - Register indirect addressing

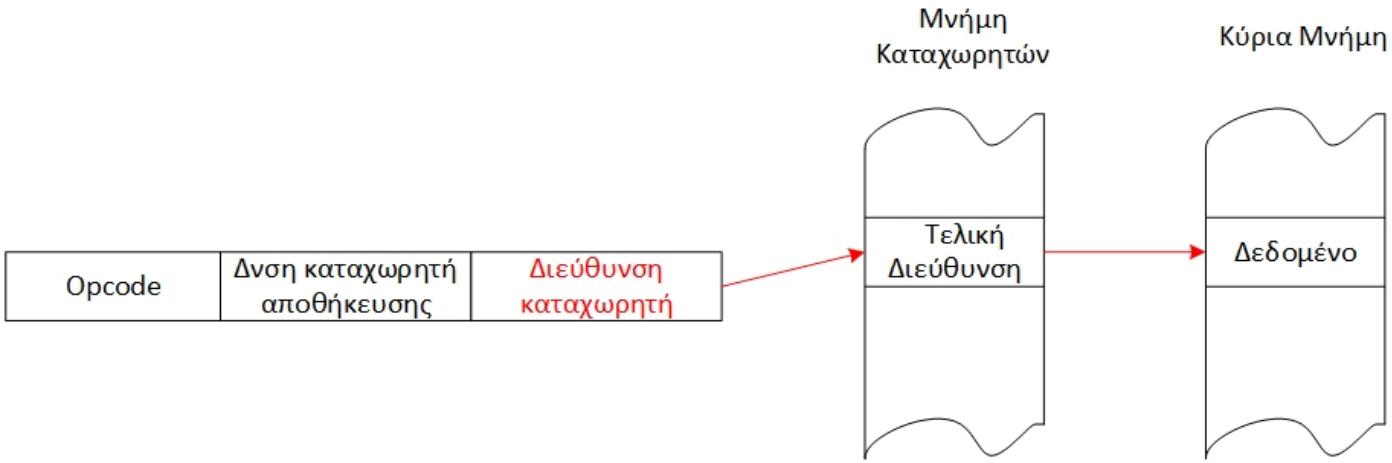
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση ενός καταχωρητή στην οποία θα βρούμε τη τελική (effective) διεύθυνση από την οποία θα πάρουμε το δεδομένο.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.
- 8085 : STAX B (mem([B][C])= Acc)

Εμμεση δ/νσιοδότηση με καταχωρητή - Register indirect addressing

- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση ενός καταχωρητή στην οποία θα βρούμε τη τελική (effective) διεύθυνση από την οποία θα πάρουμε το δεδομένο.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.
- 8085 : STAX B (mem([B][C])= Acc)
- AT91 : LDR R0, [R1] (R0 = mem([R1]))

Εμμεση δ/νσιοδότηση με μνήμη - Memory indirect addressing

- Μορφή :

- Στην εντολή παρέχεται η διεύθυνση μνήμης στην οποία θα βρούμε τη τελική (effective) διεύθυνση από την οποία θα πάρουμε το δεδομένο.

Εμμεση δ/νσιοδότηση με μνήμη - Memory indirect addressing

- Μορφή :

- Στην εντολή παρέχεται η διεύθυνση μνήμης στην οποία θα βρούμε τη τελική (effective) διεύθυνση από την οποία θα πάρουμε το δεδομένο.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.

Εμμεση δ/νσιοδότηση με μνήμη - Memory indirect addressing

- Μορφή :

- Στην εντολή παρέχεται η διεύθυνση μνήμης στην οποία θα βρούμε τη τελική (effective) διεύθυνση από την οποία θα πάρουμε το δεδομένο.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.
- Η διπλή προσπέλαση μνήμης καθιστά αυτές τις εντολές πολύ δαπανηρές σε κύκλους.
- 8085 : Δεν υποστηρίζεται
- AT91 : Δεν υποστηρίζεται

Αυτόματη αύξηση στις έμμεσες αναφορές

- Το ενδιάμεσο στοιχείο στις έμμεσες αναφορές παίζει το ρόλο του δείκτη της τελικής διεύθυνσης.

Αυτόματη αύξηση στις έμμεσες αναφορές

- Το ενδιάμεσο στοιχείο στις έμμεσες αναφορές παίζει το ρόλο του δείκτη της τελικής διεύθυνσης.
- Υποθέστε ένα μονοδιάστατο πίνακα, τα στοιχεία του οποίου ξεκινάνε στη διεύθυνση A400. Αποθηκεύοντας το A400 στον R, μπορούμε να προσπελαύνουμε το κ στοιχείο του πίνακα κάνοντας έμμεση αναφορά με τον R αφού πρώτα του προσθέσουμε το $\kappa - 1$.

Αυτόματη αύξηση στις έμμεσες αναφορές

- Το ενδιάμεσο στοιχείο στις έμμεσες αναφορές παίζει το ρόλο του δείκτη της τελικής διεύθυνσης.
- Υποθέστε ένα μονοδιάστατο πίνακα, τα στοιχεία του οποίου ξεκινάνε στη διεύθυνση A400. Αποθηκεύοντας το A400 στον R, μπορούμε να προσπελαύνουμε το κ στοιχείο του πίνακα κάνοντας έμμεση αναφορά με τον R αφού πρώτα του προσθέσουμε το $\kappa - 1$.
- Σε πολλές περιπτώσεις χρειάζεται να χρησιμοποιήσουμε ένα ένα όλα τα στοιχεία του πίνακα με τη σειρά.

Αυτόματη αύξηση στις έμμεσες αναφορές

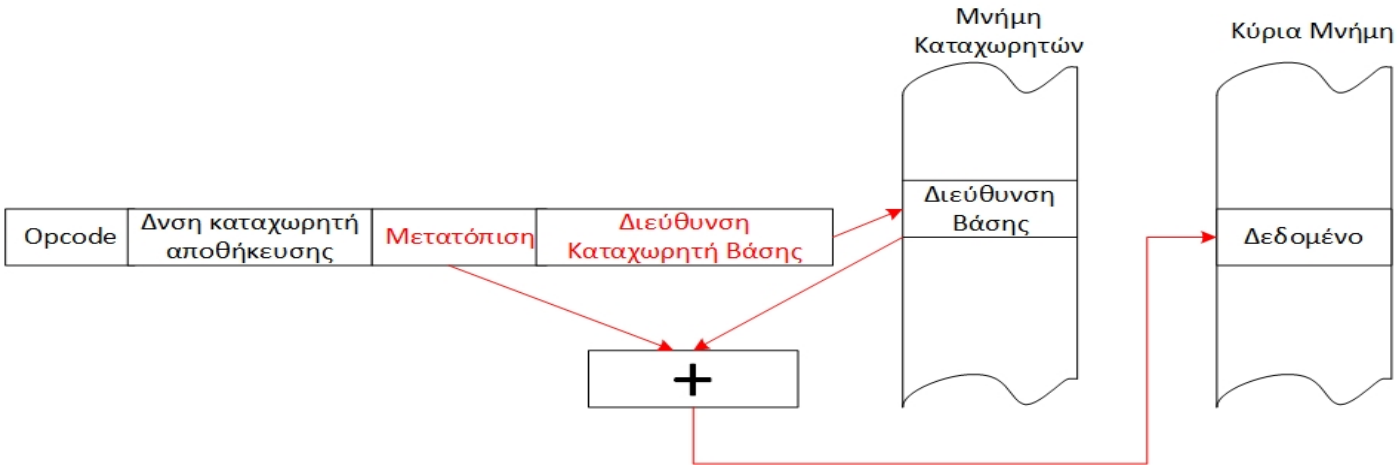
- Το ενδιάμεσο στοιχείο στις έμμεσες αναφορές παίζει το ρόλο του δείκτη της τελικής διεύθυνσης.
- Υποθέστε ένα μονοδιάστατο πίνακα, τα στοιχεία του οποίου ξεκινάνε στη διεύθυνση A400. Αποθηκεύοντας το A400 στον R, μπορούμε να προσπελαύνουμε το κ στοιχείο του πίνακα κάνοντας έμμεση αναφορά με τον R αφού πρώτα του προσθέσουμε το $\kappa - 1$.
- Σε πολλές περιπτώσεις χρειάζεται να χρησιμοποιήσουμε ένα ένα όλα τα στοιχεία του πίνακα με τη σειρά.
- Για το σκοπό αυτό $\exists$ έμμεσες αναφορές με αυτόματη αύξηση του περιεχομένου του καταχωρητή.

Αυτόματη αύξηση στις έμμεσες αναφορές

- Το ενδιάμεσο στοιχείο στις έμμεσες αναφορές παίζει το ρόλο του δείκτη της τελικής διεύθυνσης.
- Υποθέστε ένα μονοδιάστατο πίνακα, τα στοιχεία του οποίου ξεκινάνε στη διεύθυνση A400. Αποθηκεύοντας το A400 στον R, μπορούμε να προσπελαύνουμε το κ στοιχείο του πίνακα κάνοντας έμμεση αναφορά με τον R αφού πρώτα του προσθέσουμε το $\kappa - 1$.
- Σε πολλές περιπτώσεις χρειάζεται να χρησιμοποιήσουμε ένα ένα όλα τα στοιχεία του πίνακα με τη σειρά.
- Για το σκοπό αυτό $\exists$ έμμεσες αναφορές με αυτόματη αύξηση του περιεχομένου του καταχωρητή.
- Η αύξηση μπορεί να προηγείται της προσπέλασης (pre-increment) ή να την ακολουθεί (post-increment).
- Η ποσότητα αύξησης είναι ανάλογη του μεγέθους της εντολής. Εντολές byte αυξάνουν κατά 1 το δείκτη. Εντολές 16 bit κατά 2, κοκ.

Σχετική Διευθυνσιοδότηση - Relative addressing

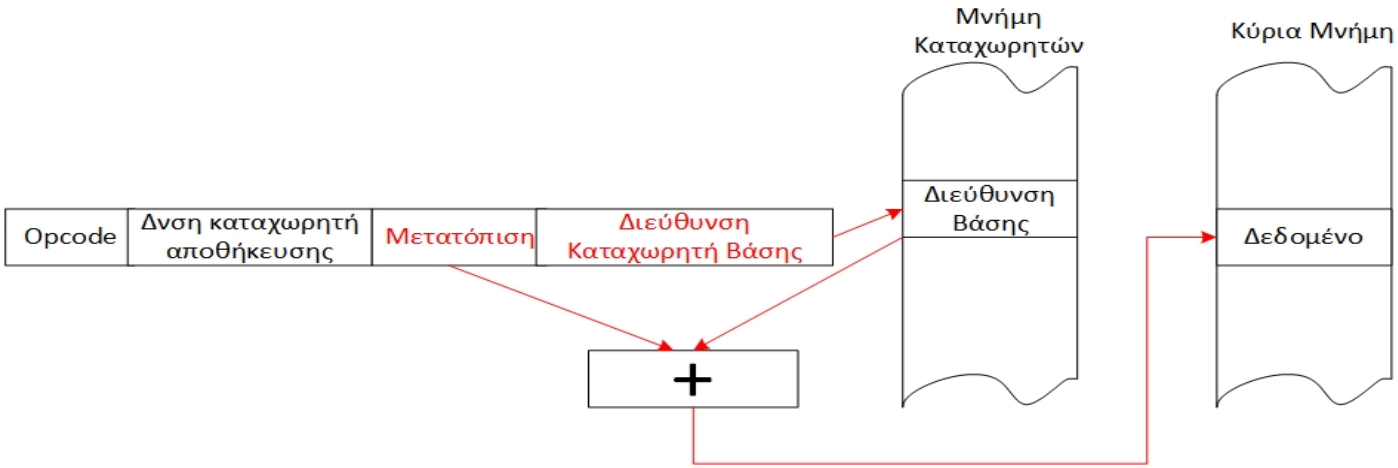
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση ενός ενδιάμεσου στοιχείου μνήμης και μια μετατόπιση (θετική ή αρνητική). Η τελική διεύθυνση προκύπτει μετατοπίζοντας τη διεύθυνση που περιέχεται στο ενδιάμεσο στοιχείο μνήμης τόσες θέσεις όση η μετατόπιση.

Σχετική Διευθυνσιοδότηση - Relative addressing

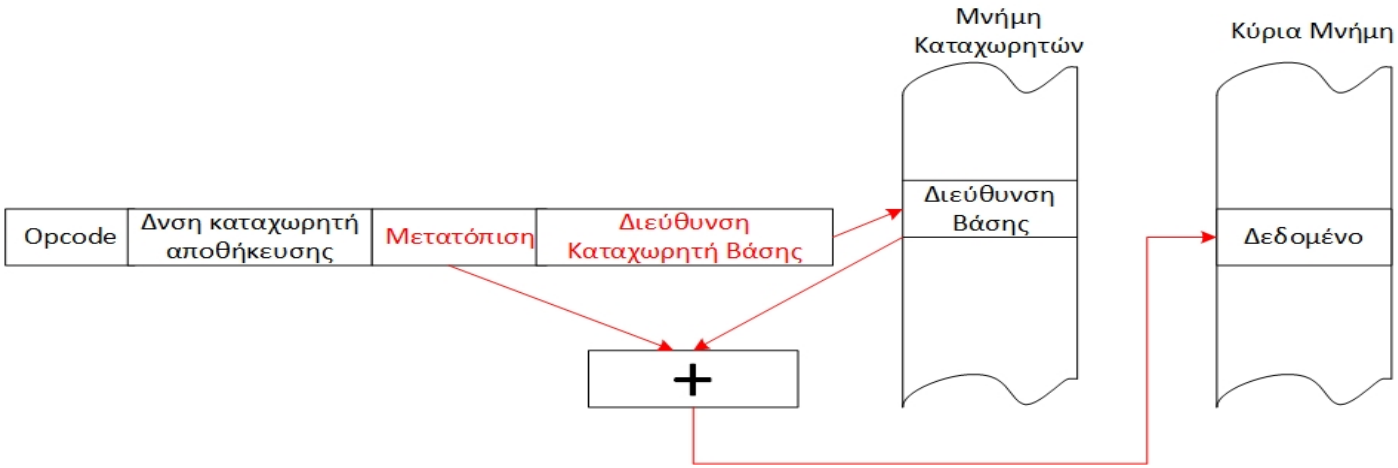
- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση ενός ενδιάμεσου στοιχείου μνήμης και μια μετατόπιση (θετική ή αρνητική). Η τελική διεύθυνση προκύπτει μετατοπίζοντας τη διεύθυνση που περιέχεται στο ενδιάμεσο στοιχείο μνήμης τόσες θέσεις όση η μετατόπιση.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.

Σχετική Διευθυνσιοδότηση - Relative addressing

- Μορφή :



- Στην εντολή παρέχεται η διεύθυνση ενός ενδιάμεσου στοιχείου μνήμης και μια μετατόπιση (θετική ή αρνητική). Η τελική διεύθυνση προκύπτει μετατοπίζοντας τη διεύθυνση που περιέχεται στο ενδιάμεσο στοιχείο μνήμης τόσες θέσεις όση η μετατόπιση.
- Ο καταχωρητής αποθήκευσης μπορεί να υπονοείται.
- 8085 : Δεν υποστηρίζεται
- AT91 : LDR R0, [R1, #4] (R0 = mem([R1]+4))

Βάσει του συνόλου εντολών

- Ανάλογα με το σύνολο εντολών τους, τα υπολογιστικά συστήματα μπορούν να διακριθούν σε αρχιτεκτονικές :
  - μηχανισμού στοίβας (σωρού - stack)

Βάσει του συνόλου εντολών

- Ανάλογα με το σύνολο εντολών τους, τα υπολογιστικά συστήματα μπορούν να διακριθούν σε αρχιτεκτονικές :
  - μηχανισμού στοίβας (σωρού - stack)
  - συσσωρευτή (accumulator based)

Βάσει του συνόλου εντολών

- Ανάλογα με το σύνολο εντολών τους, τα υπολογιστικά συστήματα μπορούν να διακριθούν σε αρχιτεκτονικές :
  - μηχανισμού στοίβας (σωρού - stack)
  - συσσωρευτή (accumulator based)
  - καταχωρητών γενικού σκοπού (register architectures)
  - καταχωρητή – μνήμης

Βάσει του συνόλου εντολών

- Ανάλογα με το σύνολο εντολών τους, τα υπολογιστικά συστήματα μπορούν να διακριθούν σε αρχιτεκτονικές :
  - μηχανισμού στοίβας (σωρού - stack)
  - συσσωρευτή (accumulator based)
  - καταχωρητών γενικού σκοπού (register architectures)
  - καταχωρητή – μνήμης
  - καταχωρητή – καταχωρητή
    - με δύο τελούμενα
    - με τρία τελούμενα

Βάσει του συνόλου εντολών

- Ανάλογα με το σύνολο εντολών τους, τα υπολογιστικά συστήματα μπορούν να διακριθούν σε αρχιτεκτονικές :
  - μηχανισμού στοίβας (σωρού - stack)
  - συσσωρευτή (accumulator based)
  - καταχωρητών γενικού σκοπού (register architectures)
  - καταχωρητή – μνήμης
  - καταχωρητή – καταχωρητή
    - με δύο τελούμενα
    - με τρία τελούμενα
  - καταχωρητή – καταχωρητή & καταχωρητή – μνήμης

Λειτουργία και εντολές της στοίβας

- Η στοίβα δομείται σε μια δεσμευμένη περιοχή της μνήμης.

Λειτουργία και εντολές της στοίβας

- Η στοίβα δομείται σε μια δεσμευμένη περιοχή της μνήμης.
- Προσπελαύνεται μόνο μέσω του καταχωρητή στοίβας (stack pointer - SP)

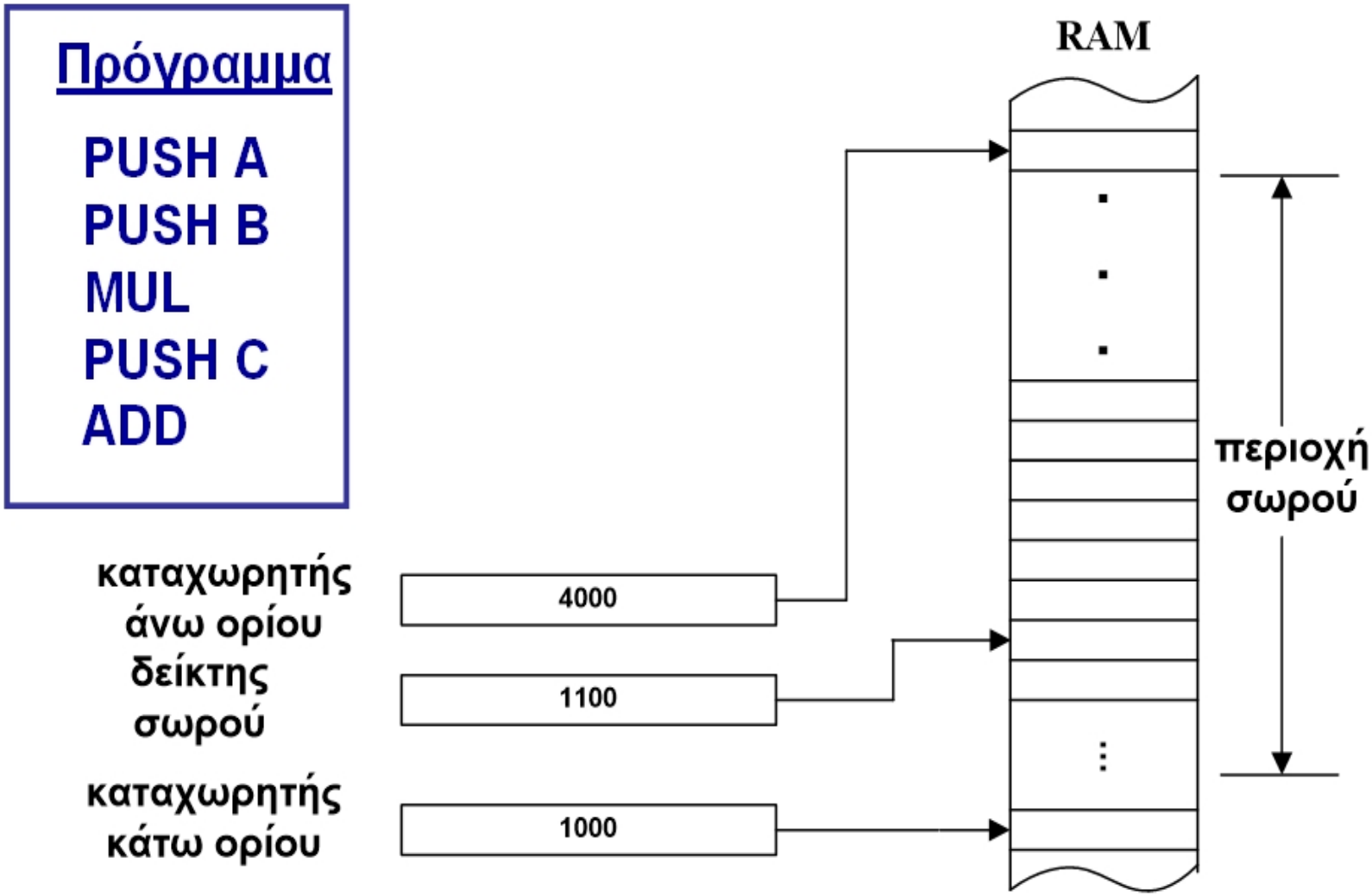
Λειτουργία και εντολές της στοίβας

- Η στοίβα δομείται σε μια δεσμευμένη περιοχή της μνήμης.
- Προσπελαύνεται μόνο μέσω του καταχωρητή στοίβας (stack pointer - SP)
- Τα δεδομένα που απαιτούνται για την εκτέλεση μίας πράξης βρίσκονται στην κορυφή της στοίβας, απ' όπου και μεταφέρονται στην αριθμητική/λογική μονάδα για την εκτέλεση της πράξης. Το αποτέλεσμα της πράξης αποθηκεύεται στην κορυφή της στοίβας.

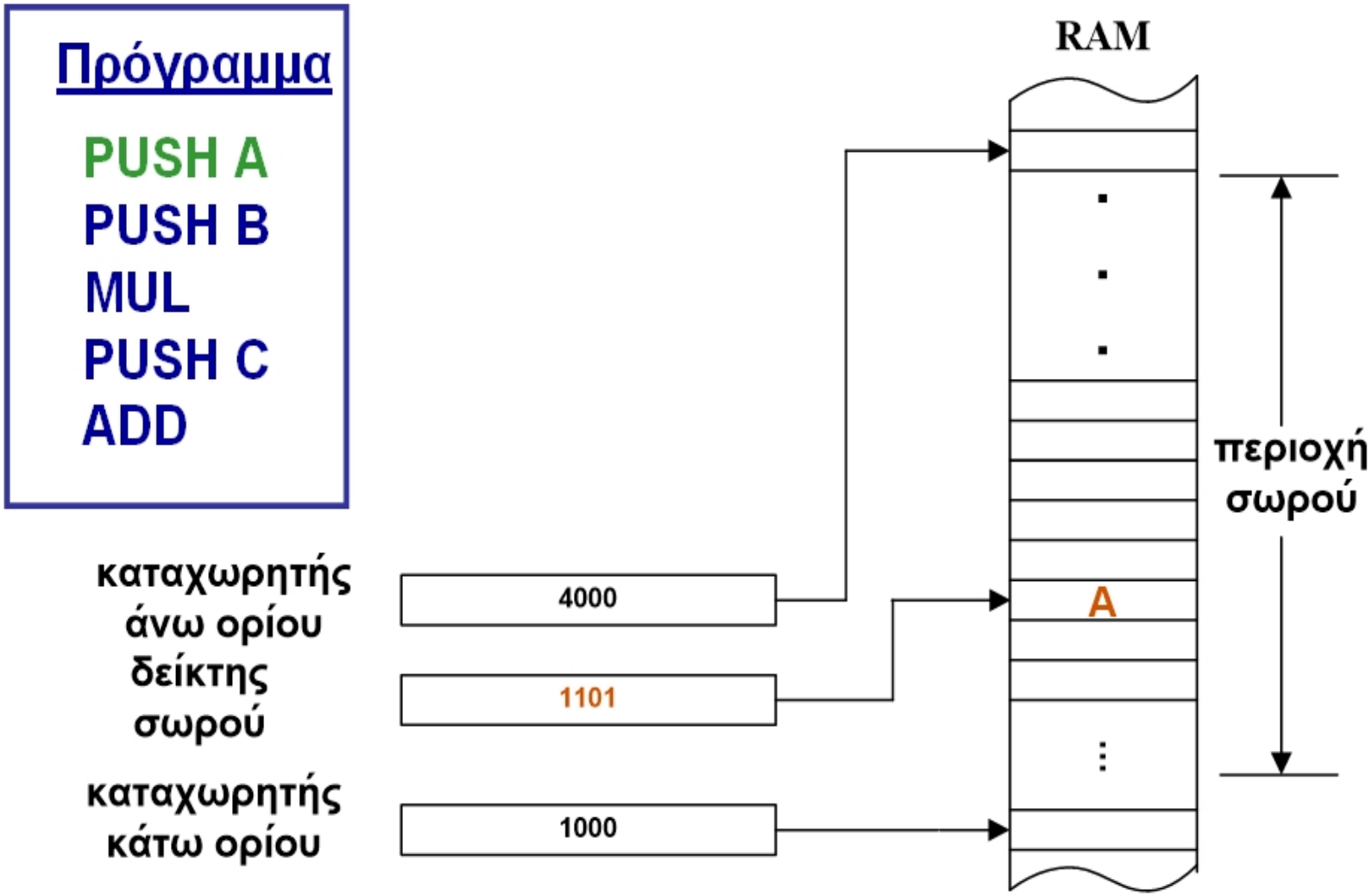
Λειτουργία και εντολές της στοίβας

- Η στοίβα δομείται σε μια δεσμευμένη περιοχή της μνήμης.
- Προσπελαύνεται μόνο μέσω του καταχωρητή στοίβας (stack pointer - SP)
- Τα δεδομένα που απαιτούνται για την εκτέλεση μίας πράξης βρίσκονται στην κορυφή της στοίβας, απ' όπου και μεταφέρονται στην αριθμητική/λογική μονάδα για την εκτέλεση της πράξης. Το αποτέλεσμα της πράξης αποθηκεύεται στην κορυφή της στοίβας.
- Εντολές:
  - PUSH (προώθηση – μεταφορά στη στοίβα)
  - POP (ανάκτηση – μεταφορά από τη στοίβα)
  - ADD/SUB/MUL/DIV (βασικές αριθμητικές πράξεις)

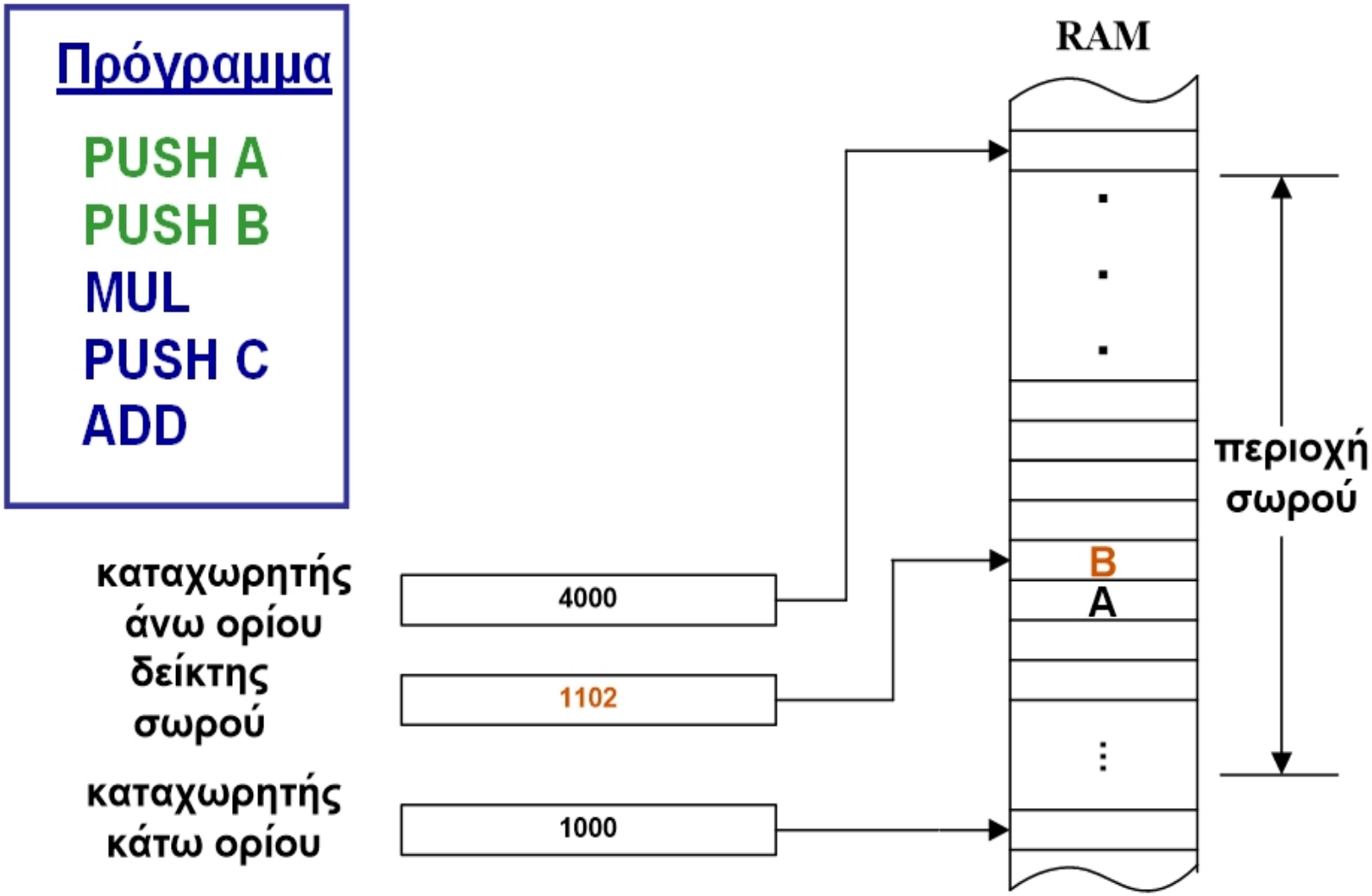
Παράδειγμα χρήσης της στοίβας (1/10)



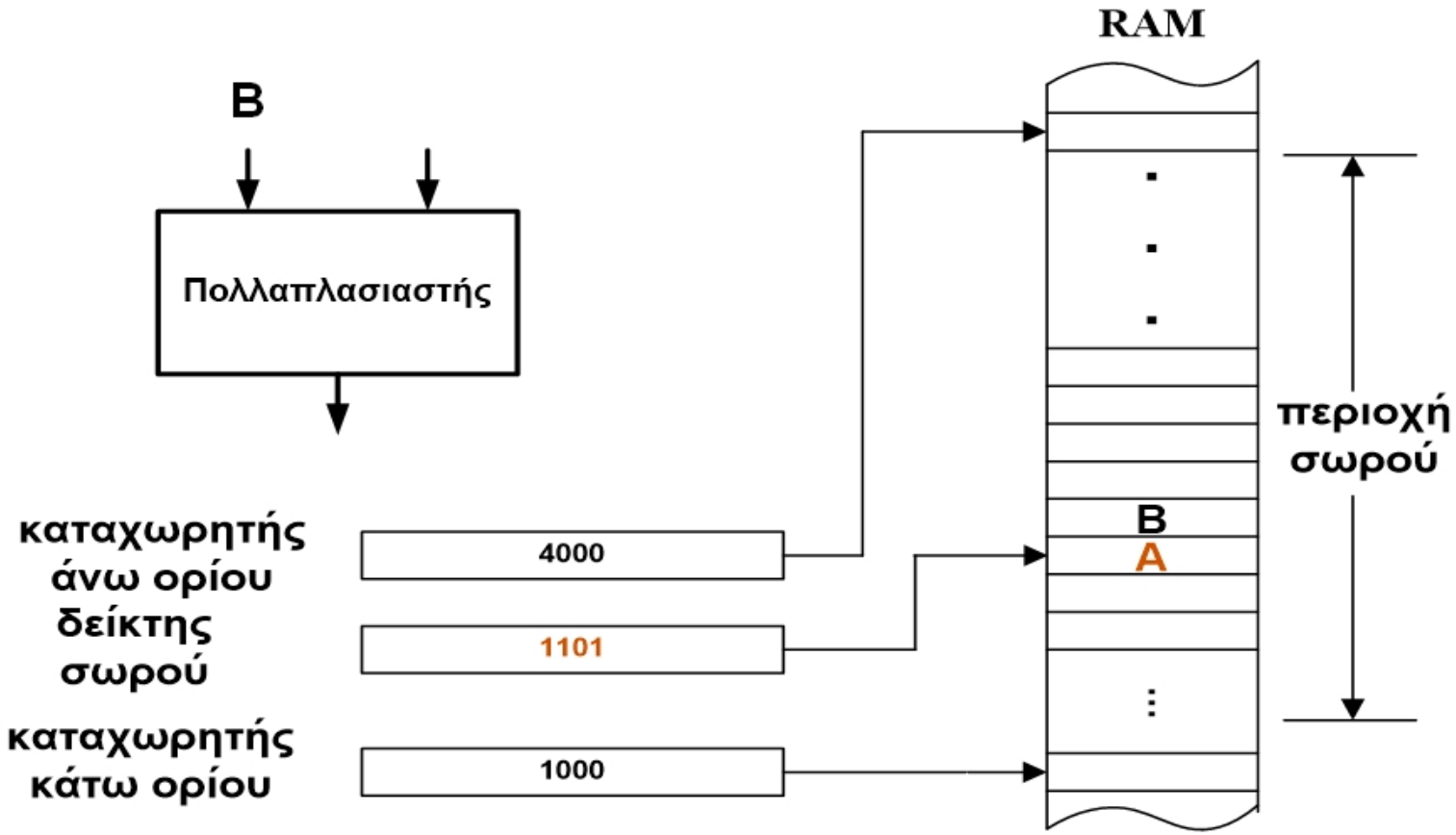
Παράδειγμα χρήσης της στοίβας (2/10)



Παράδειγμα χρήσης της στοίβας (3/10)

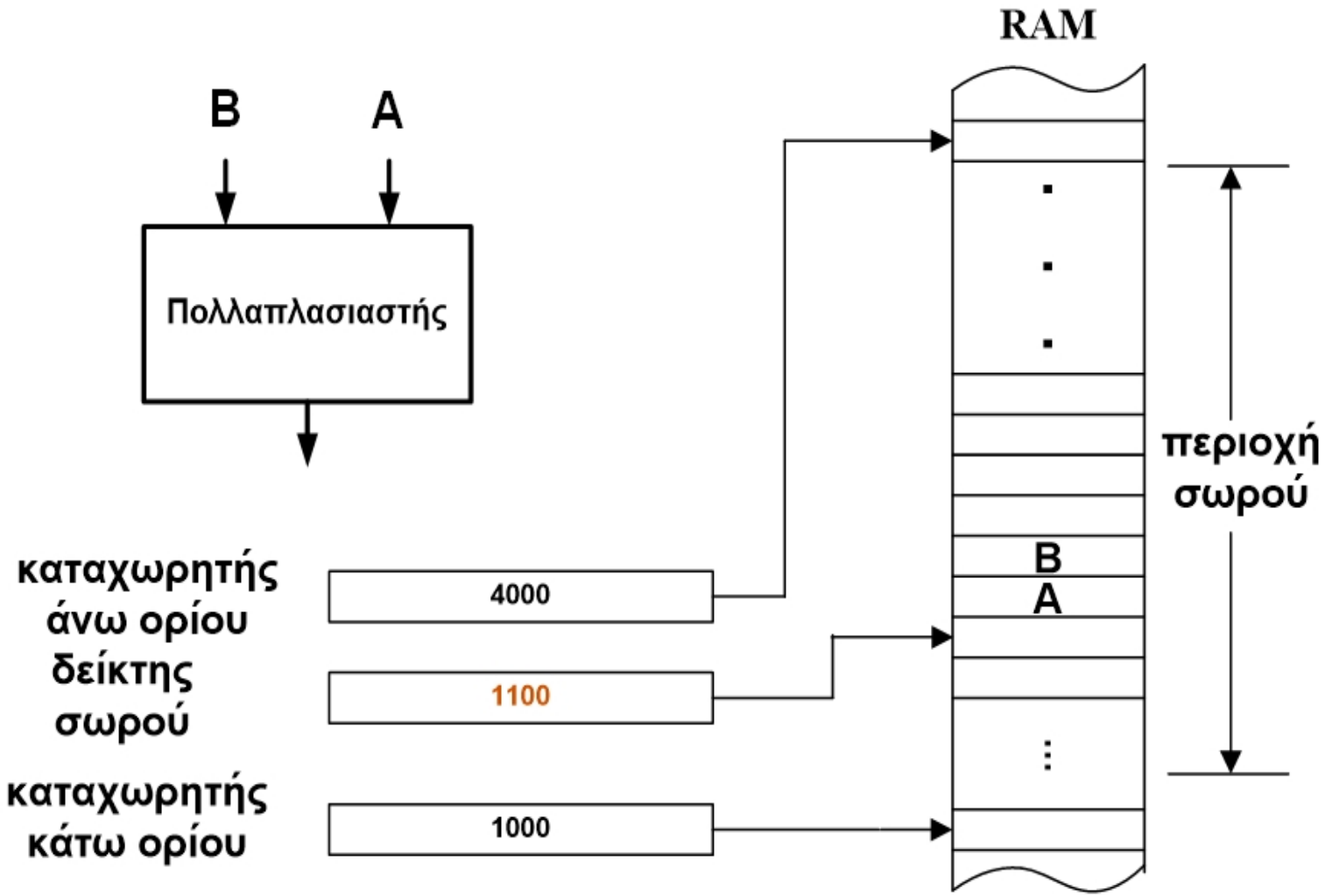


Παράδειγμα χρήσης της στοίβας (4/10)

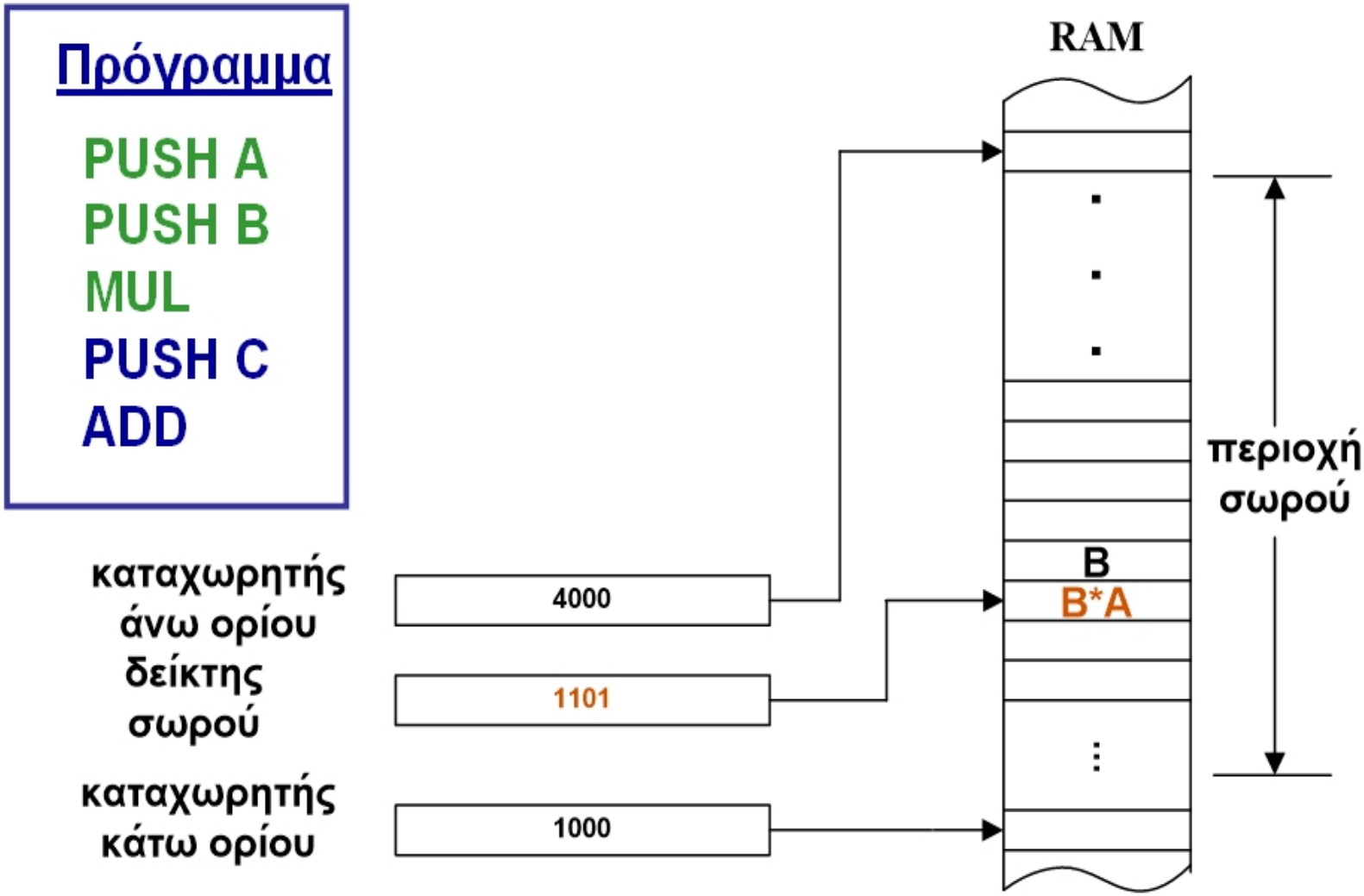


Η εντολή MUL εκτελείται σε τρία βήματα

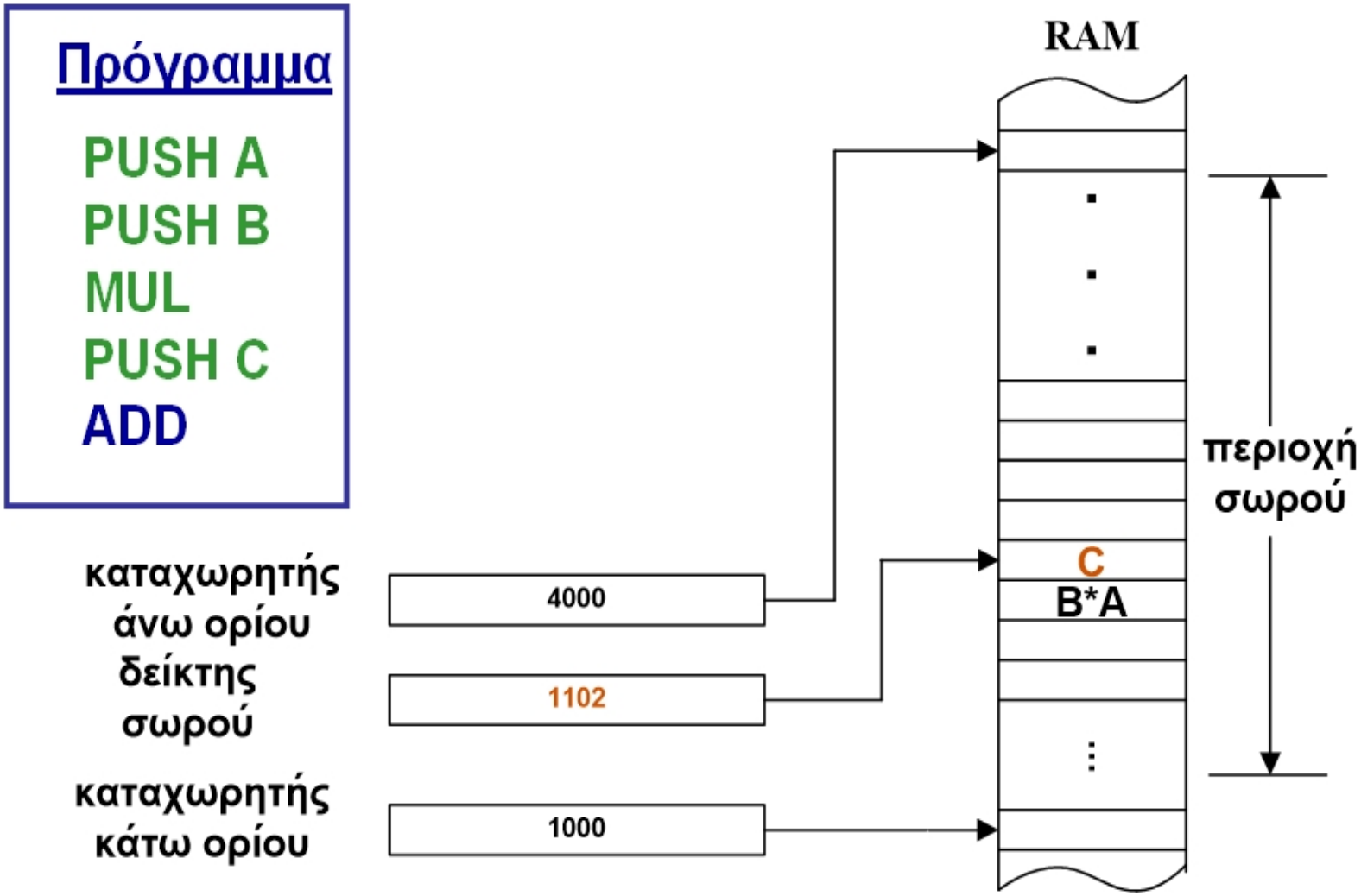
Παράδειγμα χρήσης της στοίβας (5/10)



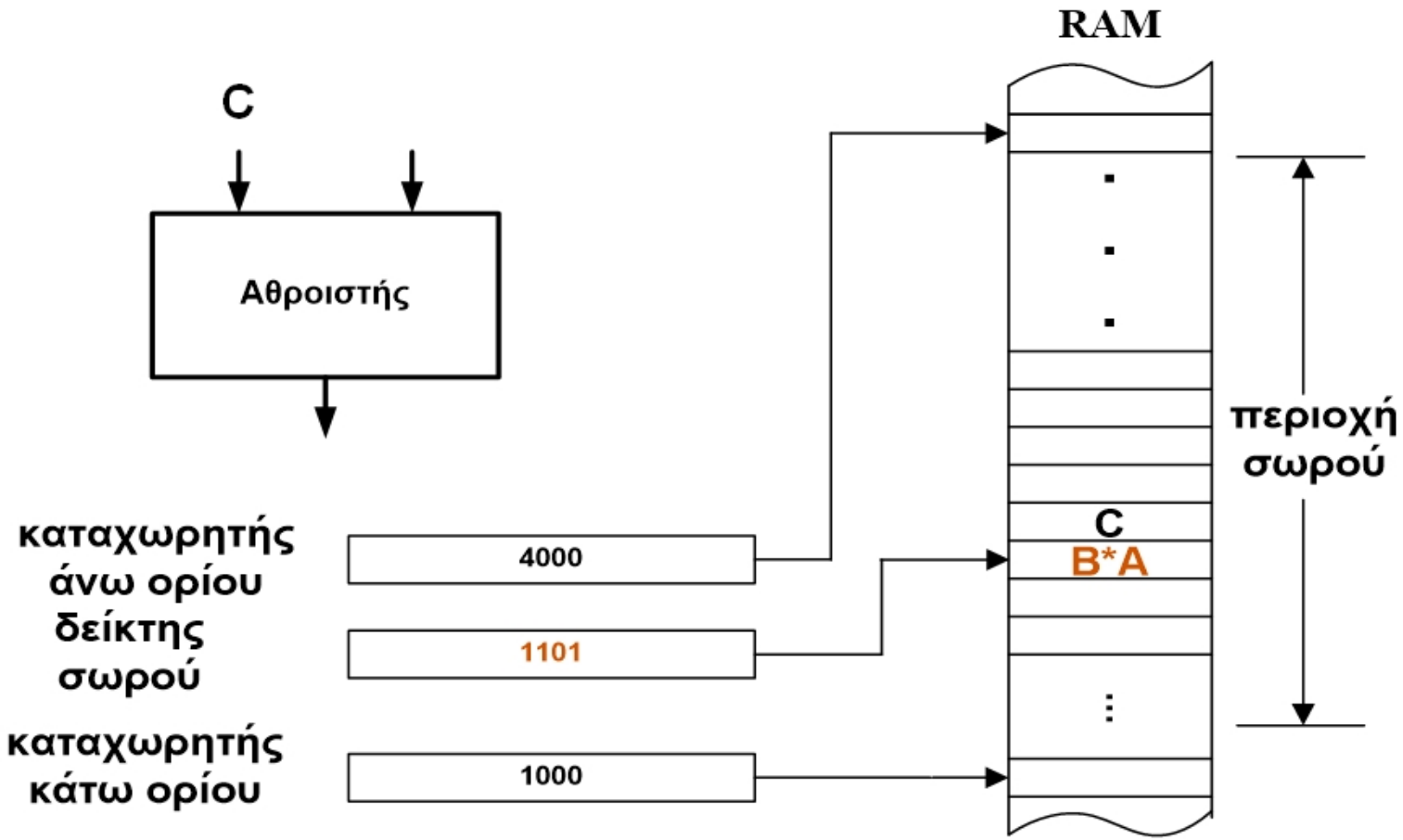
Παράδειγμα χρήσης της στοίβας (6/10)



Παράδειγμα χρήσης της στοίβας (7/10)

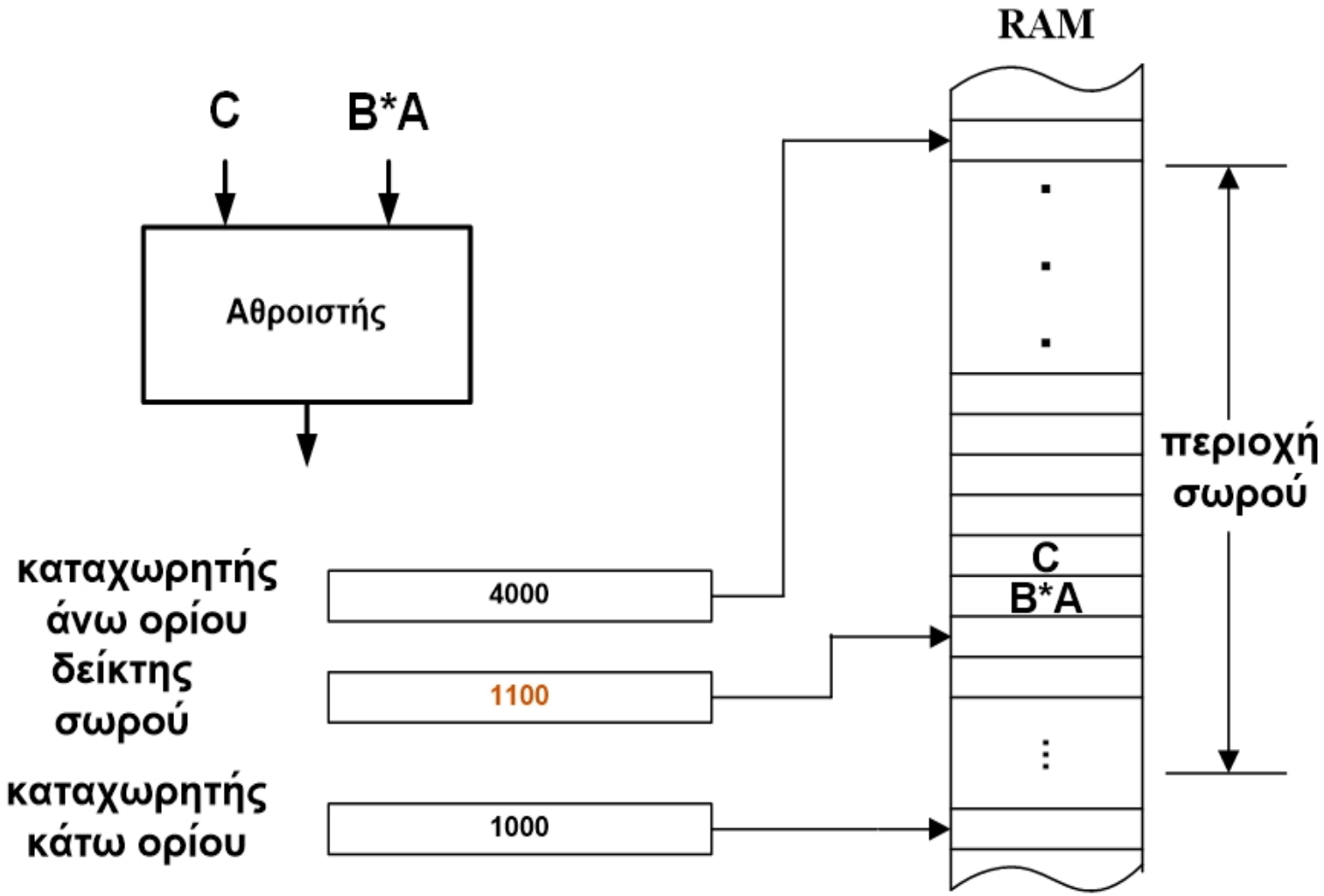


Παράδειγμα χρήσης της στοίβας (8/10)

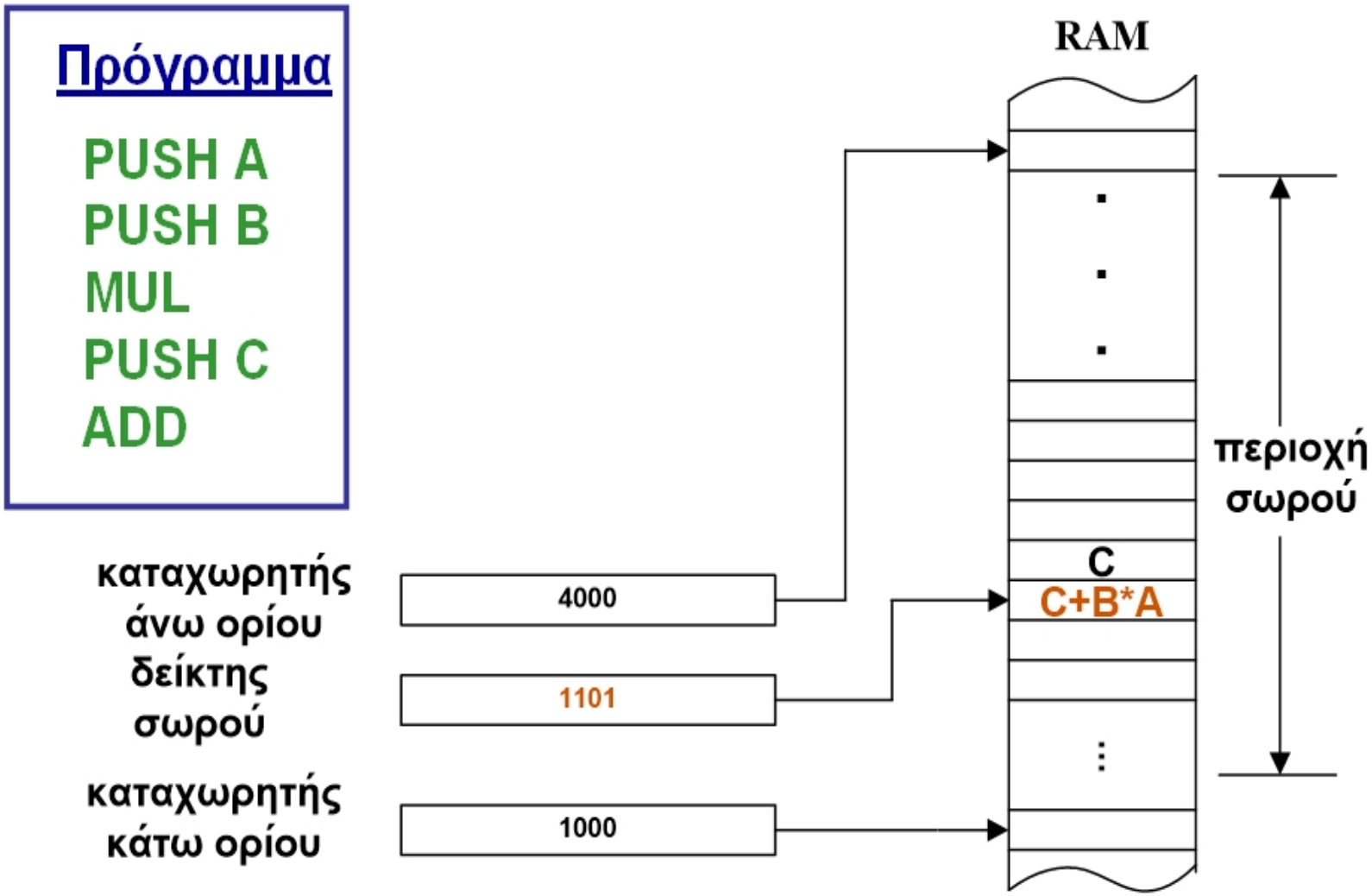


Η εντολή ADD εκτελείται σε τρία βήματα

Παράδειγμα χρήσης της στοίβας (9/10)



Παράδειγμα χρήσης της στοίβας (10/10)

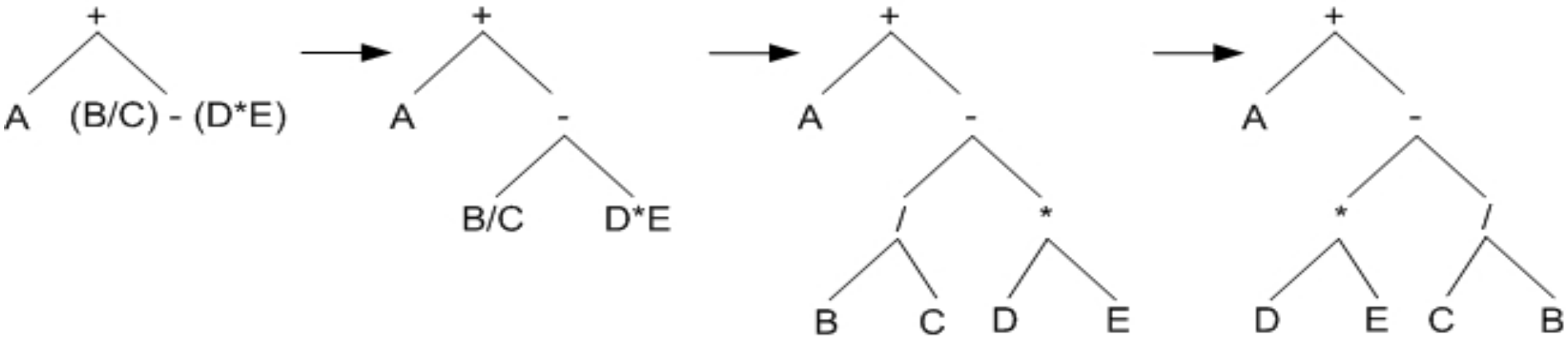


Μετατροπή έκφρασης σε postfix notation

- Εστω ο υπολογισμός της έκφρασης $F = A + B/C - D * E$ ή ισοδύναμα $F = A + [(B/C) - (D * E)]$.

Μετατροπή έκφρασης σε postfix notation

- Εστω ο υπολογισμός της έκφρασης $F = A + B/C - D * E$ ή ισοδύναμα $F = A + [(B/C) - (D * E)]$.
- Για την εκτέλεσή του σε μηχανή στοίβας χρειάζεται να βρω τη postfix μορφή της έκφρασης. Αυτό γίνεται με τη κατασκευή του δέντρου της έκφρασης, την αντιμετάθεση των υποδέντρων των μη αντιμεταθετικών τελεστών και τη μεταδιατεταγμένη (postorder) διαπέραση του.
- Postfix μορφή : ADE*CB/-+



PUSH A	MUL	DIV	POP F
PUSH D	PUSH C	SUB	
PUSH E	PUSH B	ADD	

Μηχανή συσσωρευτή

- Σε κάθε εντολή υπονοείται πάντα ως πρώτο έντελο ή και σαν καταχωρητής αποθήκευσης του αποτελέσματος, ένας συγκεκριμένος καταχωρητής, ο συσσωρευτής.

Μηχανή συσσωρευτή

- Σε κάθε εντολή υπονοείται πάντα ως πρώτο έντελο ή και σαν καταχωρητής αποθήκευσης του αποτελέσματος, ένας συγκεκριμένος καταχωρητής, ο συσσωρευτής.
- $F = A + B/C - D * E$

Μηχανή συσσωρευτή

- Σε κάθε εντολή υπονοείται πάντα ως πρώτο έντελο ή και σαν καταχωρητής αποθήκευσης του αποτελέσματος, ένας συγκεκριμένος καταχωρητής, ο συσσωρευτής.
- $F = A + B/C - D * E$

LOAD	E	DIV	C
MUL	D	SUB	F
STORE	F	ADD	A
LOAD	B	STORE	F

Μηχανή 2 εντέλων με εντολές καταχωρητή - μνήμης

- Εκ των δύο εντέλων το ένα είναι καταχωρητής και το άλλο διεύθυνση μνήμης.
- Το αποτέλεσμα αποθηκεύεται πάντα (με εξαίρεση την εντολή STORE) στον καταχωρητή.

Μηχανή 2 εντέλων με εντολές καταχωρητή - μνήμης

- Εκ των δύο εντέλων το ένα είναι καταχωρητής και το άλλο διεύθυνση μνήμης.
- Το αποτέλεσμα αποθηκεύεται πάντα (με εξαίρεση την εντολή STORE) στον καταχωρητή.
- $F = A + B/C - D * E$

Μηχανή 2 εντέλων με εντολές καταχωρητή - μνήμης

- Εκ των δύο εντέλων το ένα είναι καταχωρητής και το άλλο διεύθυνση μνήμης.
- Το αποτέλεσμα αποθηκεύεται πάντα (με εξαίρεση την εντολή STORE) στον καταχωρητή.
- $F = A + B/C - D * E$

LOAD	R1, D	‖	LOAD	R1, B	‖	SUB	R1, F
MUL	R1, E	‖	DIV	R1, C	‖	STORE	R1, F
STORE	R1, F	‖	ADD	R1, A	‖		

Μηχανή 2 εντέλων, εντολές καταχωρητή-μνήμης & καταχωρητή-καταχωρητή

- Οι εντολές καταχωρητή-μνήμης χρησιμοποιούνται μόνο για την προσκόμμιση δεδομένων στους καταχωρητές και την αποθήκευση αποτελεσμάτων στη μνήμη.
- Όλες οι αριθμητικές και λογικές εντολές είναι μεταξύ καταχωρητών.
- Η αποθήκευση γίνεται πάντα στον μοναδικό καταχωρητή ή στον πρώτο εκ των δύο καταχωρητών που συμμετέχουν στην εντολή.

Μηχανή 2 εντέλων, εντολές καταχωρητή-μνήμης & καταχωρητή-καταχωρητή

- Οι εντολές καταχωρητή-μνήμης χρησιμοποιούνται μόνο για την προσκόμμιση δεδομένων στους καταχωρητές και την αποθήκευση αποτελεσμάτων στη μνήμη.
- Όλες οι αριθμητικές και λογικές εντολές είναι μεταξύ καταχωρητών.
- Η αποθήκευση γίνεται πάντα στον μοναδικό καταχωρητή ή στον πρώτο εκ των δύο καταχωρητών που συμμετέχουν στην εντολή.
- $F = A + B/C - D * E$

Μηχανή 2 εντέλων, εντολές καταχωρητή-μνήμης & καταχωρητή-καταχωρητή

- Οι εντολές καταχωρητή-μνήμης χρησιμοποιούνται μόνο για την προσκόμμιση δεδομένων στους καταχωρητές και την αποθήκευση αποτελεσμάτων στη μνήμη.
- Όλες οι αριθμητικές και λογικές εντολές είναι μεταξύ καταχωρητών.
- Η αποθήκευση γίνεται πάντα στον μοναδικό καταχωρητή ή στον πρώτο εκ των δύο καταχωρητών που συμμετέχουν στην εντολή.
- $F = A + B/C - D * E$

LOAD	R1, A	ADD	R1, R2	SUB	R1, R2
LOAD	R2, B	LOAD	R2, D	STORE	F, R1
LOAD	R3, C	LOAD	R3, E		
DIV	R2, R3	MUL	R2, R3		

Μηχανή 3 εντέλων, εντολές καταχωρητή-μνήμης & μεταξύ καταχωρητών

- Οι εντολές καταχωρητή-μνήμης χρησιμοποιούνται μόνο για την προσκόμμιση δεδομένων στους καταχωρητές και την αποθήκευση αποτελεσμάτων στη μνήμη.
- Όλες οι αριθμητικές και λογικές εντολές είναι μεταξύ καταχωρητών.
- Η αποθήκευση γίνεται πάντα στον μοναδικό καταχωρητή ή στον πρώτο εκ των καταχωρητών που συμμετέχουν στην εντολή.

Μηχανή 3 εντέλων, εντολές καταχωρητή-μνήμης & μεταξύ καταχωρητών

- Οι εντολές καταχωρητή-μνήμης χρησιμοποιούνται μόνο για την προσκόμμιση δεδομένων στους καταχωρητές και την αποθήκευση αποτελεσμάτων στη μνήμη.
- Όλες οι αριθμητικές και λογικές εντολές είναι μεταξύ καταχωρητών.
- Η αποθήκευση γίνεται πάντα στον μοναδικό καταχωρητή ή στον πρώτο εκ των καταχωρητών που συμμετέχουν στην εντολή.
- $F = A + B/C + D * E$

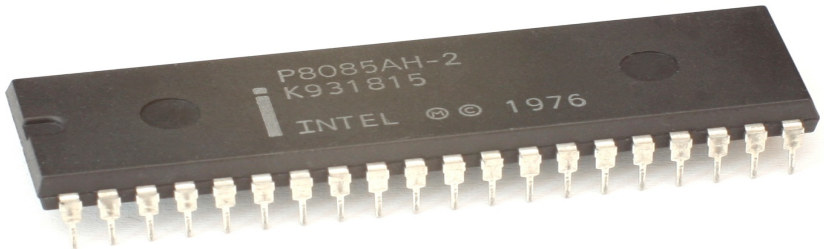
Μηχανή 3 εντέλων, εντολές καταχωρητή-μνήμης & μεταξύ καταχωρητών

- Οι εντολές καταχωρητή-μνήμης χρησιμοποιούνται μόνο για την προσκόμμιση δεδομένων στους καταχωρητές και την αποθήκευση αποτελεσμάτων στη μνήμη.
- Όλες οι αριθμητικές και λογικές εντολές είναι μεταξύ καταχωρητών.
- Η αποθήκευση γίνεται πάντα στον μοναδικό καταχωρητή ή στον πρώτο εκ των καταχωρητών που συμμετέχουν στην εντολή.
- $F = A + B/C + D * E$

LOAD	R1, A	LOAD	R3, D	ADD	R1, R1, R2
LOAD	R2, B	LOAD	R4, E	STORE	F, R1
LOAD	R3, C	MUL	R3, R3, R4		
DIV	R2, R2, R3	ADD	R2, R2, R3		

Γενικά για τον 8085

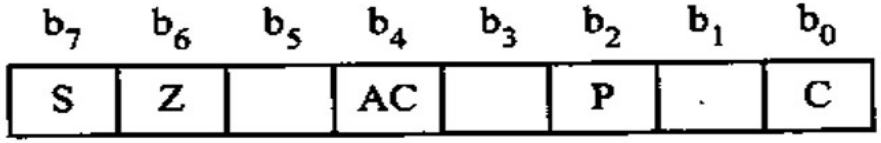
- Ο 8085 παρουσιάστηκε από την Intel το 1975, ως αντικαταστάτης του 4004 και του 8080 που είχαν προηγηθεί ιστορικά.
- Συνεχίζει να χρησιμοποιείται έως τις ημέρες μας.
- Ακολουθεί αρχιτεκτονική συσσωρευτή (Accumulator-based architecture).
- Έχει αρτηρία διευθύνσεων των 16 δυαδικών ψηφίων και αρτηρία δεδομένων των 8 δυαδικών ψηφίων.
- Η λέξη μηχανής είναι των 8 δυαδικών ψηφίων (σε 1 κύκλο εντολής εκτελεί πράξεις πάνω σε bytes).
- Είναι διαθέσιμος ως DIP ολοκληρωμένο 40 ακροδεκτών (pins)



Καταχωρητές του 8085

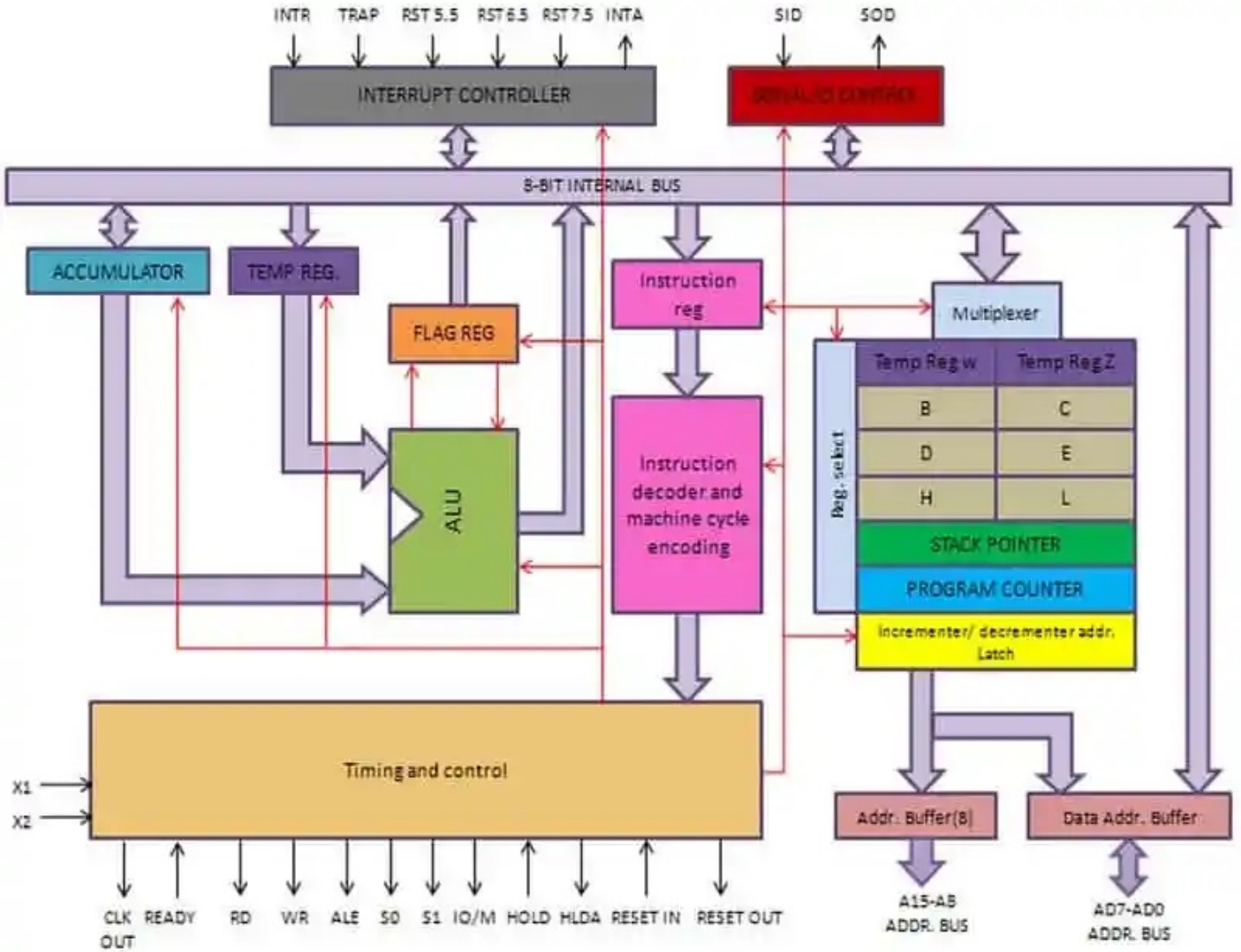
- Ειδικού Σκοπού:
  - Καταχωρητής εντολών 8 bit (instruction register).
  - Μετρητής προγράμαμτος 16 bit (program counter).
  - Δείκτης στοίβας 16 bit (stack pointer).
  - Καταχωρητής διευθύνσεων μνήμης 16 bit (memory address register).
  - Καταχωρητής δεδομένων μνήμης 8 bit (memory data register).
  - Καταχωρητής σημαιών κατάστασης 8 bit (PSW / flag register).
- Γενικού Σκοπού:
  - Συσσωρευτής 8 bit (Accumulator, A).
  - 6 ακόμη καταχωρητές 8 bit αναφερόμενοι ως B, C, D, E, H, L
  - Μπορούν να χρησιμοποιούνται αυτόνομα ή σε ζεύγη BC, DE, HL, των 16 ψηφίων. Το ζεύγος HL αναφέρεται ως M για register-indirect προσπελάσεις μνήμης.

Σημαίες Κατάστασης του PSW



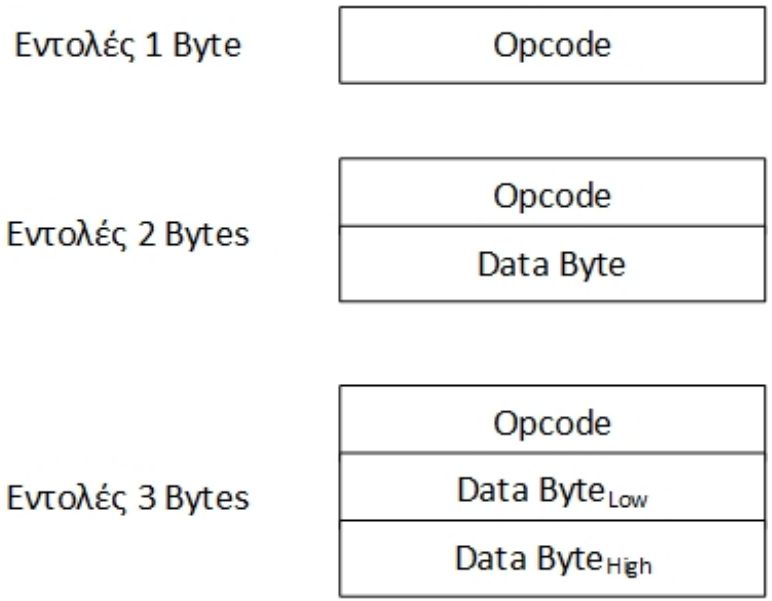
- Sign (Πρόσημο): Η σημαία παίρνει την τιμή 1 εάν το πιο σημαντικό δυαδικό ψηφίο του αποτελέσματος της εντολής είναι 1.
- Zero (Μηδενικό): Η σημαία παίρνει την τιμή 1 εάν το αποτέλεσμα της εντολής είναι 0.
- Auxiliary Carry (Βοηθητικό Κρατούμενο): Δηλώνει την ύπαρξη κρατουμένου μεταξύ του δυαδικού ψηφίου 3 και 4 του αποτελέσματος.
- Parity (Ισοτιμία): Η σημαία παίρνει την τιμή 1 εάν το αποτέλεσμα της εντολής έχει άρτια ισοτιμία.
- Carry (Κρατούμενο): Η σημαία παίρνει την τιμή 1 εάν προέκυψε κρατούμενο κατά την εκτέλεση της προηγούμενης εντολής.

Αρχιτεκτονικό Διάγραμμα του 8085



Μεγέθη εντολών

- Οι εντολές του 8085 μπορεί να έχουν μήκος 1, 2, ή 3 bytes.
- Οι εντολές με πολλά bytes αποθηκεύονται σε διαδοχικές θέσεις μνήμης.
- Το πρώτο byte κάθε εντολής περιέχει πάντα τον κωδικό λειτουργίας (opcode) της εντολής.
- Στις εντολές 3 bytes το δεδομένο των 16 bit αποθηκεύεται κατά Little Indian
- Όλα τα δεδομένα αναγράφονται στο 16δικό.

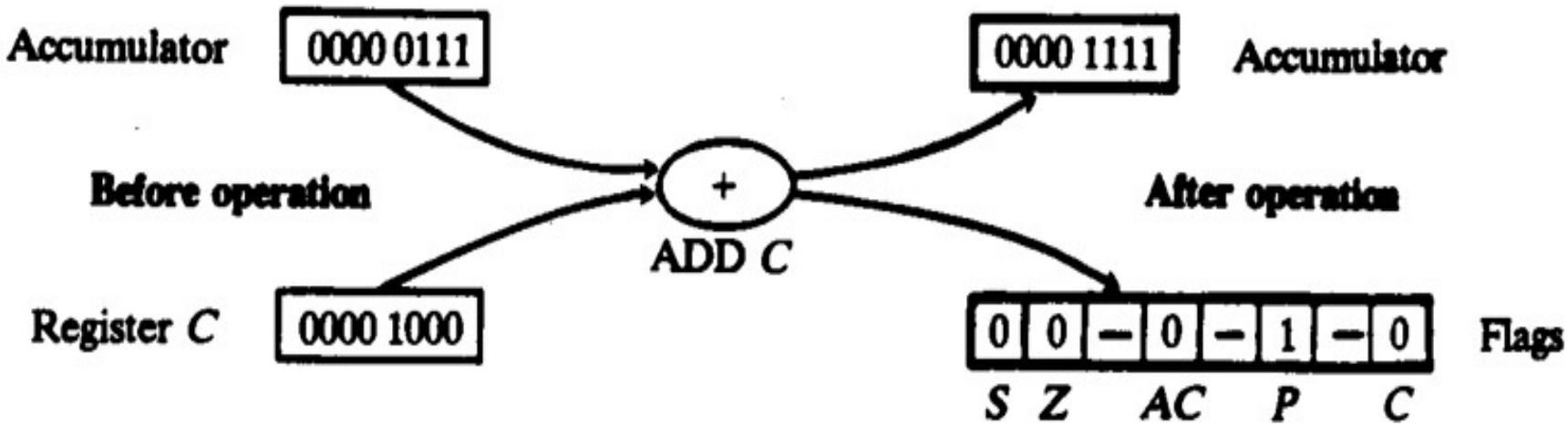


Τρόποι Διευθυνσιοδότησης

- Ο 8085 υποστηρίζει 4 τρόπους διευθυνσιοδότησης:
  - Κατ' ευθείαν με καταχωρητή (Register Direct)
    - Π.χ. INR A, ADD C, XRA B
    - Είναι εντολές 1 Byte (Opcode + Δνση Καταχωρητή)
  - Κατ' ευθείαν με μνήμη (Memory Direct)
    - Π.χ. LDA 2030
    - Είναι εντολές 3 Bytes (1^o Byte: Opcode, 2^o Byte: Low Byte of 16-bit Data, 3^o Byte: High Byte of 16-bit Data)
  - Άμεση (Immediate)
    - Π.χ. ADI F3, JNZ 43AA, LXI H 04FF
    - Είναι εντολές 2 ή 3 Bytes (1^o Byte: Opcode, 2^o Byte: 8-bit Data ή Low Byte of 16-bit Data, 3^o Byte (αν υπάρχει): High Byte of 16-bit Data)
  - Έμμεση μέσω καταχωρητή (Register indirect)
    - Π.χ. ADD M, LDAX B
    - Είναι εντολές 1 Byte (Opcode + Δνση Καταχωρητή)

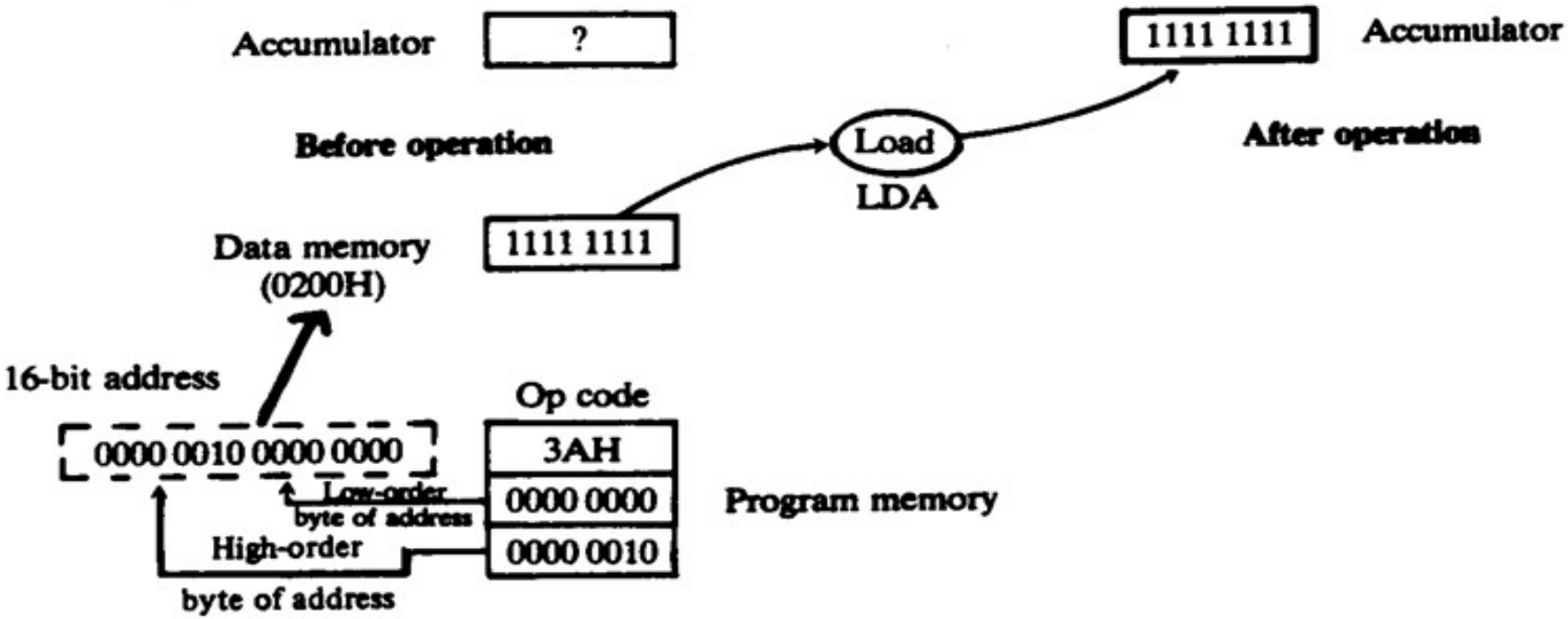
Κατ' ευθείαν με καταχωρητή (Register Direct)

- Η εντολή καθορίζει τον καταχωρητή ή το ζεύγος καταχωρητών όπου βρίσκονται αποθηκευμένα τα δεδομένα.
- ADD C



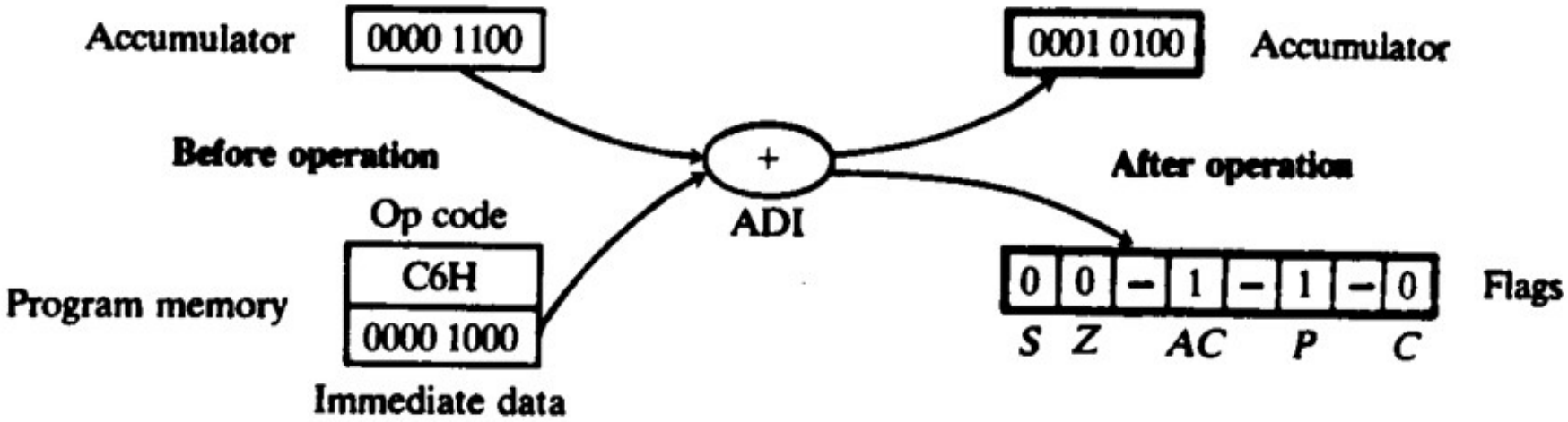
Κατ' ευθείαν με μνήμη (Memory Direct)

- Τα bytes 2 και 3 της εντολής περιέχουν τη διεύθυνση της μνήμης όπου βρίσκονται τα δεδομένα.
- Η διεύθυνση είναι αποθηκευμένη κατά Little Endian.
- LDA 0200



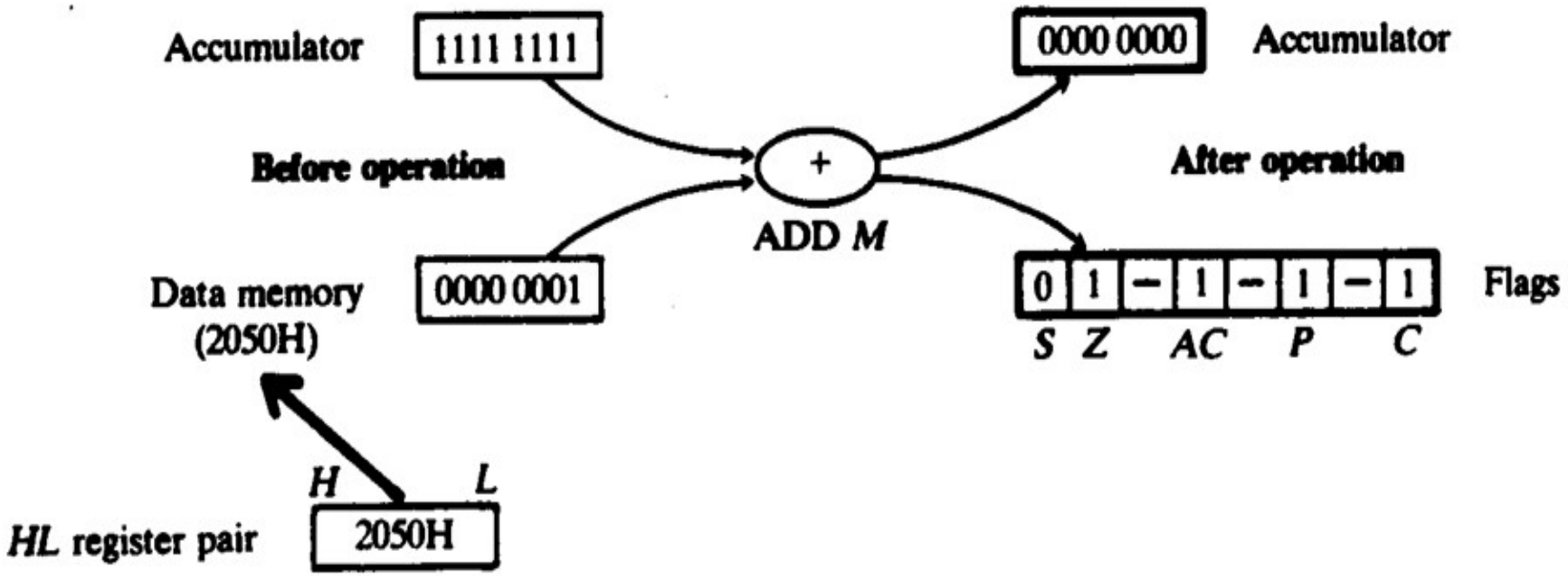
Άμεση (Immediate)

- Η εντολή περιέχει τα ίδια τα δεδομένα τα οποία μπορεί να είναι είτε 8 bit είτε 16 bit.
- Τα δεδομένα αναγράφονται στο 16δικό και εφόσον είναι των 16 bits είναι αποθηκευμένα κατά Little Endian.
- ADI 08



Έμμεση μέσω καταχωρητή (Register indirect)

- Η εντολή καθορίζει ένα ζεύγος καταχωρητών, τα περιεχόμενα των οποίων είναι η τελική (effective) διεύθυνση της μνήμης η οποία περιέχει τα δεδομένα.
- Πιθανά ζεύγη καταχωρητών BC, DE, HL (αναγράφεται ως M)
- ADD M



Σύνολο Εντολών (Instruction Set)

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NOP 1 4 - - - - - -	LXI B,d16 3 10 - - - - - -	STAX B 1 7 - - - - - -	INX B 1 6 - - - - - -	INR B 1 4 S Z A P -	DCR B 1 4 S Z A P -	MVI B,d8 2 7 - - - - - -	RLC 1 4 - - - - C		DAD B 1 10 - - - - C	LDAX B 1 7 - - - - - -	DCX B 1 6 - - - - - -	INR C 1 4 S Z A P -	DCR C 1 4 S Z A P -	MVI C,d8 2 7 - - - - - -	RRC 1 4 - - - - C
1x		LXI D,d16 3 10 - - - - - -	STAX D 1 7 - - - - - -	INX D 1 6 - - - - - -	INR D 1 4 S Z A P -	DCR D 1 4 S Z A P -	MVI D,d8 2 7 - - - - - -	RAL 1 4 - - - - C		DAD D 1 10 - - - - C	LDAX D 1 7 - - - - - -	DCX D 1 6 - - - - - -	INR E 1 4 S Z A P -	DCR E 1 4 S Z A P -	MVI E,d8 2 7 - - - - - -	RAR 1 4 - - - - C
2x	RIM 1 4 - - - - - -	LXI H,d16 3 10 - - - - - -	SHLD a16 3 16 - - - - - -	INX H 1 6 - - - - - -	INR H 1 4 S Z A P -	DCR H 1 4 S Z A P -	MVI H,d8 2 7 - - - - - -	DAA 1 4 S Z A P C		DAD H 1 10 - - - - C	LHLD a16 3 16 - - - - - -	DCX H 1 6 - - - - - -	INR L 1 4 S Z A P -	DCR L 1 4 S Z A P -	MVI L,d8 2 7 - - - - - -	CMA 1 4 - - - - - -
3x	SIM 1 4 - - - - - -	LXI SP,d16 3 10 - - - - - -	STA a16 3 13 - - - - - -	INX SP 1 6 - - - - - -	INR M 1 10 S Z A P -	DCR M 1 10 S Z A P -	MVI M,d8 2 10 - - - - - -	STC 1 4 - - - - C		DAD SP 1 10 - - - - C	LDA a16 3 13 - - - - - -	DCX SP 1 6 - - - - - -	INR A 1 4 S Z A P -	DCR A 1 4 S Z A P -	MVI A,d8 2 7 - - - - - -	CMC 1 4 - - - - C
4x	MOV B,B 1 4 - - - - - -	MOV B,C 1 4 - - - - - -	MOV B,D 1 4 - - - - - -	MOV B,E 1 4 - - - - - -	MOV B,H 1 4 - - - - - -	MOV B,L 1 4 - - - - - -	MOV B,M 1 7 - - - - - -	MOV B,A 1 4 - - - - - -	MOV C,B 1 4 - - - - - -	MOV C,C 1 4 - - - - - -	MOV C,D 1 4 - - - - - -	MOV C,E 1 4 - - - - - -	MOV C,H 1 4 - - - - - -	MOV C,L 1 4 - - - - - -	MOV C,M 1 7 - - - - - -	MOV C,A 1 4 - - - - - -
5x	MOV D,B 1 4 - - - - - -	MOV D,C 1 4 - - - - - -	MOV D,D 1 4 - - - - - -	MOV D,E 1 4 - - - - - -	MOV D,H 1 4 - - - - - -	MOV D,L 1 4 - - - - - -	MOV D,M 1 7 - - - - - -	MOV D,A 1 4 - - - - - -	MOV E,B 1 4 - - - - - -	MOV E,C 1 4 - - - - - -	MOV E,D 1 4 - - - - - -	MOV E,E 1 4 - - - - - -	MOV E,H 1 4 - - - - - -	MOV E,L 1 4 - - - - - -	MOV E,M 1 7 - - - - - -	MOV E,A 1 4 - - - - - -
6x	MOV H,B 1 4 - - - - - -	MOV H,C 1 4 - - - - - -	MOV H,D 1 4 - - - - - -	MOV H,E 1 4 - - - - - -	MOV H,H 1 4 - - - - - -	MOV H,L 1 4 - - - - - -	MOV H,M 1 7 - - - - - -	MOV H,A 1 4 - - - - - -	MOV L,B 1 4 - - - - - -	MOV L,C 1 4 - - - - - -	MOV L,D 1 4 - - - - - -	MOV L,E 1 4 - - - - - -	MOV L,H 1 4 - - - - - -	MOV L,L 1 4 - - - - - -	MOV L,M 1 7 - - - - - -	MOV L,A 1 4 - - - - - -
7x	MOV M,B 1 7 - - - - - -	MOV M,C 1 7 - - - - - -	MOV M,D 1 7 - - - - - -	MOV M,E 1 7 - - - - - -	MOV M,H 1 7 - - - - - -	MOV M,L 1 7 - - - - - -	HLT 1 5 - - - - - -	MOV M,A 1 7 - - - - - -	MOV A,B 1 4 - - - - - -	MOV A,C 1 4 - - - - - -	MOV A,D 1 4 - - - - - -	MOV A,E 1 4 - - - - - -	MOV A,H 1 4 - - - - - -	MOV A,L 1 4 - - - - - -	MOV A,M 1 7 - - - - - -	MOV A,A 1 4 - - - - - -
8x	ADD B 1 4 S Z A P C	ADD C 1 4 S Z A P C	ADD D 1 4 S Z A P C	ADD E 1 4 S Z A P C	ADD H 1 4 S Z A P C	ADD L 1 4 S Z A P C	ADD M 1 7 S Z A P C	ADD A 1 4 S Z A P C	ADC B 1 4 S Z A P C	ADC C 1 4 S Z A P C	ADC D 1 4 S Z A P C	ADC E 1 4 S Z A P C	ADC H 1 4 S Z A P C	ADC L 1 4 S Z A P C	ADC M 1 7 S Z A P C	ADC A 1 4 S Z A P C
9x	SUB B 1 4 S Z A P C	SUB C 1 4 S Z A P C	SUB D 1 4 S Z A P C	SUB E 1 4 S Z A P C	SUB H 1 4 S Z A P C	SUB L 1 4 S Z A P C	SUB M 1 7 S Z A P C	SUB A 1 4 S Z A P C	SBB B 1 4 S Z A P C	SBB C 1 4 S Z A P C	SBB D 1 4 S Z A P C	SBB E 1 4 S Z A P C	SBB H 1 4 S Z A P C	SBB L 1 4 S Z A P C	SBB M 1 7 S Z A P C	SBB A 1 4 S Z A P C
Ax	ANA B 1 4 S Z A P C	ANA C 1 4 S Z A P C	ANA D 1 4 S Z A P C	ANA E 1 4 S Z A P C	ANA H 1 4 S Z A P C	ANA L 1 4 S Z A P C	ANA M 1 7 S Z A P C	ANA A 1 4 S Z A P C	XRA B 1 4 S Z A P C	XRA C 1 4 S Z A P C	XRA D 1 4 S Z A P C	XRA E 1 4 S Z A P C	XRA H 1 4 S Z A P C	XRA L 1 4 S Z A P C	XRA M 1 7 S Z A P C	XRA A 1 4 S Z A P C
Bx	ORA B 1 4 S Z A P C	ORA C 1 4 S Z A P C	ORA D 1 4 S Z A P C	ORA E 1 4 S Z A P C	ORA H 1 4 S Z A P C	ORA L 1 4 S Z A P C	ORA M 1 7 S Z A P C	ORA A 1 4 S Z A P C	CMP B 1 4 S Z A P C	CMP C 1 4 S Z A P C	CMP D 1 4 S Z A P C	CMP E 1 4 S Z A P C	CMP H 1 4 S Z A P C	CMP L 1 4 S Z A P C	CMP M 1 7 S Z A P C	CMP A 1 4 S Z A P C
Cx	RNZ 1 12/6 - - - - - -	POP B 1 10 - - - - - -	JNZ a16 3 10/7 - - - - - -	JMP a16 3 10 - - - - - -	CNZ a16 3 18/9 - - - - - -	PUSH B 1 12 - - - - - -	ADI d8 2 7 S Z A P C	RST 0 1 12 - - - - - -	RZ 1 12/6 - - - - - -	RET 1 10 - - - - - -	JZ a16 3 10/7 - - - - - -		CZ a16 3 18/9 - - - - - -	CALL a16 3 18 - - - - - -	ACI d8 2 7 S Z A P C	RST 1 1 12 - - - - - -
Dx	RNC 1 12/6 - - - - - -	POP D 1 10 - - - - - -	JNC a16 3 10/7 - - - - - -	OUT d8 2 10 - - - - - -	CNC a16 3 18/9 - - - - - -	PUSH D 1 12 - - - - - -	SUI d8 2 7 S Z A P C	RST 2 1 12 - - - - - -	RC 1 12/6 - - - - - -		JC a16 3 10/7 - - - - - -	IN d8 2 10 - - - - - -	CC a16 3 18/9 - - - - - -		SBI d8 2 7 S Z A P C	RST 3 1 12 - - - - - -
Ex	RPO 1 12/6 - - - - - -	POP H 1 10 - - - - - -	JPO a16 3 10/7 - - - - - -	XTHL 1 16 - - - - - -	CPO a16 3 18/9 - - - - - -	PUSH H 1 12 - - - - - -	ANI d8 2 7 S Z A P C	RST 4 1 12 - - - - - -	RPE 1 12/6 - - - - - -	PCHL 1 6 - - - - - -	JPE a16 3 10/7 - - - - - -	XCHG 1 4 - - - - - -	CPE a16 3 18/9 - - - - - -		XRI d8 2 7 S Z A P C	RST 5 1 12 - - - - - -
Fx	RP 1 12/6 - - - - - -	POP PSW 1 10 S Z A P C	JP a16 3 10/7 - - - - - -	DI 1 4 - - - - - -	CP a16 3 18/9 - - - - - -	PUSH PSW 1 12 - - - - - -	ORI d8 2 7 S Z A P C	RST 6 1 12 - - - - - -	RM 1 12/6 - - - - - -	SPHL 1 6 - - - - - -	JM a16 3 10/7 - - - - - -	EI 1 4 - - - - - -	CM a16 3 18/9 - - - - - -		CPI d8 2 7 S Z A P C	RST 7 1 12 - - - - - -

Misc/control instructions
Jumps/calls
8bit load/store/move instructions
16bit load/store/move instructions
8bit arithmetic/logical instructions
16bit arithmetic/logical instructions

Length in bytes → INS reg (2 7 S Z A P C) ← Instruction mnemonic, ← Duration in cycles, ← Flags affected

Duration of conditional calls and returns is different when action is taken or not. This is indicated by two numbers separated by "/". The higher number (on the left side of "/") means duration of instruction when action is taken, the lower number (on the right side of "/") means duration of instruction when action is not taken.

Κατηγορίες Εντολών

- Εντολές μεταφοράς δεδομένων
- Αριθμητικές εντολές
- Λογικές εντολές
- Εντολές ολίσθησης
- Εντολές διακλάδωσης
- Εντολές εισόδου/εξόδου
- Εντολές κλήσης υπορουτίνας
- Εντολές στοίβας
- Εντολές ελέγχου

Εντολές μεταφοράς δεδομένων

- MOV Rx, Ry — $Rx \longleftarrow Ry$
- MOV Rx, M — $Rx \longleftarrow mem[HL]$
- MOV M, Rx — $mem[HL] \longleftarrow Rx$
- MVI Rx, byte — $Rx \longleftarrow byte$
- LXI Rp, bytes — $Rp \longleftarrow bytes$
- LDA addr — $A \longleftarrow mem[addr]$
- STA addr — $mem[addr] \longleftarrow A$

Αριθμητικές εντολές

- ADD Rx — $A \longleftarrow A + Rx$
- ADD M — $A \longleftarrow A + mem[HL]$
- ADI byte — $A \longleftarrow A + byte$
- ADC Rx — $A \longleftarrow A + Rx + Carry$
- ADC M — $A \longleftarrow A + mem[HL] + Carry$
- SUB Rx — $A \longleftarrow A - Rx$
- SUB M — $A \longleftarrow A - mem[HL]$
- SUI byte — $A \longleftarrow A - byte$
- SBB Rx — $A \longleftarrow A - Rx - Borrow$
- SBB M — $A \longleftarrow A - mem[HL] - Borrow$
- CMP Rx — $Compute\ the\ flags\ of\ A - Rx$
- CMP M — $Compute\ the\ flags\ of\ A - mem[HL]$
- INR Rx — $Rx \longleftarrow Rx + 1$
- DCR Rx — $Rx \longleftarrow Rx - 1$
- INX Rp — $Rp \longleftarrow Rp + 1$

Λογικές εντολές

- ANA Rx — $A \longleftarrow A \wedge Rx$
- ANA M — $A \longleftarrow A \wedge mem[HL]$
- ANI byte — $A \longleftarrow A \wedge byte$
- ORA Rx — $A \longleftarrow A \vee Rx$
- ORA M — $A \longleftarrow A \vee mem[HL]$
- ORI byte — $A \longleftarrow A \vee byte$
- XRA Rx — $A \longleftarrow A \oplus Rx$
- XRA M — $A \longleftarrow A \oplus mem[HL]$
- XRI byte — $A \longleftarrow A \oplus byte$
- CMA — $A \longleftarrow (-A)_{1s}$

Εντολές Ολίσθησης

- RLC $A \longleftarrow left\ \ rotation(A), Carry = leftmost\ \ bit$
- RRC $A \longleftarrow right\ \ rotation(A), Carry = rightmost\ \ bit$
- RAL $(Carry, A) \longleftarrow left\ \ rotation(Carry, A)$
- RAR $(A, Carry) \longleftarrow right\ \ rotation(A, Carry)$

Εντολές αλλαγής ροής

- JMP addr — $PC \longleftarrow addr$
- JZ addr — $If\ \ Z = 1, PC \longleftarrow addr$
- JNZ addr — $If\ \ Z = 0, PC \longleftarrow addr$
- JC addr — $If\ \ Carry = 1, PC \longleftarrow addr$
- JNC addr — $If\ \ Carry = 0, PC \longleftarrow addr$
- JP addr — $If\ \ S = 0, PC \longleftarrow addr$
- JM addr — $If\ \ S = 1, PC \longleftarrow addr$
- JPO addr — $If\ \ P = 0, PC \longleftarrow addr$
- JPE addr — $If\ \ P = 1, PC \longleftarrow addr$

Εντολές στοίβας & Κλήσης υπορουτινών

- PUSH Rp $Stack[SP] \longleftarrow RP,\ \ SP \longleftarrow SP - 2$
- POP Rp $RP \longleftarrow Stack[SP],\ \ SP \longleftarrow SP + 2$

Εντολές στοίβας & Κλήσης υπορουτινών

- PUSH Rp $Stack[SP] \longleftarrow RP,\ \ SP \longleftarrow SP - 2$
- POP Rp $RP \longleftarrow Stack[SP],\ \ SP \longleftarrow SP + 2$

- CALL addr $Stack[SP] \longleftarrow PC,\ \ SP \longleftarrow SP - 2,\ \ PC \longleftarrow addr$
- RET $PC \longleftarrow Stack[SP],\ \ SP \longleftarrow SP + 2$

Παράδειγμα 1

- Να αναπτυχθεί πρόγραμμα που να υπολογίζει τον αντίθετο του byte που βρίσκεται αποθηκευμένο στη θέση μνήμης 20F3 και να τον αποθηκεύει στη θέση μνήμης 20F4.

Παράδειγμα 1

- Να αναπτυχθεί πρόγραμμα που να υπολογίζει τον αντίθετο του byte που βρίσκεται αποθηκευμένο στη θέση μνήμης 20F3 και να τον αποθηκεύει στη θέση μνήμης 20F4.

```
LDA 20F3
CMA
INR A
STA 20F4
HLT
```

Παράδειγμα 2

- Να αναπτυχθεί πρόγραμμα που να μηδενίζει τα 4 πιο σημαντικά ψηφία του αθροίσματος του 100_{10} με την τιμή της θέσης μνήμης 2000 και να αποθηκεύει το αποτέλεσμα στη 2001.

Παράδειγμα 2

- Να αναπτυχθεί πρόγραμμα που να μηδενίζει τα 4 πιο σημαντικά ψηφία του αθροίσματος του 100_{10} με την τιμή της θέσης μνήμης 2000 και να αποθηκεύει το αποτέλεσμα στη 2001.

```
LXI H,2000
MOV A, M
ADI 64
ANI 0F
INX H
MOV M, A
HLT
```

Παράδειγμα 3

- Να δώσετε τη γενική μορφή ενός προγράμματος που να εκτελεί ένα βρόχο (loop) 20 φορές.

Παράδειγμα 3

- Να δώσετε τη γενική μορφή ενός προγράμματος που να εκτελεί ένα βρόχο (loop) 20 φορές.

```
        MVI B,14
LOOP:

        Εντολές

        DCR B
        JNZ LOOP
        HLT
```

Παράδειγμα 4

- Να αναπτύξετε ένα πρόγραμμα που να υπολογίζει το άθροισμα Ν αριθμών. Το Ν βρίσκεται στη θέση μνήμης 04FF και η λίστα των προσθετέων ξεκινά από τη 0500. Το άθροισμα θα πρέπει να αποθηκεύεται στην αμέσως επόμενη θέση μνήμης από το τέλος των προσθετέων.

Παράδειγμα 4

- Να αναπτύξετε ένα πρόγραμμα που να υπολογίζει το άθροισμα Ν αριθμών. Το Ν βρίσκεται στη θέση μνήμης 04FF και η λίστα των προσθετέων ξεκινά από τη 0500. Το άθροισμα θα πρέπει να αποθηκεύεται στην αμέσως επόμενη θέση μνήμης από το τέλος των προσθετέων.

```
        LXI H,04FF
        MOV B, M
        INX H
        SUB A
LOOP:   ADD M
        INX H
        DCR B
        JNZ LOOP
        MOV M,A
        HLT
```

Παράδειγμα 5

- Να βρείτε και να αποθηκεύσετε στη θέση μνήμης 3000 τον ελάχιστο μη προσημασμένο αριθμό μιας λίστας. Ο αριθμός των στοιχείων της λίστας υπάρχει στη θέση μνήμης 1FFF και οι αριθμοί της λίστας ξεκινούν από τη διεύθυνση 2000.

Παράδειγμα 5

- Να βρείτε και να αποθηκεύσετε στη θέση μνήμης 3000 τον ελάχιστο μη προσημασμένο αριθμό μιας λίστας. Ο αριθμός των στοιχείων της λίστας υπάρχει στη θέση μνήμης 1FFF και οι αριθμοί της λίστας ξεκινούν από τη διεύθυνση 2000.

```
        LXI H,1FFF
        MOV B, M
        INX H
        MOV A, M
        DCR B
        JZ END
AGAIN:  INX H
        CMP M
        JC NEXT
        MOV A, M
NEXT:   DCR B
        JNZ AGAIN
END:    STA 3000
        HLT
```

Παράδειγμα 6

- Να αναπτύξετε πρόγραμμα που να υπολογίζει το ακέραιο μέρος της ρίζας του byte που βρίσκεται στη θέση μνήμης 2030 και να το αποθηκεύει στη θέσημνήμης 2031.

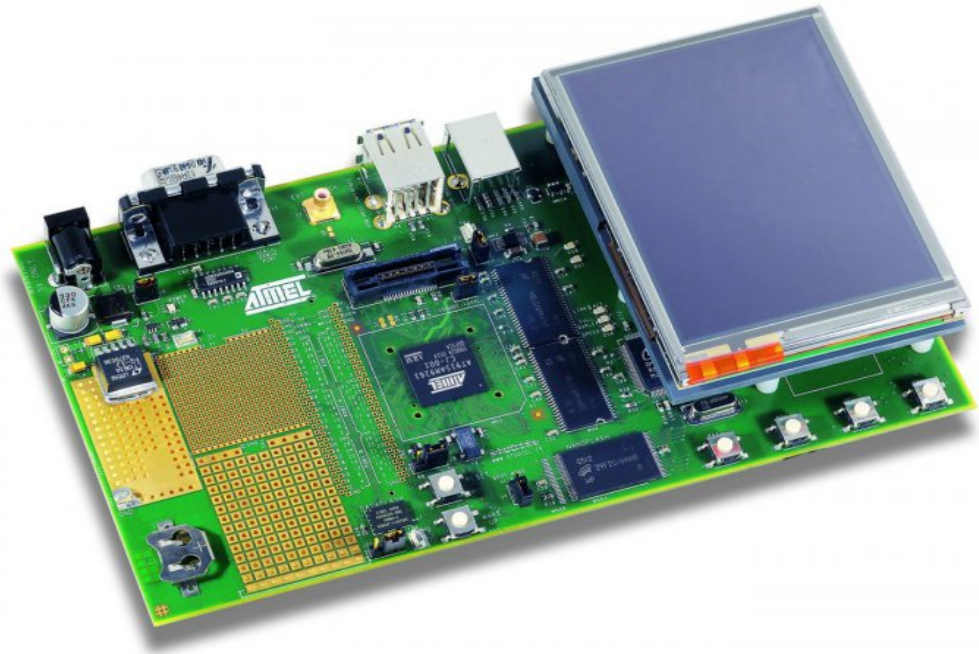
Παράδειγμα 6

- Να αναπτύξετε πρόγραμμα που να υπολογίζει το ακέραιο μέρος της ρίζας του byte που βρίσκεται στη θέση μνήμης 2030 και να το αποθηκεύει στη θέσημνήμης 2031.

```
        MVI B, 01
        MVI C, 00
        LXI H, 2030
        MOV A, M
J1:     SUB B
        JC J2
        INR B
        INR B
        INR C
        JMP J1
J2:     INX H
        MOV M, C
        HLT
```

Γενικά για το AT91

- Το AT91 είναι ένα σύστημα αναπτυγμένο πάνω στην πλακέτα AT91SAM9261EK.
- Η πλακέτα είναι χτισμένη γύρω από την υλοποίηση από την ATMEL της RISC αρχιτεκτονικής 926EJ-S της ARM.
- Είναι μια αρχιτεκτονική με καταχωρητές γενικού σκοπού (load - store architecture) με αριθμητικές και λογικές εντολές 3 εντέλων.
- Χρησιμοποιεί έναν τροποποιημένο πυρήνα Linux με όλα τα εργαλεία του GNU.



Αρχιτεκτονικά στοιχεία & ιδιαιτερότητες

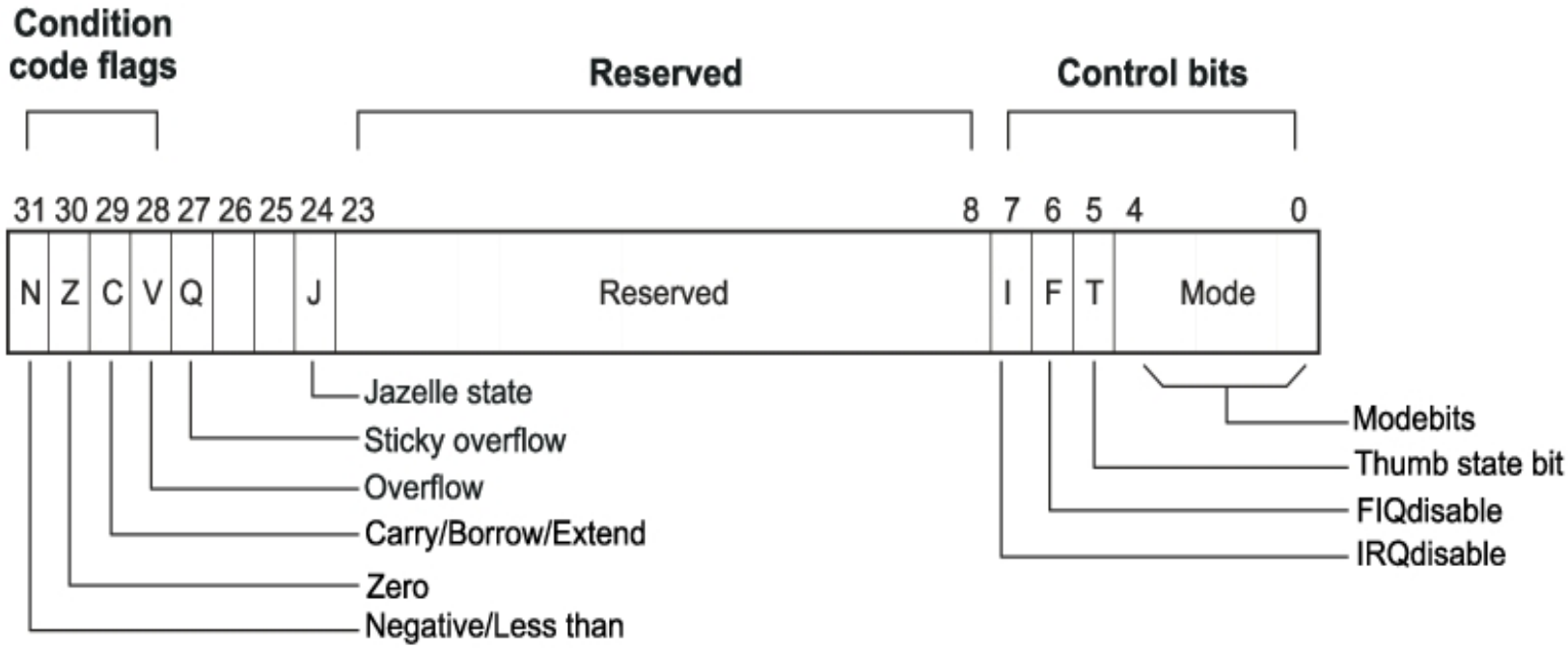
- Το AT91 μπορεί να λειτουργήσει είτε με εντολές των 32 δυαδικών ψηφίων (ARM mode) είτε των 16 δυαδικών ψηφίων (Thumb mode).
- Η αρτηρία διευθύνσεων του AT91 είναι των 32 δυαδικών ψηφίων.
- Η λέξη της μνήμης στο AT91 είναι των 8 δυαδικών ψηφίων.
- Σε έναν κύκλο εντολής μπορούμε να κάνουμε πράξεις σε τελούμενα ενός byte, 2 bytes (halfword) ή 4 bytes (word - λέξη μηχανής).
- Παρότι η ARM αρχιτεκτονική υποστηρίζει τόσο Little όσο και Big Endian αποθήκευση δεδομένων Halfword και Word, η επιλογή του Linux καθόρισε το AT91 να είναι Little Endian και ταυτόχρονα να επιβάλλεται στοίχιση δεδομένων (alignment).
- Το AT91 χρησιμοποιεί τη τεχνική της συνεχούς διοχέτευσης (pipelining) για αύξηση της απόδοσης.
- Υπάρχει ξεχωριστή μονάδα πολλαπλασιασμού υλοποιημένη στο υλικό.

Καταχωρητές

- Το AT91 μπορεί να βρίσκεται σε ένα από τα System & User, FIQ, Supervisor, Abort, IRQ, Undefined modes λειτουργίας. Σε καθένα απ' αυτά έχει ένα συγκεκριμένο set καταχωρητών από τους συνολικά 37 το οποίο μπορεί να χρησιμοποιήσει.
- Στο System & User mode, μπορούμε να προσπελάσουμε 16 καταχωρητές των 32 ψηφίων, εκ των οποίων:
  - Ο r15 είναι ο μετρητής προγράμματος (PC).
  - Ο r14 είναι ο καταχωρητής διασύνδεσης κώδικα (Branch & Link).
  - Ο r13 είναι ο δείκτης στοίβας (Stack Pointer)
- Σε μια εντολή μπορούμε να χρησιμοποιήσουμε τα 8 λιγότερα σημαντικά ψηφία ή τα 16 λιγότερο σημαντικά ψηφία ή τέλος και τα 32 ψηφία κάποιου καταχωρητή.
- Φυσικά υπάρχουν και οι γνωστοί IR, MAR, MDR.

Καταχωρητής Κατάστασης

- CPSR (Current Processor Status Register)



- Μπορούμε να επιλέξουμε αν θα ανανεωθούν οι σημαίες κατάστασης με S στο τέλος της εντολής. Πχ. ADD vs ADDS
- Εκτέλεση υπό συνθήκη, π.χ. ADDCS (add if Carry Set)

Εξελιγμένες δυνατότητες ολίσθησης

- Ένα μοναδικό στοιχείο των ARM αρχιτεκτονικών είναι ότι μπορούν στον ίδιο κύκλο μηχανής να κάνουν ολίσθηση και κάποια αριθμητική / λογική πράξη.
- Δηλαδή στη μία είσοδο (port) της ALU οδηγείται κάποιος καταχωρητής και στην άλλη μπορεί να οδηγηθεί καταχωρητής, δεδομένο που παρέχεται άμεσα στην εντολή ή τα προηγούμενα ολισθημένα κατά 1 έως 31 θέσεις (έξοδος του ολισθητή).

Τρόποι Διευθυνσιοδότησης

- Το AT91 υποστηρίζει τους εξής τρόπους διευθυνσιοδότησης:

Κατ' ευθείαν με καταχωρητή	MOV r0, r1
Κατ' ευθείαν με Μνήμη	LDR r0, 0xFF3ABAB8
Άμεσος	ADD r1, r2, #12
Έμμεσος με καταχωρητή	LDR r0, [r1]
Έμμεσος δεικτοδοτημένος	LDR r0, [r1, #4]
Έμμεσος με ενημέρωση	LDR r0, [r1], #4
Έμμεσος δεικτοδοτημένος με ενημέρωση	LDR r0, [r1, #4]!
Έμμεσος δεικτοδοτημένος με 2 καταχωρητές	LDR r0, [r1, r2]
Έμμεσος δεικτοδοτημένος με 2 καταχωρητές & κλιμάκωση	LDR r0, [r1, r2, LSL #2]

Μορφή Προγράμματος

```
.arm
.text
.global main

main:
STMDB R13!, {R0-R12}

Εντολές του προγράμματός μας

LDMIA R13!, {R0-R12}
MOV PC, R14

.data
Τοποθετούμε τα δεδομένα του προγράμματός μας.
```

Παράδειγμα 1

- Μεταφορά δεδομένου 32 δυαδικών ψηφίων

Παράδειγμα 1

- Μεταφορά δεδομένου 32 δυαδικών ψηφίων

```
.arm
.text
.global main

main:
STMDB R13!, {R0-R12}
LDR R0,=Value
LDR R1, [R0]
LDR R0,=Result
STR R1, [R0]
LDMIA R13!, {R0-R12}
MOV PC, R14

.data
Value: .word 0x55AAFEFE
Result: .word 0
```

Παράδειγμα 2

- Πρόσθεση δεδομένων 32 δυαδικών ψηφίων

Παράδειγμα 2

- Πρόσθεση δεδομένων 32 δυαδικών ψηφίων

```
.arm
.text
.global main

main:
STMDB R13!, {R0-R12}
LDR R0,=Values
LDR R1, [R0]
LDR R2, [R0, #4]
ADD R2, R2, R1
LDR R0,=Result
STR R2, [R0]
LDMIA R13!, {R0-R12}
MOV PC, R14

.data
Values:
.word 0x98726534
.word e5e4e3e2
Result: .word 0xffffffff
```

- Γιατί δεν ανανεώθηκε ο CPSR;

Παράδειγμα 3

- Πρόσθεση δεδομένων 32 δυαδικών ψηφίων με CPSR update

Παράδειγμα 3

- Πρόσθεση δεδομένων 32 δυαδικών ψηφίων με CPSR update

```
.arm                          | LDR R0,=Result
.text                         | STR R2, [R0]
.global main                  | LDMIA R13!, {R0-R12}
                              | MOV PC, R14
main:                         |
STMDB R13!, {R0-R12}          | .data
LDR R0,=Values                | Values:
LDR R1, [R0]                  | .word 0x98726534
LDR R2, [R0, #4]              | .word e5e4e3e2
ADDS R2, R2, R1               | Result:  .word 0xffffffff
```

- Στο AT91 ο προγραμματιστής ελέγχει την ανανέωση των σημαιών!

Παράδειγμα 4

- Εύρεση του αθροίσματος ενός πίνακα από αριθμούς

Παράδειγμα 4

- Εύρεση του αθροίσματος ενός πίνακα από αριθμούς

```
...
LDR R1, =Length
LDR R1, [R1]
LDR R2, =Numbers
MOV R0, #0

Loop:
LDR R3, [R2], #4
ADD R0, R0, R3
SUBS R1, R1, #1
BGT Loop

LDR R4, =Result
STR R0, [R4]
...
```

Παράδειγμα 5

- Εύρεση του μεγαλύτερου από 5 αριθμούς 1 Byte

Παράδειγμα 5

- Εύρεση του μεγαλύτερου από 5 αριθμούς 1 Byte

```
...
LDR R0, =Values
LDRB R1, [R0, #0]
ADD R3, R0, #4

Loop:
LDRB R2, [R0, #1]!
CMP R2, R1
MOVHI R1, R2
CMP R0, R3
BLO Loop

LDR R0, =Result
STRB R1, [R0]
...
```

Γενικά για τις αρτηρίες

- Οι αρτηρίες επιτρέπουν τη διασύνδεση των μονάδων ενός υπολογιστικού συστήματος, με στόχο την ανταλλαγή πληροφοριών μεταξύ τους και την εκτέλεση των προγραμμάτων.

Γενικά για τις αρτηρίες

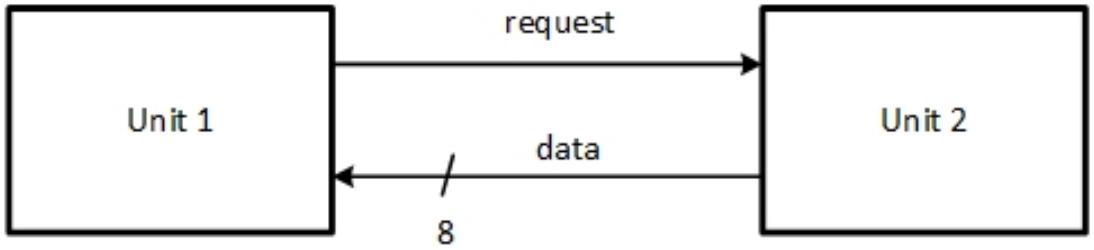
- Οι αρτηρίες επιτρέπουν τη διασύνδεση των μονάδων ενός υπολογιστικού συστήματος, με στόχο την ανταλλαγή πληροφοριών μεταξύ τους και την εκτέλεση των προγραμμάτων.
- Αρτηρίες θα βρούμε:
  - Εσωτερικά σε κάποια ολοκληρωμένα κυκλώματα
  - Πάνω στις πλακέτες και θα διασυνδέουν διάφορα ολοκληρωμένα, και
  - Ως θύρες επέκτασης ενός υπολογιστικού συστήματος.

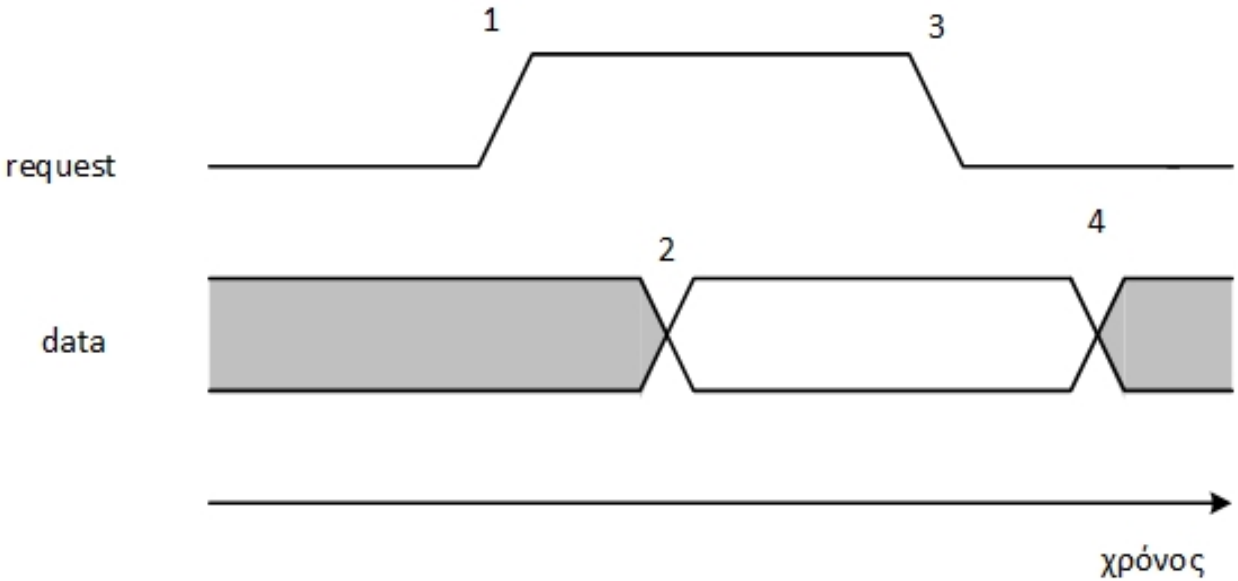
Γενικά για τις αρτηρίες

- Οι αρτηρίες επιτρέπουν τη διασύνδεση των μονάδων ενός υπολογιστικού συστήματος, με στόχο την ανταλλαγή πληροφοριών μεταξύ τους και την εκτέλεση των προγραμμάτων.
- Αρτηρίες θα βρούμε:
  - Εσωτερικά σε κάποια ολοκληρωμένα κυκλώματα
  - Πάνω στις πλακέτες και θα διασυνδέουν διάφορα ολοκληρωμένα, και
  - Ως θύρες επέκτασης ενός υπολογιστικού συστήματος.
- Με τον όρο αρτηρία (bus) εννοούμε:
  - Αφενός τις γραμμές διασύνδεσης (κάθε μία μεταφέρει ένα δυαδικό σήμα), και
  - Αφετέρου, τους κανόνες επικοινωνίας, που καλούνται πρωτόκολλο (Bus protocol)

Χρονικό διάγραμμα

- Ο πλέον διαδεδομένος τρόπος περιγραφής του πρωτοκόλλου μιας αρτηρίας είναι μέσω ενός χρονικού διαγράμματος.





Κύριοι & Σκλάβοι

- Μια μονάδα που μπορεί να ξεκινήσει και να καθοδηγήσει μια συναλλαγή πάνω από μια αρτηρία ονομάζεται μονάδα αφέντης ή μονάδα κύριος της αρτηρίας (bus master).
- Οι μονάδες που μόνο αποκρίνονται στις απαιτήσεις των κυρίων μιας αρτηρίας ονομάζονται μονάδες υπηρέτες ή μονάδες δούλοι (slaves)

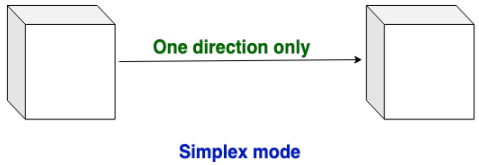
Ταξινόμηση των αρτηριών

- Οι διάφορες αρτηρίες που μπορούμε να συναντήσουμε μπορούν να ταξινομηθούν ανάλογα με:
  - Την κατεύθυνση της μετάδοσης δεδομένων,
  - Την ποσότητα της πληροφορίας που μπορεί να διακινηθεί ανά συναλλαγή (ισοδύναμα, με τον διατιθέμενο αριθμό γραμμών δεδομένων),
  - Το πόσες μονάδες διαμοιράζονται την αρτηρία,
  - Την τεχνική συγχρονισμού που υιοθετείται,
  - Την τεχνική σηματοδότησης που ακολουθείται,
  - Τον τρόπο μεταφοράς δεδομένων, και
  - Την διαιτησία που ακολουθείται.

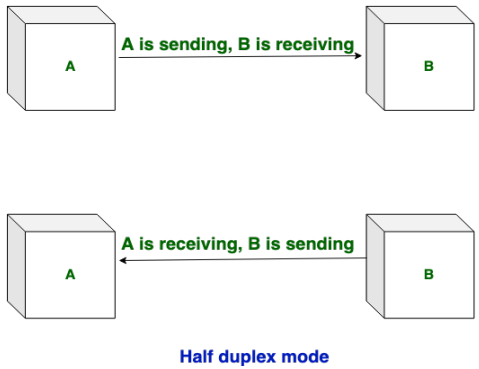
Κατεύθυνση της μεταφοράς

Ανάλογα με την κατεύθυνση της μεταφοράς, χωρίζονται σε:

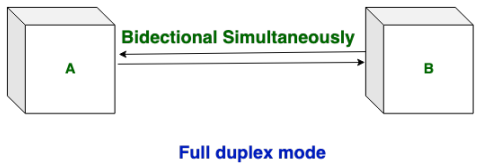
- Simplex (Μονής Κατεύθυνσης)



- Half - Duplex (δύο εναλλακτικών κατευθύνσεων), και



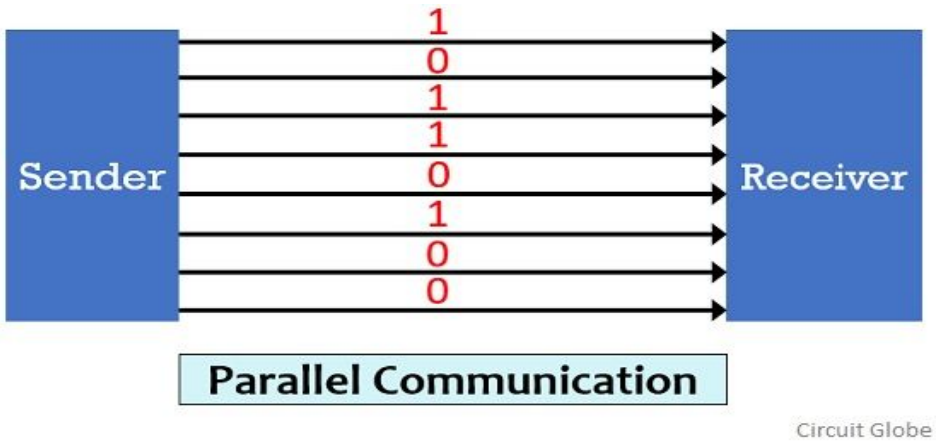
- Full - Duplex (Δύο κατευθύνσεων).



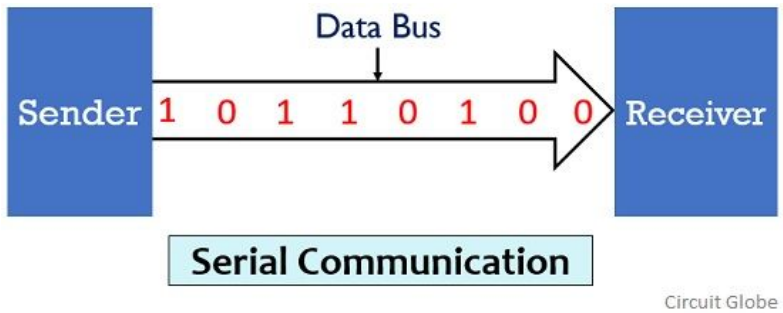
Αριθμός γραμμών δεδομένων

Ανάλογα με τον αριθμό των διατιθέμενων γραμμών δεδομένων, χωρίζονται σε:

- Parallel Buses (Παράλληλες), και



- Serial Buses (Σειραϊκές).



Παράλληλες αρτηρίες

- Συνήθως το πρωτόκολλό τους περιλαμβάνει:
  - Γραμμές ελέγχου,
  - Γραμμές διευθύνσεων, και
  - Γραμμές δεδομένων.
- Υπάρχει περίπτωση για πολυπλεξία στο χρόνο (time-multiplexing) των γραμμών διευθύνσεων και δεδομένων, με αποτέλεσμα:
  - Οικονομία στις απαιτούμενες γραμμές, αλλά και
  - Μικρότερο ρυθμό μεταφοράς (bandwidth).
- Υπάρχει περίπτωση για ομαδική μεταφορά (burst) με την αποστολή μίας μόνο διεύθυνσης.
- Οι παράλληλες αρτηρίες πάσχουν από το πρόβλημα του συγχρονισμού των δεδομένων τους και από χαμηλές ταχύτητες όταν το μήκος τους αυξάνεται.

Σειραϊκές αρτηρίες

- Συνήθως το πρωτόκολλό τους περιλαμβάνει:
  - Γραμμές ελέγχου,
  - Μία γραμμή δεδομένων, και
  - Γραμμές ισχύος.
- Έχουν μικρότερο κόστος υλοποίησης και επιτυγχάνουν υψηλότερες ταχύτητες έναντι των παράλληλων για αυξημένο μήκος διασύνδεσης.

Παράλληλες vs Σειραϊκές

Μέτρο Σύγκρισης	**Παράλληλες**	**Σειραϊκές**
Ταχύτητα (μικρό μήκος)	Μέτρια	Χαμηλή
Γραμμές Δεδομένων	Πολλαπλές	Μία
Κόστος	Υψηλό	Χαμηλό
Η/Μ Παρεμβολή	Πιθανή	Απίθανη
Αναβάθμιση	Αρκετά δύσκολη	Εύκολη
Κατεύθυνση	Half-Duplex	Full-Duplex
Κατάλληλη για	Μικρές αποστάσεις	Μεγάλες αποστάσεις
Συχνότητα ρολογιού	Μέτρια	Πολύ υψηλή

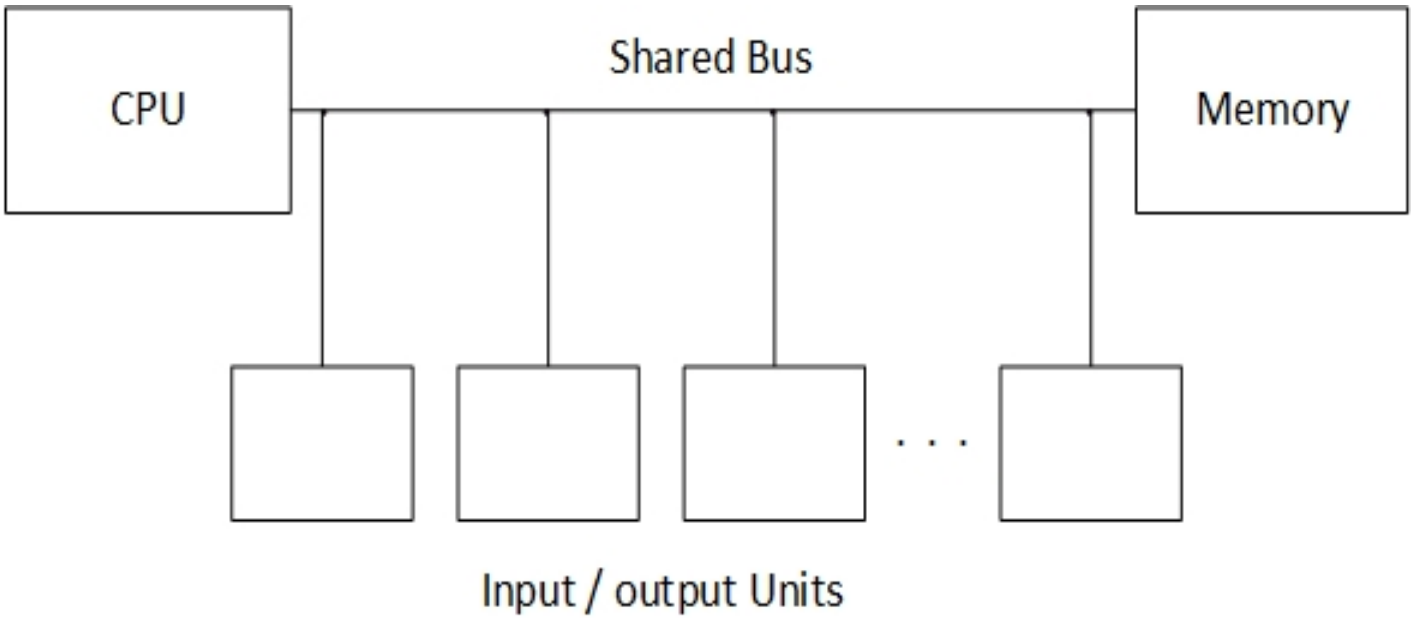
Μερικές Γνωστές Αρτηρίες

	Όνομα αρτηρίας			
Χαρακτηριστικά	**PCI**	**SCSI**	**Firewire**	**USB**
είδος	**παράλληλη**	**παράλληλη**	**σειριακή**	**σειριακή**
χρήση	**συστήματος**	**εισ/εξ**	**εισ/εξ**	**εισ/εξ**
Εύρος σε δυαδικά ψηφία δεδομένων	**32-64**	**8-32**	**1**	**1**
Μέγιστος ρυθμός μεταφοράς δεδομένων (MB/sec)	**133-512**	**5-40**	**50 (Firewire 400) 100(Firewire 800)**	**V1: 0,2 V1.1: 1,5 V2.0: 60 V3.0:625**
Μέγιστος αριθμός συνδεδεμένων μονάδων	**32**	**7-31**	**63**	**127**
Μέγιστο μήκος (μέτρα)	**1**	**3-25**	**4,5**	**5**

Αριθμός Μονάδων Χρηστών της Αρτηρίας

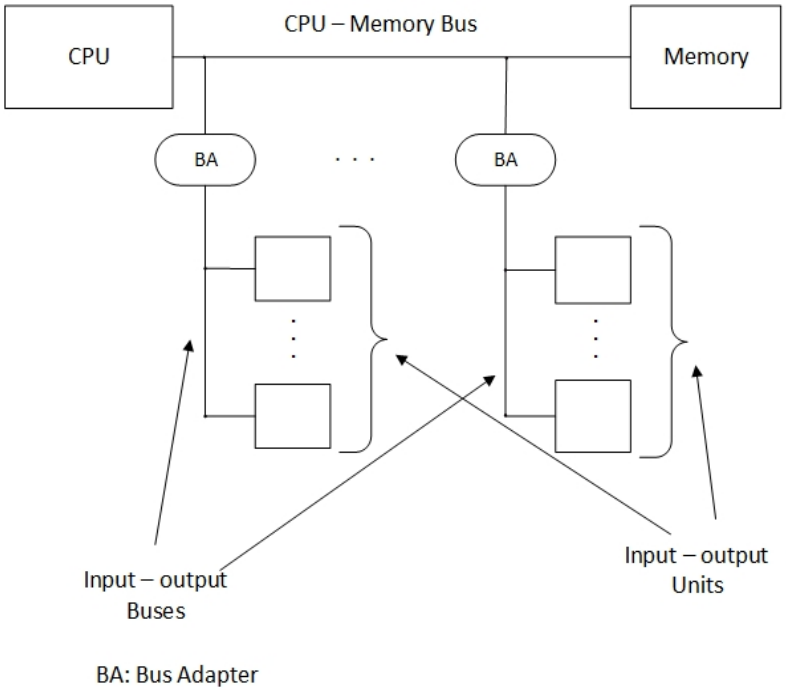
- Κάθε σήμα αρτηρίας μπορεί να οδηγείται μόνο από 1 μονάδα ανά χρονική στιγμή.
- Οι υπόλοιπες θα πρέπει να αναμένουν να γίνει διαθέσιμη η αρτηρία.
- Αρτηρίες αποκλειστικής χρήσης (dedicated buses ή point to point):
  - Χρησιμοποιούνται για την αποκλειστική χρήση από δύο μόνο μονάδες.
  - Για τη διασύνδεση n μονάδων θα απαιτηθούν $\binom{n}{2}$ αρτηρίες.
  - - : Πολύ υψηλό κόστος.
  - + : Ανοχή σφαλμάτων.
- Αρτηρίες κοινής χρήσης (shared bus):
  - Χρησιμοποιείται για τη σύνδεση όλων των μονάδων του συστήματος.
  - Απαιτεί την ύπαρξη μηχανισμού διαιτησίας.
  - + : Χαμηλό κόστος, ευκολία στην προσθήκη μονάδων.
  - - : Αργή επικοινωνία, ευαισθησία σε βλάβες

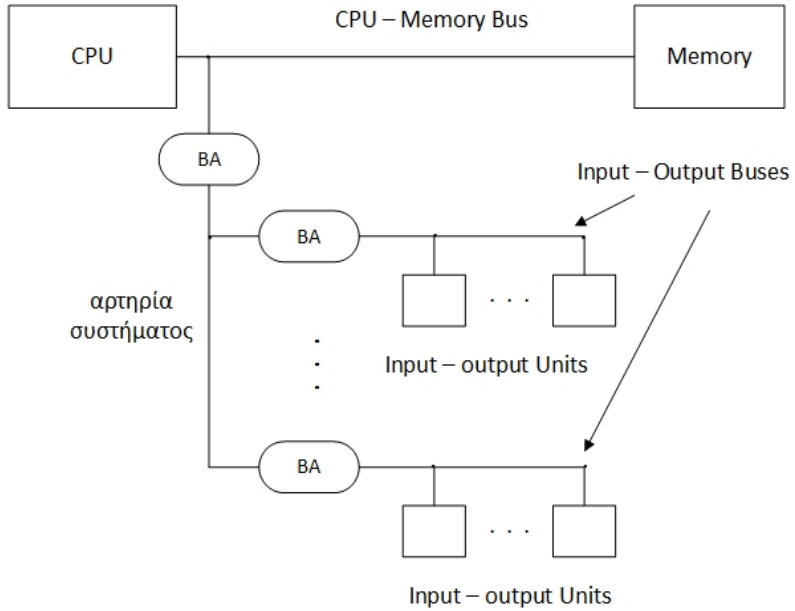
Παράδειγμα αρτηρίας κοινής χρήσης



Πολλαπλές αρτηρίες κοινής χρήσης

- Μπορούμε να θεωρήσουμε τα συστήματα στα οποία χρησιμοποιούνται μόνο αρτηρίες αποκλειστικής χρήσης ή μόνο μία διαμοιραζόμενη κοινή αρτηρία σαν τις δύο ακραίες πιθανές περιπτώσεις.
- Στην πράξη χρησιμοποιούνται μείγματα των δύο αυτών.





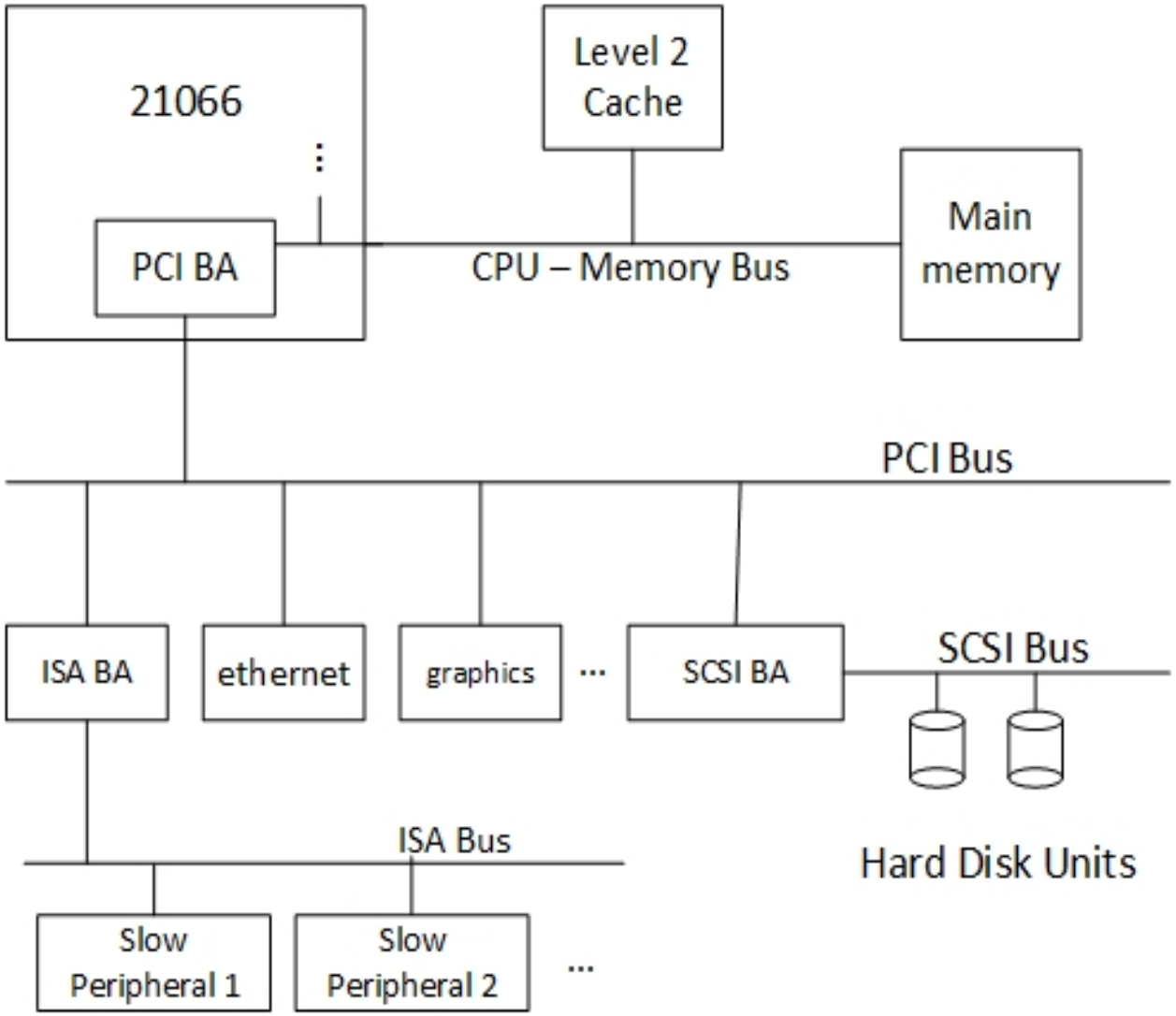
Είδη αρτηριών κοινής χρήσης

- Αρτηρίες επεξεργαστή - μνήμης
  - Μικρού μήκους
  - Υψηλής ταχύτητας
  - Ταιριάζουν με τα χαρακτηριστικά του επεξεργαστή και της μνήμης. Σκοπός η μεγιστοποίηση του ρυθμού μεταφοράς πληροφορίας μεταξύ τους.
  - Intellectual Property (π.χ. AHB της ARM)
- Αρτηρίες εισόδου-εξόδου
  - Μεγαλύτερου μήκους
  - Μπορούν να συνδεθούν πολλά είδη μονάδων εισόδου-εξόδου με διαφορετικούς ρυθμούς μεταφοράς δεδομένων
  - π.χ. ISA, EISA, EIDE, Micro-Channel, SCSI, SATA
- Αρτηρίες συστήματος (local bus), π.χ. PCI

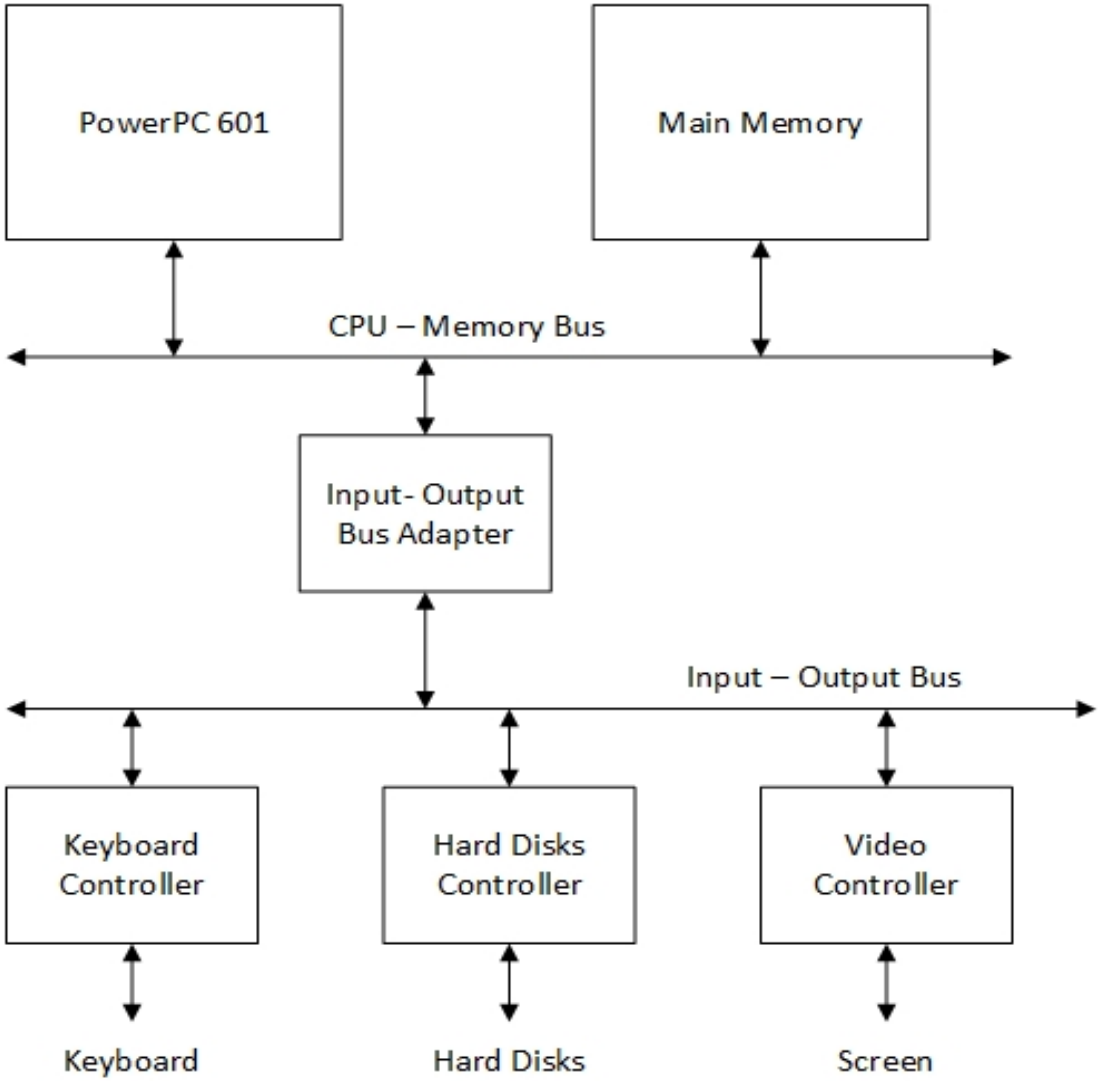
Πλεονεκτήματα χρήσης πολλαπλών αρτηριών

- Στην ουσία έχουμε κι εδώ ένα ιεραρχικό σύστημα.
- Η αρτηρία επεξεργαστή-μνήμης μπορεί να σχεδιαστεί να είναι πολύ πιο γρήγορη από ότι η αρτηρία συστήματος ή η αρτηρία εισόδου/εξόδου.
- Νέες μονάδες ή αρτηρίες εισόδου/εξόδου μπορούν να συνδεθούν στην αρτηρία συστήματος, χωρίς να επηρεαστεί η ταχύτητα της αρτηρίας επεξεργαστή-μνήμης.

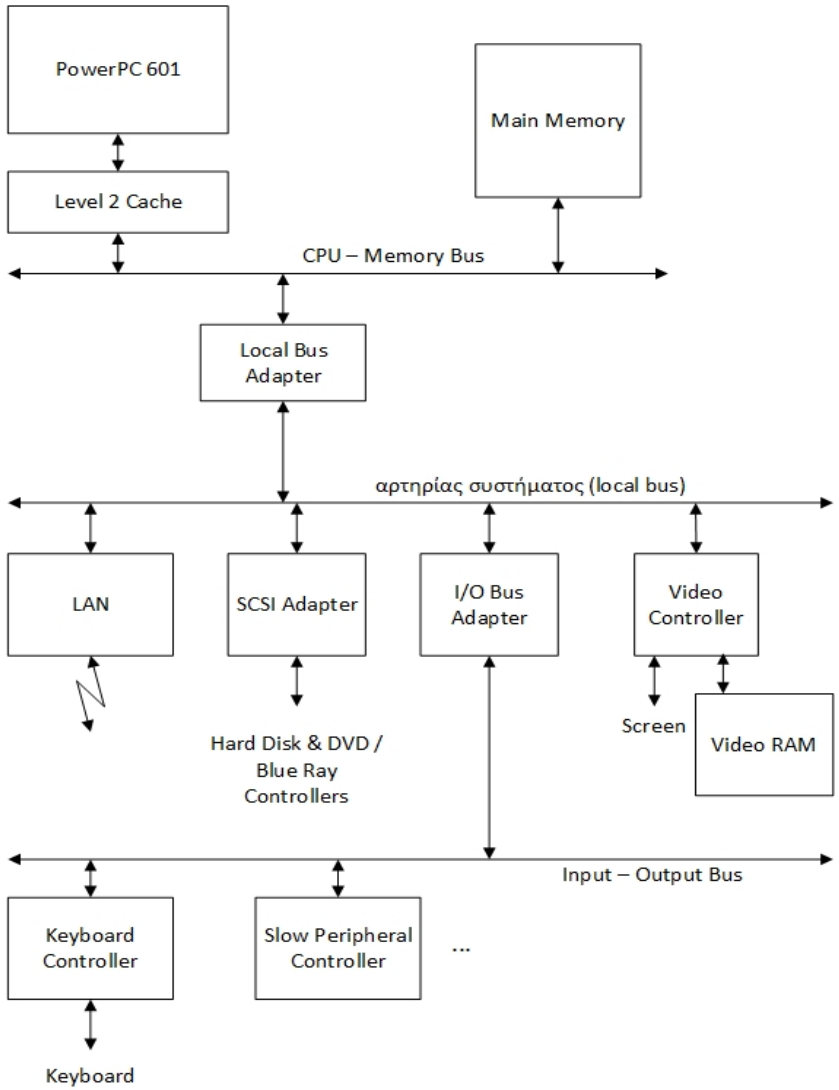
Παράδειγμα βασισμένο στον 21066



Παράδειγμα χαμηλών απαιτήσεων βασισμένο στον PowerPC



Παράδειγμα υψηλών απαιτήσεων βασισμένο στον PowerPC

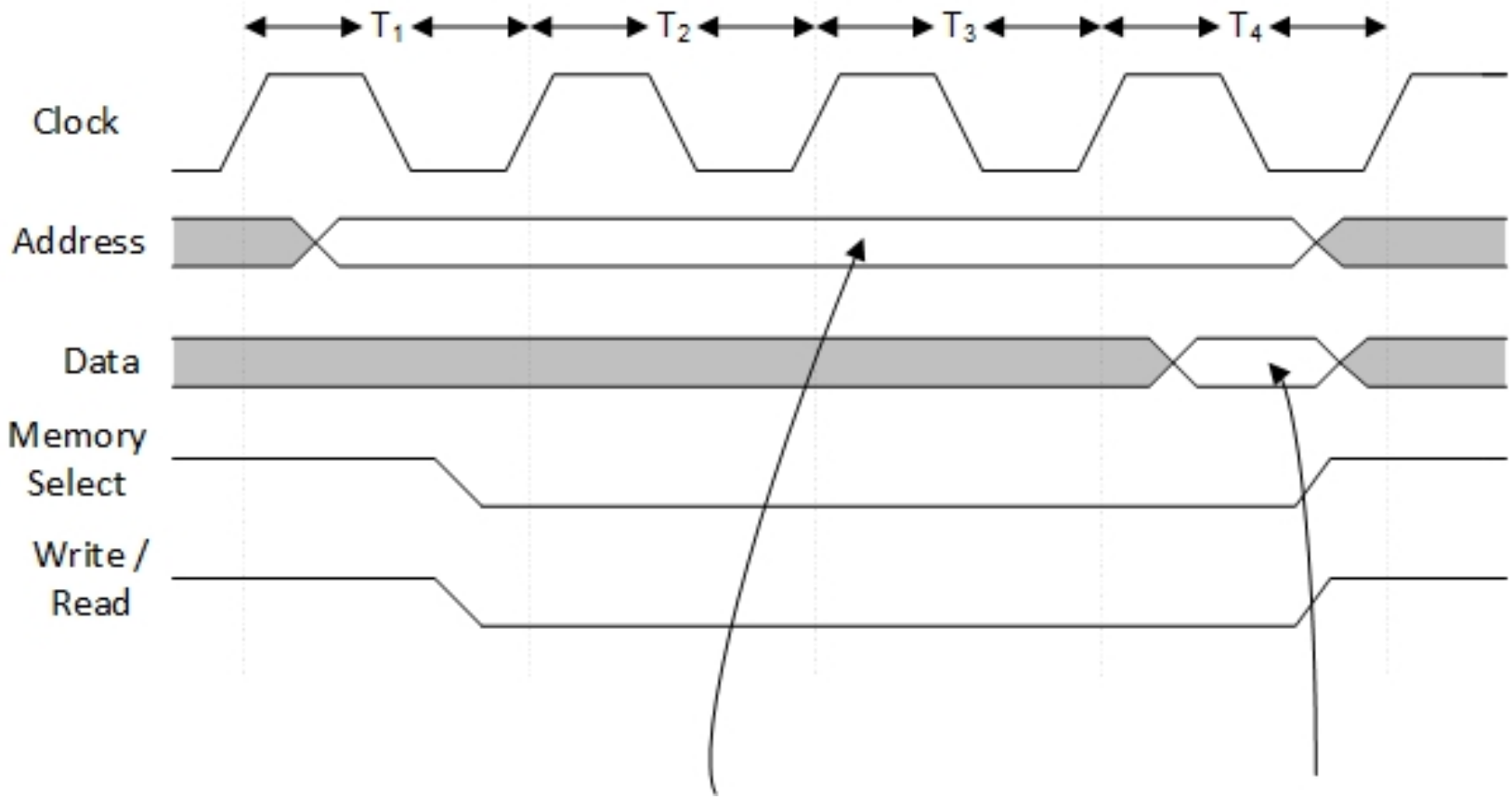


Ελεγκτές συσκευών

- Αποδεσμεύουν την ΚΜΕ από τις λεπτομέρειες λειτουργίας της συσκευής εισόδου/εξόδου
- Αποδεσμεύουν την αρτηρία από τις συγκεκριμένες απαιτήσεις διασύνδεσης της συσκευής εισόδου/εξόδου. Με τον τρόπο αυτό διευκολύνεται η αλλαγή της τεχνολογίας της συσκευής εισόδου/εξόδου
- Η διαχείριση της διαφοράς της ταχύτητας μεταξύ ΚΜΕ και κύριας μνήμης από την μια πλευρά και της μονάδας εισόδου/εξόδου από την άλλη με τη διάθεση κατάλληλου χώρου ενδιάμεσης αποθήκευσης πληροφορίας (buffering)
- Μετατροπή της μορφής ή της κωδικοποίησης των δεδομένων για συμμόρφωση με το πρωτόκολλο της αρτηρίας στην οποία συνδέονται.

Συγχρονισμός αρτηριών

- Δύο κατηγορίες: Σύγχρονες και ασύχρονες.
- Στις σύγχρονες κάποια γραμμή μεταδίδει ένα σήμα χρονισμού που χρονίζει τις ενέργειες του πρωτοκόλλου.



Σύγχρονες αρτηρίες

- Πλεονεκτήματα:
  - Μεγάλη ταχύτητα υπό όρους
  - Μικρό κόστος υλοποίησης του πρωτόκολλου

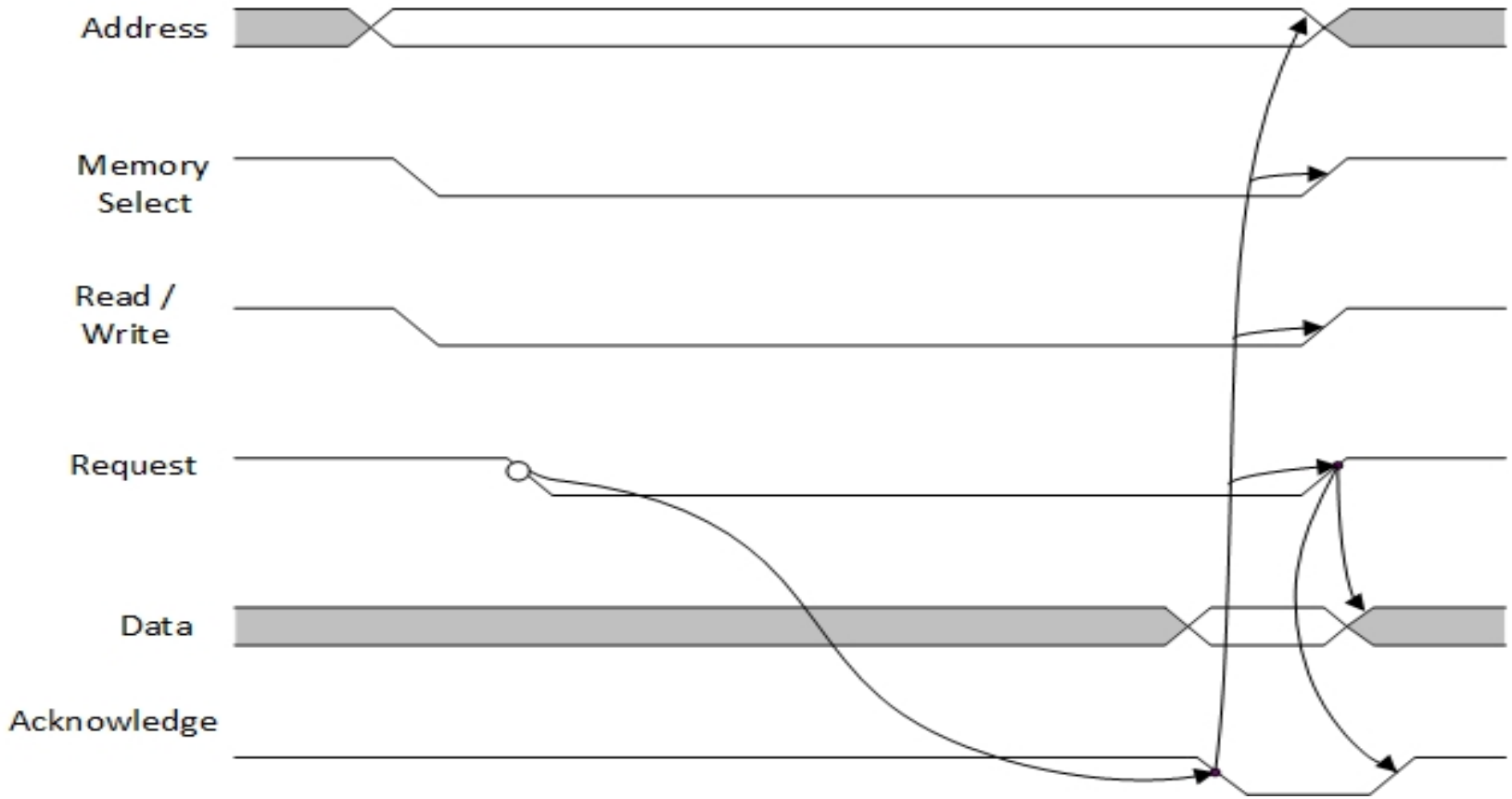
Σύγχρονες αρτηρίες

- Πλεονεκτήματα:
  - Μεγάλη ταχύτητα υπό όρους
  - Μικρό κόστος υλοποίησης του πρωτόκολλου
- Μειονεκτήματα:
  - Όλες οι μονάδες που είναι συνδεδεμένες στην αρτηρία πρέπει να χρησιμοποιούν το ίδιο σήμα χρονισμού
  - Για να είναι γρήγορες δεν μπορούν να έχουν μεγάλο μήκος
  - Για να είναι γρήγορες το πλήθος των μονάδων που συνδέεται σ' αυτές πρέπει να είναι μικρό
- Συνήθως η αρτηρία επεξεργαστή - μνήμης είναι σύγχρονη.

Σύγχρονες αρτηρίες με εισαγωγή κύκλων καθυστέρησης

- Οι διάφορες μονάδες εισόδου - εξόδου μπορεί να χρειάζονται διαφορετικούς κύκλους για την ολοκλήρωση μιας μεταφοράς.
- Αυτό σημαίνει ότι για κάθε μεταφορά θα πρέπει να περιμένουμε τον αριθμό κύκλων που απαιτούνται από τη πιο αργή μονάδα, καταλήγοντας σε μια πολύ αργή αρτηρία.
- Εναλλακτικά, μπορούμε επεκτείνοντας τα σήματα ελέγχου να δώσουμε στις μονάδες τη δυνατότητα να εισάγουν ατομικά έξτρα κύκλους.
- Αυτό γίνεται με την εισαγωγή κύκλων καθυστέρησης (wait cycles).

Ασύγχρονες αρτηρίες



- Πρωτόκολλο Χειραψίας (Handshaking) με Request - Acknowledge
- +: Διασύνδεση μονάδων διαφορετικής ταχύτητας (αρτηρίες εισόδου - εξόδου)

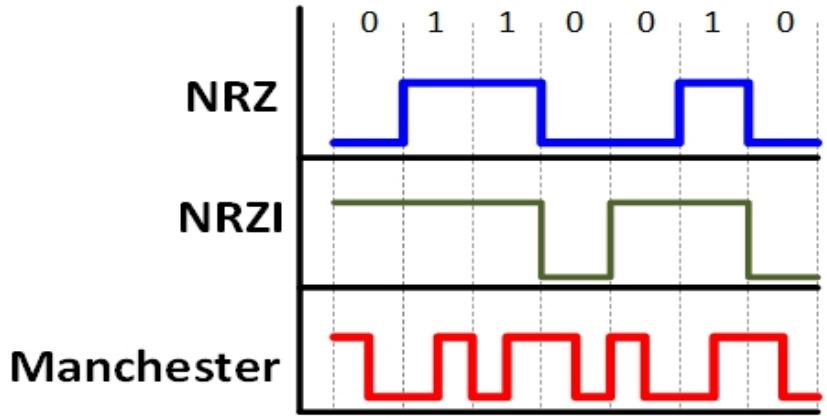
Διαιτησία

- Στις αρτηρίες κοινής χρήσης απαιτείται διαιτησία (arbitration), γιατί:
  - Η αρτηρία μπορεί να χρησιμοποιείται ήδη, ή/και
  - Την ίδια χρονική στιγμή απαιτούν την αρτηρία περισσότερες από μια μονάδες
- Υπάρχουν 3 πιθανά σχήματα διαιτησίας:
  - Διαιτησία με χρήση αλυσίδας προτεραιότητας (Daisy chain),
  - Κεντρική παράλληλη διαιτησία (Centralized Parallel), και
  - Κατανεμημένη διαιτησία (Distributed),
    - Με αυτοεπιλογή ή
    - Με ανίχνευση σύγκρουσης

Σηματοδοσία (1/2)

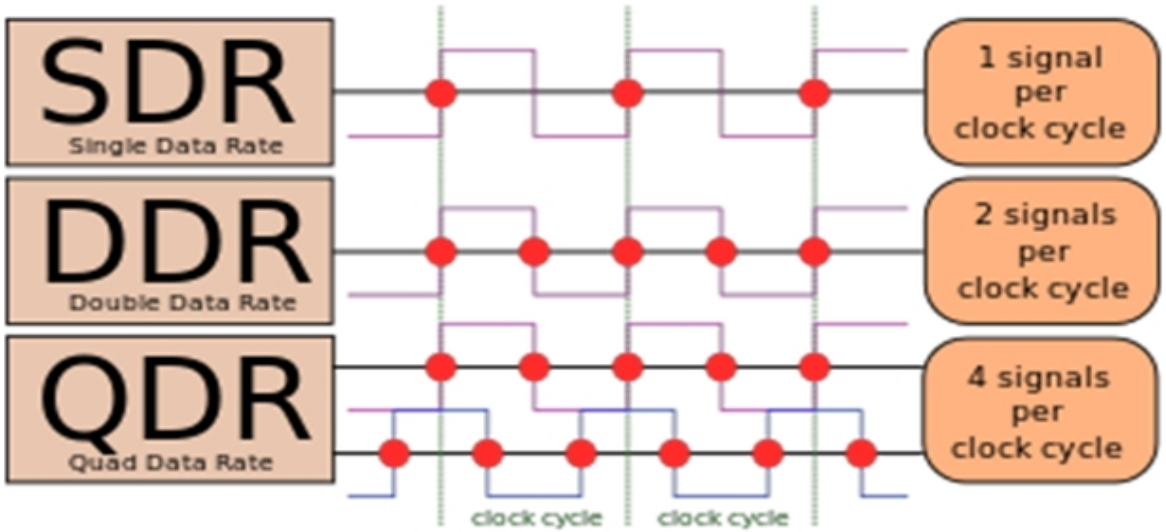
- Σηματοδοσία: Η μέθοδος κωδικοποίησης της ψηφιακής πληροφορίας πάνω στο φυσικό μέσο.
- Οι διάφορες μέθοδοι σηματοδοσίας διαφέρουν:
  - Στον αριθμό των γραμμών που χρησιμοποιούνται:
    - 1 γραμμή με full voltage swing (single ended)
    - 2 γραμμές με διαφορική κωδικοποίηση (differential) π.χ. SATA, XAUI, PCI Express, DDR SDRAM, USB 3.0
    - 4 γραμμές με διαφορική κωδικοποίηση (quad differential)
  - Στην κωδικοποίηση του 0 και του 1:

NRZ
NRZI
NRZI με bit stuffing
Manchester



Σηματοδοσία (2/2)

- Οι διάφορες μέθοδοι σηματοδοσίας διαφέρουν:
  - Στο ρυθμό σημάτων ανά χρονικό κύκλο:
    - SDR (Single Data Rate)
    - DDR (Double Data Rate)
    - QDR (Quad Data Rate)



Ταχύτητα Αρτηρίας

- Μέτρα απόδοσης:
  - Καθυστέρηση (Latency): f(μήκος αρτηρίας, πλήθος των μονάδων που είναι συνδεμένες με την αρτηρία)
  - Ρυθμός Μεταφοράς (Bandwidth): f(εύρος της αρτηρίας δεδομένων, ύπαρξη διακριτών γραμμών δεδομένων και διευθύνσεων, μεταφορά συνόλου από λέξεις)

Γενικά

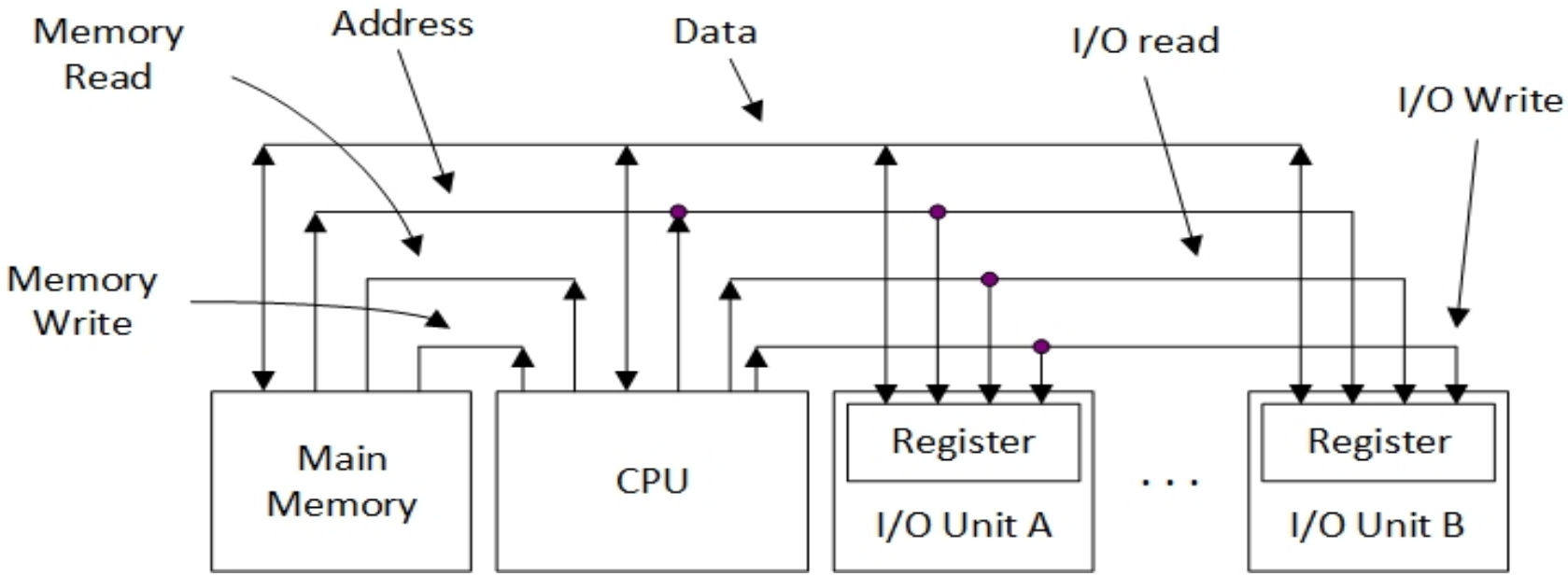
- Η ΚΜΕ έχει το γενικό έλεγχο επικοινωνίας με τις μονάδες I/O.
- Η επικοινωνία έγκειται στη διακίνηση:
  - Πληροφοριών ελέγχου, και
  - Δεδομένων
- Παρότι θα θέλαμε να αποφύγουμε και τα δύο λόγω της διαφοράς στη συχνότητα λειτουργίας της ΚΜΕ και των μονάδων I/O, το μόνο που μπορούμε να κάνουμε είναι να προσπαθήσουμε να μειώσουμε το βαθμό συμμετοχής της ΚΜΕ.

Πληροφορίες Ελέγχου

- Η διακίνηση πληροφοριών ελέγχου αφορά:
  - Αποστολή εντολών από την ΚΜΕ στις μονάδες I/O
  - Μεταφορά πληροφορίας στη ΚΜΕ σχετικά με την κατάσταση κάποιας μονάδας I/O, ή
  - Απαίτηση της μονάδας I/O για εξυπηρέτηση

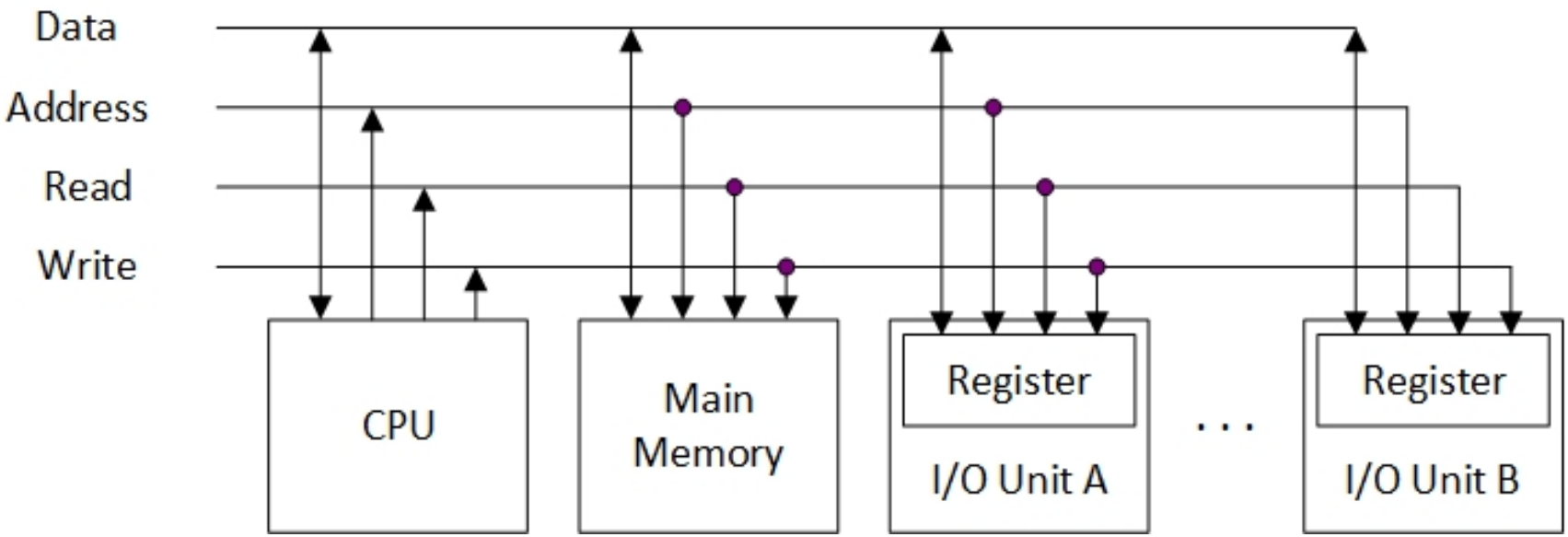
Προσεγγίσεις(1/2)

- Διαφορετικοί χώροι διευθύνσεων (Distinct Memory & I/O Spaces)
  - Ειδικές εντολές για είσοδο / έξοδο
  - Σχήμα που υιοθετείται από όλους τους Intel επεξεργαστές



Προσεγγίσεις(2/2)

- Ενιαίος χώρος διευθύνσεων (Memory Mapped I/O)
  - Οι καταχωρητές των μονάδων I/O αντιστοιχίζονται σε διευθύνσεις μνήμης.
  - Σχήμα που υιοθετείται από όλους τους Motorola επεξεργαστές



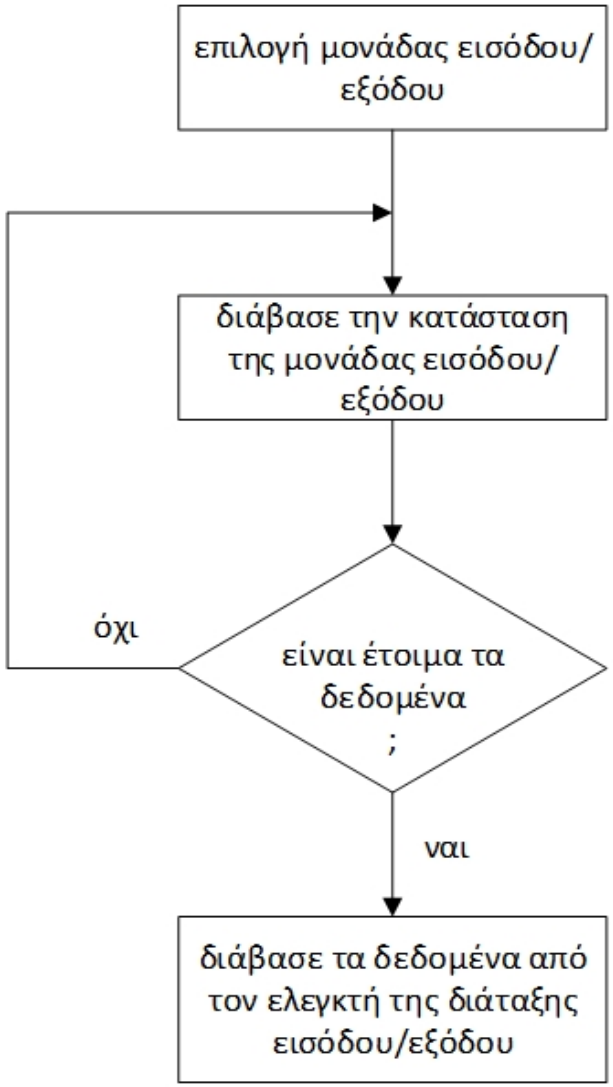
Ανακάλυψη της κατάστασης / ανάγκης εξυπηρέτησης

- Για να πληροφορηθεί η ΚΜΕ την κατάσταση ή την ανάγκη για εξυπηρέτηση από μια μονάδα I/O, μπορεί να χρησιμοποιείται:
  - Χρονοπρογραμματισμένος έλεγχος (polling).
    - Συχνότητα polling $\geq$ συχνότητα αλλαγής κατάστασης.
    - ΚΜΕ δαπανά άσκοπα χρόνο για το διάβασμα του καταχωρητή κατάστασης μονάδων χωρίς αλλαγή κατάστασης.
    - Ασύμφορο για πολλές μονάδες I/O ή για γρήγορες μονάδες.
  - Ενημέρωση της ΚΜΕ μέσω σημάτων διακοπής (interrupts).
    - Αποθήκευση του περιβάλλοντος
    - Interrupt Service Routine
    - Maskable vs non-maskable
    - Προτεραιότητα & nested interrupts
    - Polling vs vectored interrupts
    - Διαχειριστής διακοπών (Interrupt Controller)

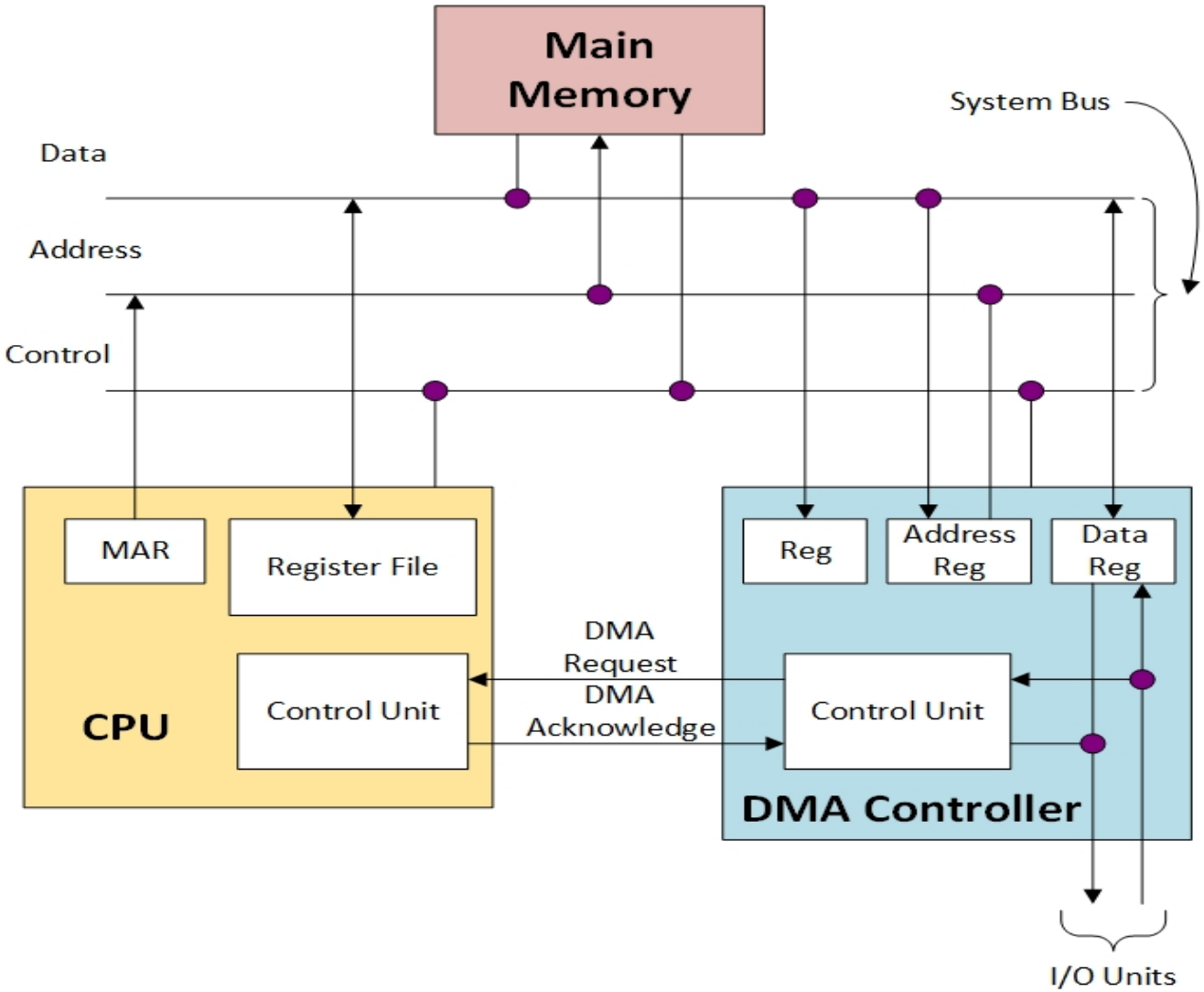
Μεταφορά Δεδομένων

- Τρεις περιπτώσεις για τη συμμετοχή της CPU στη μεταφορά δεδομένων:
  - I/O μέσω εκτέλεσης προγράμματος (programmed I/O)
  - I/O μέσω διακοπών και εκτέλεσης προγράμματος (Interrupt driven I/O)
  - I/O μέσω διακοπών και Ελεγκτή Άμεσης Προσπέλασης Μνήμης (Interrupt & DMA driven I/O)
    - Απλές μεταφορές μέσω cycle stealing
    - Πολλαπλές μεταφορές μετά από αίτηση για κυριότητα στην αρτηρία CPU - μνήμης

Programmed I/O



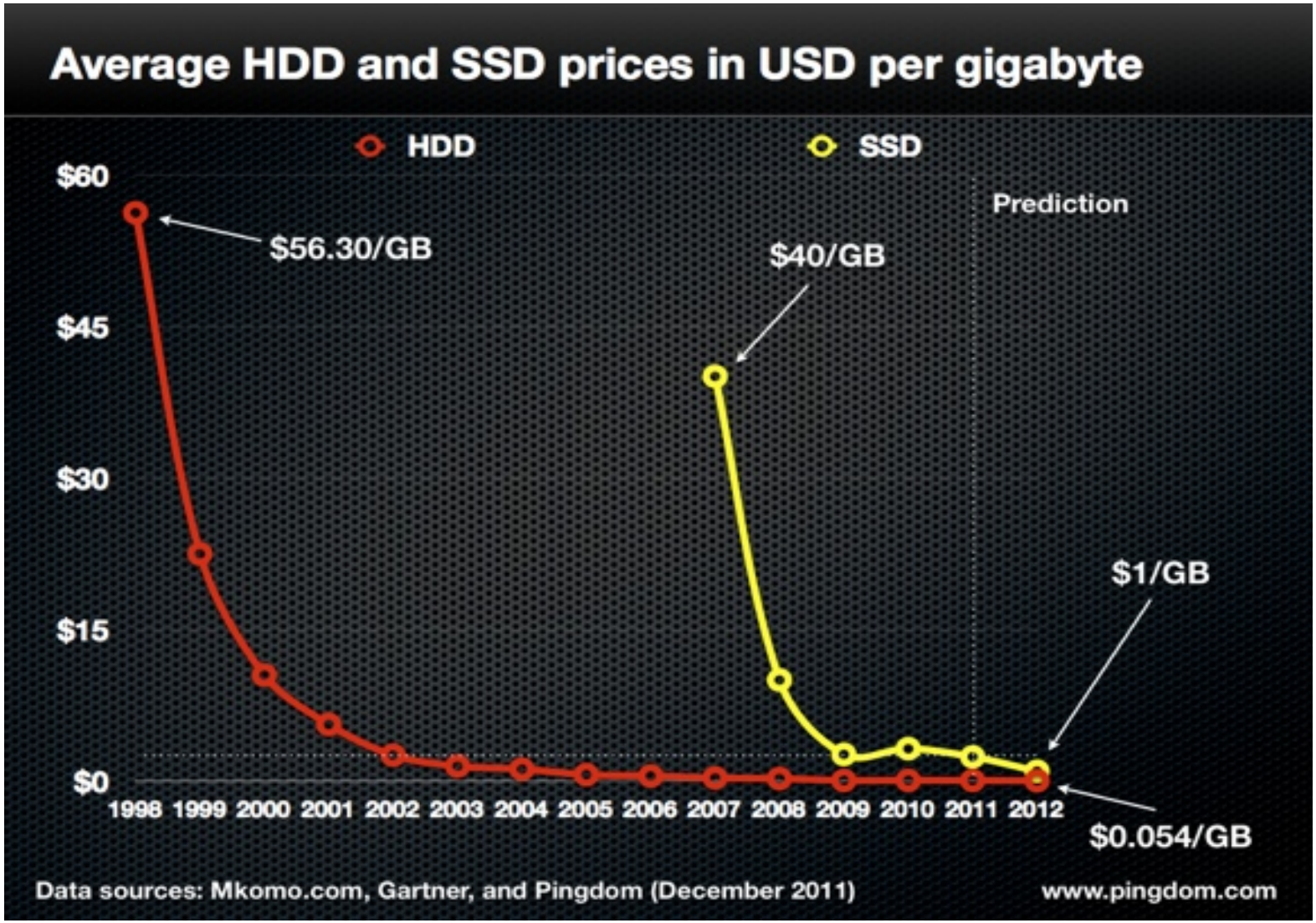
DMA driven I/O



Μονάδες αποθήκευσης

- Για την αποθήκευση μεγάλου όγκου πληροφοριών, χρησιμοποιήθηκαν / χρησιμοποιούνται:
  - Διάτρητες Κάρτες (Punch Cards)
  - Δισκέττες (Floppy Disks)
  - Μαγνητικές Ταινίες (Tapes)
  - **Μαγνητικοί δίσκοι (Hard Disks)**
  - Compact Discs (CDs)
  - USB Flash Drives
  - DVD and Blu-ray Discs
  - Secure Digital Cards (SD Card)
  - Δίσκοι με ημιαγωγικά στοιχεία δίσκοι (Solid-State Drives - SSDs)
  - Αποθηκευτικά Νέφη (Cloud Storage)
- Οι μαγνητικοί δίσκοι συνεχίζουν και στις μέρες μας να είναι η κύρια μορφή αποθήκευσης λόγω του κόστους / δυαδικό ψηφίο που προσφέρουν.

HDD vs SSD

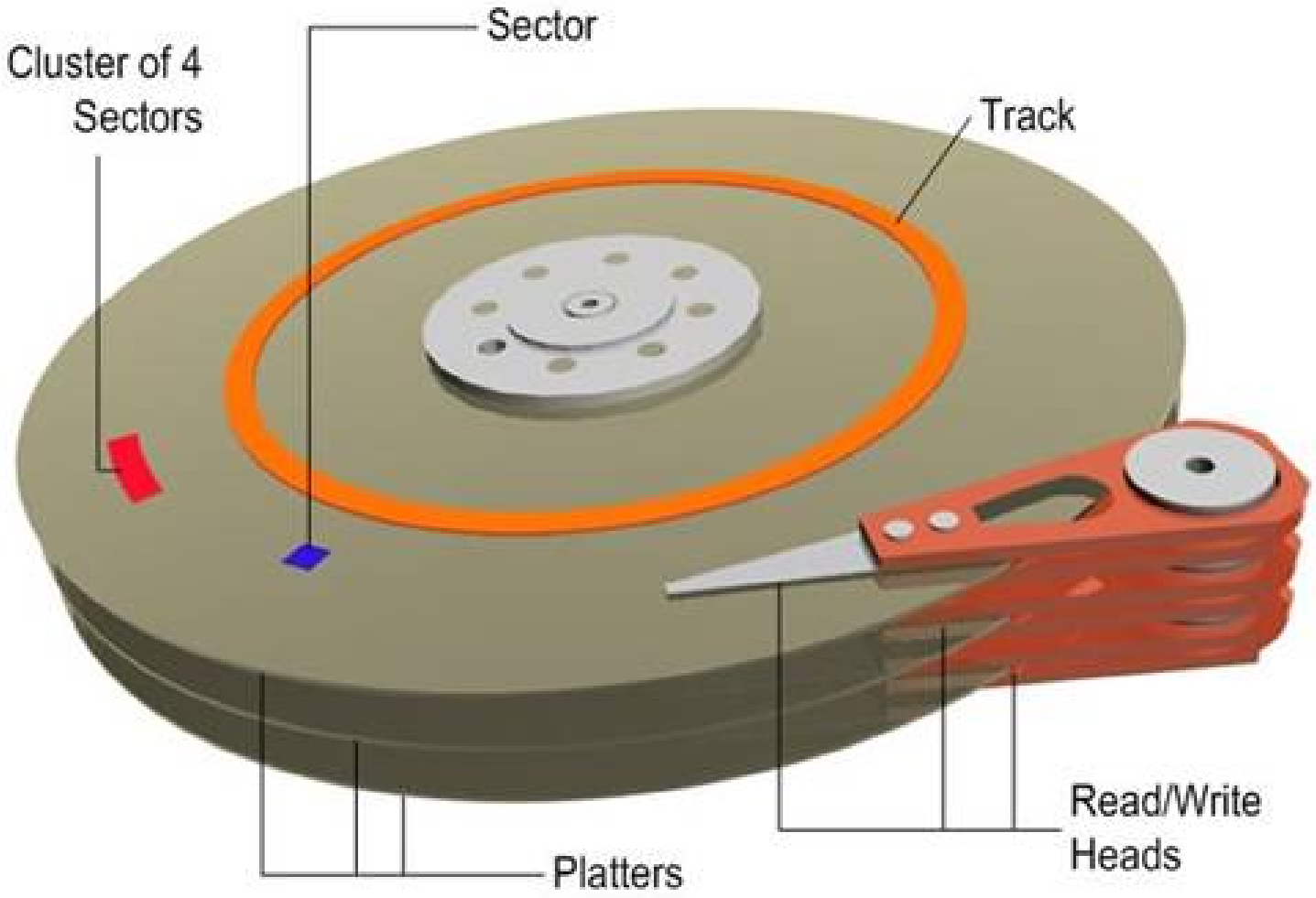

Average HDD and SSD prices in USD per gigabyte
HDD
SSD
Prediction
$60
$45
$30
$15
$0
$56.30/GB
$40/GB
$1/GB
$0.054/GB
1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
Data sources: Mkomo.com, Gartner, and Pingdom (December 2011)
www.pingdom.com

Γενικά για τους μαγνητικούς δίσκους

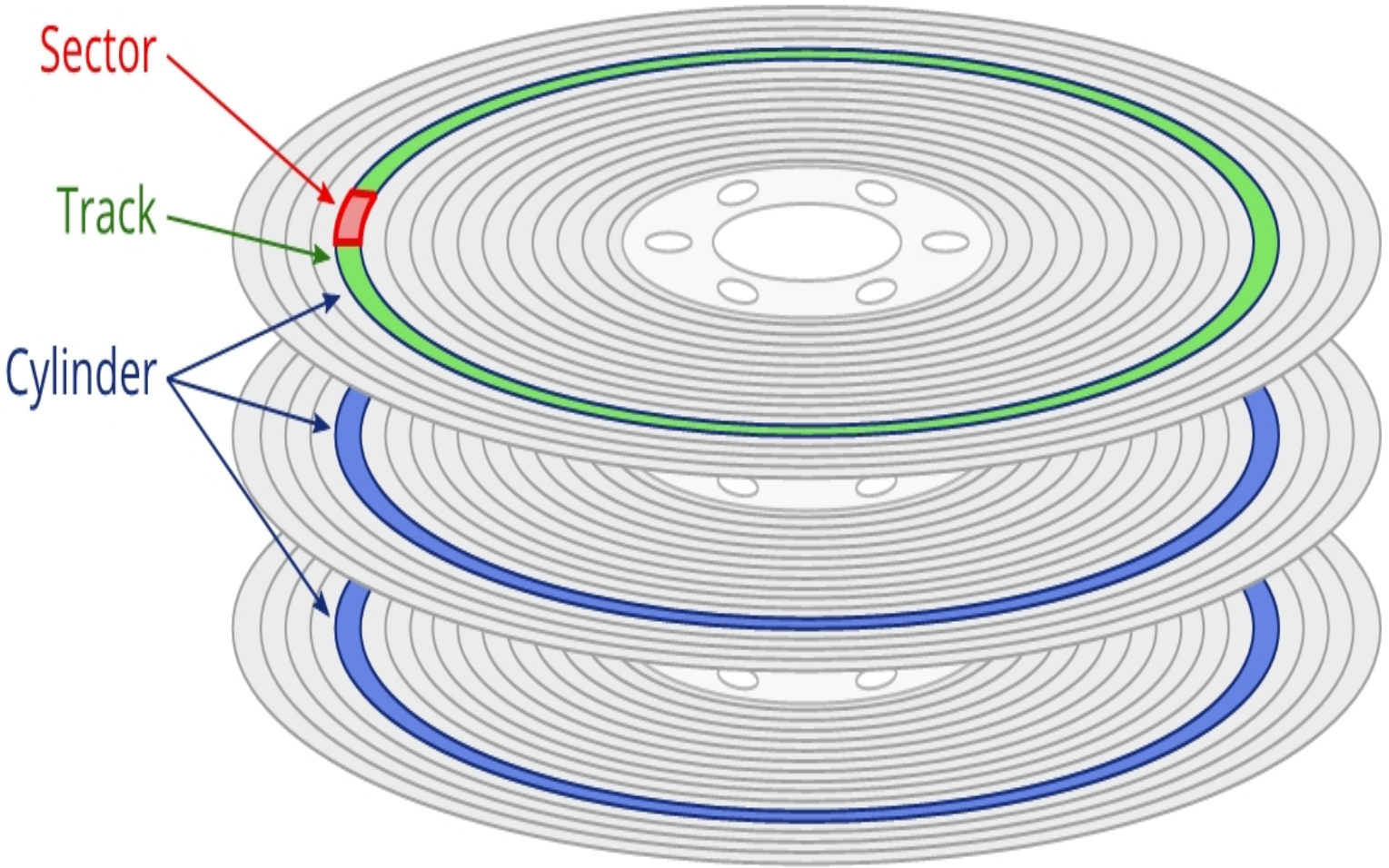
- Στους μαγνητικούς δίσκους χρησιμοποιείται μια διαδικασία μαγνήτισης μικροσωματιδίων μαλακού σιδήρου που έχουν εναποτεθεί στις επιφάνειες αποθήκευσης.
- Η παρουσία μαγνήτισης υποδεικνύει το 1 και η απουσία της το 0.
- Η μαγνήτιση ενός δίσκου έχει μια μέση διάρκεια ζωής γύρω στα 20 έτη.
- Συστατικά του δίσκου:
  - **Πλατώ (platters)**
  - **Επιφάνειες μαγνήτισης (surfaces)** - 2 ανά πλατώ.
  - **Κεφαλές ανάγνωσης / εγγραφής (heads)**. Υπάρχει τουλάχιστον μία ανά επιφάνεια.
  - Βραχίονας στήριξης των κεφαλών.
- Οργάνωση της πληροφορίας:
  - **Σε ομόκεντρους κύκλους (tracks)**. Τα tracks υπάρχουν μόνο κοντά στην εξωτερική περίμετρο του δίσκου, ώστε να έχουν το δυνατόν ίδιο μήκος.
  - **Σε κυκλικούς τομείς (sectors)**. Ο sector είναι η ελάχιστη ποσότητα πληροφορίας που μπορεί να διαβάσει / γράψει κάποιος από/προς το δίκο. Το μέγεθος του τομέα καθορίζεται κατά την αρχικοποίηση (format) του δίσκου.
  - **Σε κυλίνδρους (cylinders)**

Εσωτερικά σε ένα δίσκο

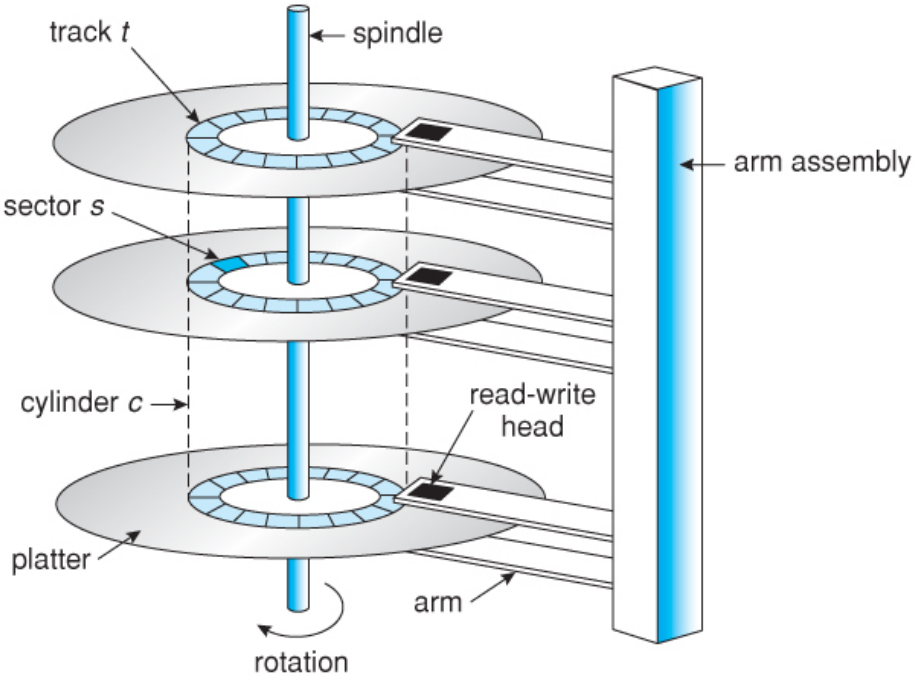
Συστατικά ενός δίσκου

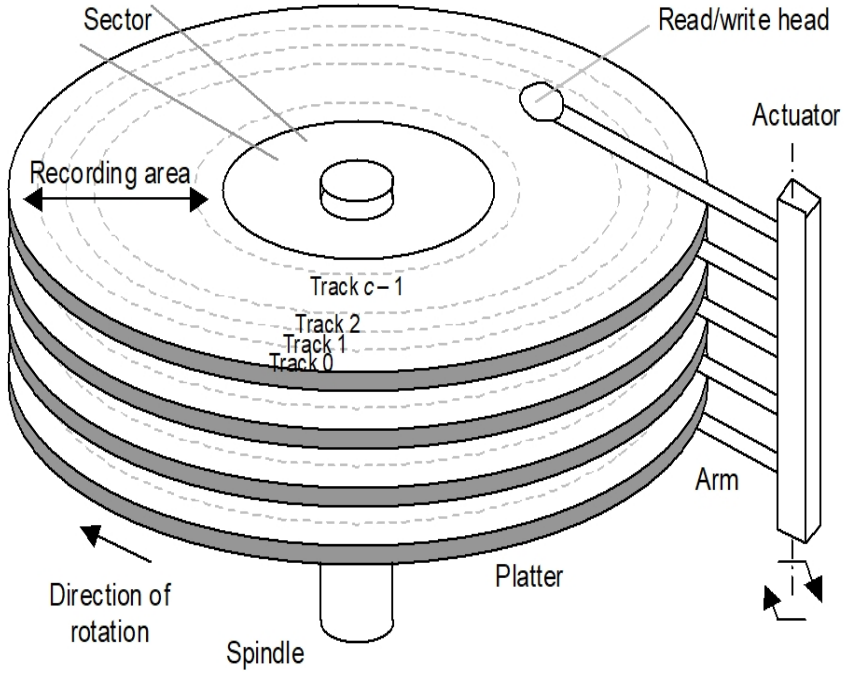


Οργάνωση πληροφορίας σε ένα δίσκο



Συνολική διάρθρωση ενός δίσκου





Χρόνος προσπέλασης πληροφορίας

- Ο χρόνος προσπέλασης μια πληροφορίας από το δίσκο εξαρτάται:
  - Από τον αριθμό κεφαλών ανά επιφάνεια
  - Από την ταχύτητα περιστροφής. Συνήθως μετριέται σε rotarions per minute (RPM) με τυπικές τιμές 5.400, 7.200 και 10.000
  - Από την οργάνωση της πληροφορίας στο δίσκο.
  - Από τον κερματισμό (fragmentation της πληροφορίας).

Δίσκος με μία κεφαλή ανά επιφάνεια

- Ο χρόνος προσπέλασης αποτελείται από:
  - Το χρόνο μετακίνησης της κεφαλής στο σωστό track: **Χρόνος αναζήτησης (seek time)**
  - Το χρόνο περιστροφής έως ότου η κεφαλή βρεθεί στην αρχή του sector: **Χρόνος αναμονής (rotational delay)**. Η μέση τιμή του είναι ίση με το χρόνο μισής περιστροφής.
  - Το χρόνο ανάγνωσης του sector: **Χρόνος ανάγνωσης (data transfer Time)**. Η μέση τιμή του είναι ίση με το χρόνο 1 περιστροφής δια τον αριθμό των sectors / track ή ισοδύναμα τον αριθμό των κυλίνδρων.

Συνιστώσες της καθυστέρησης

Data transfer time = Bytes / Data rate

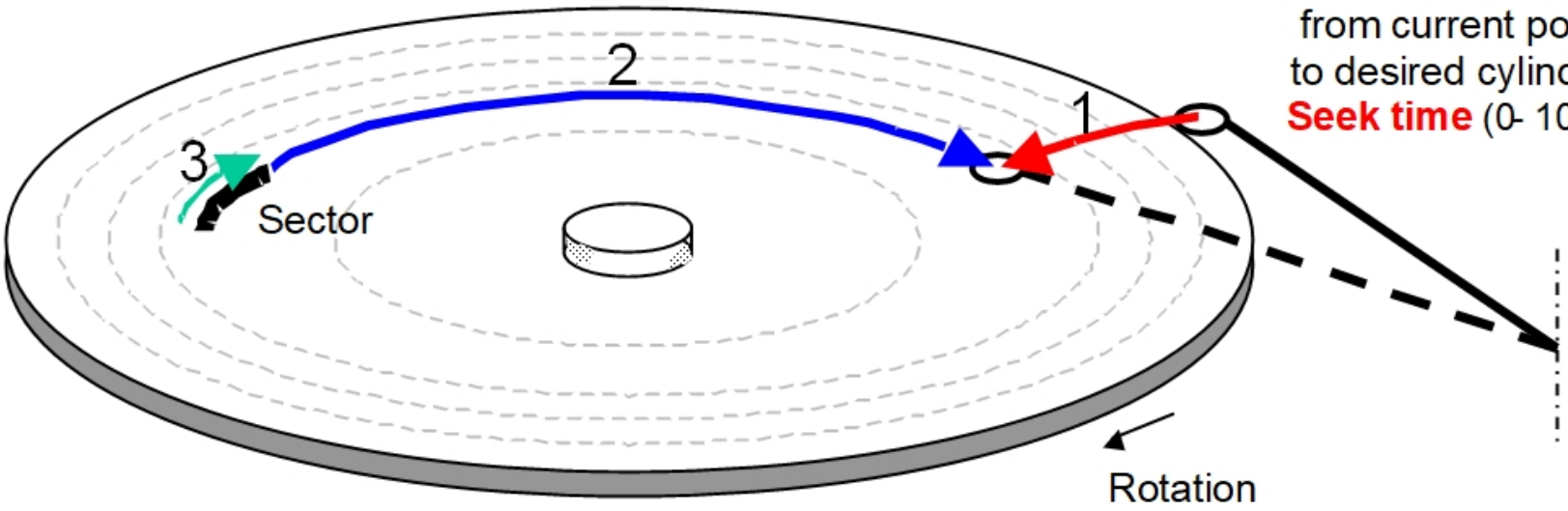
Average rotational latency = 30 000 / rpm (in ms)

Seek time = $a + b(c-1) + \beta(c-1)^{1/2}$

2. Disk rotation until the desired sector arrives under the head: **Rotational latency** (0- 10s ms)

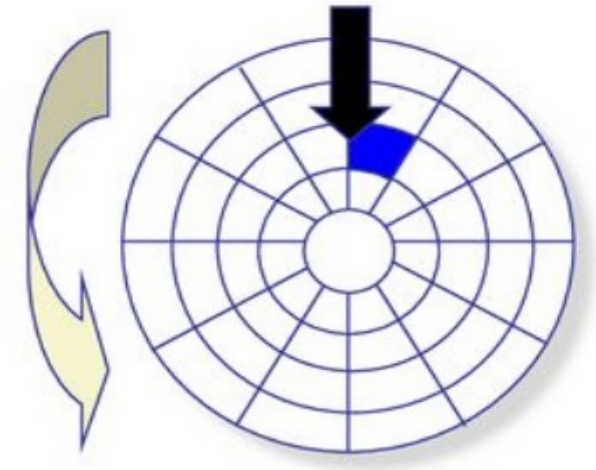
3. Disk rotation until sector has passed under the head: **Data transfer time** (< 1 ms)

1. Head movement from current position to desired cylinder: **Seek time** (0- 10s ms)



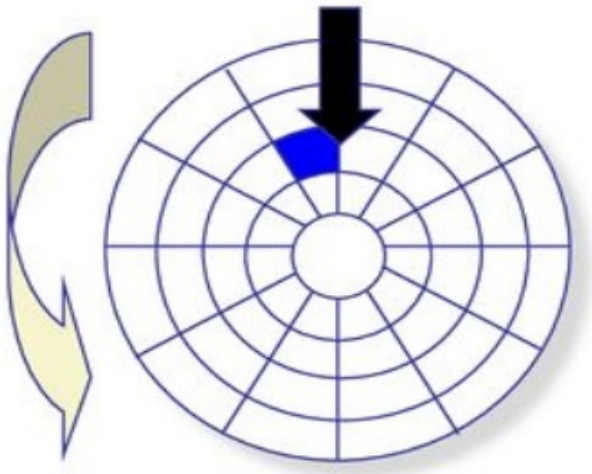
The three components of disk access time. Disks that spin faster have a shorter average and worstcase access time.

Παράδειγμα ανάγνωσης 2 sectors (1/7)



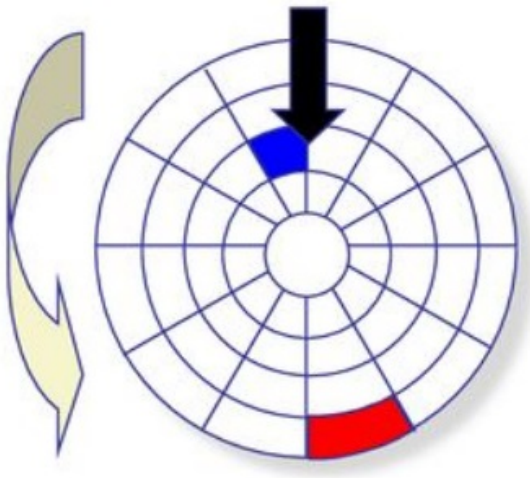
About to read blue sector

Παράδειγμα ανάγνωσης 2 sectors (2/7)



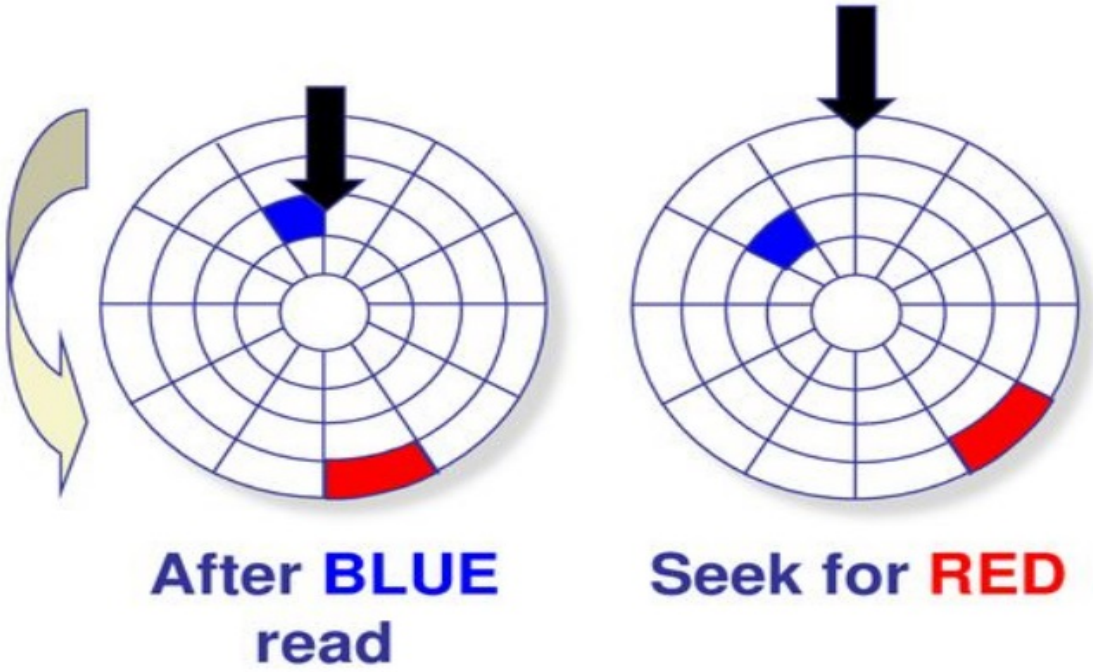
After reading blue sector

Παράδειγμα ανάγνωσης 2 sectors (3/7)

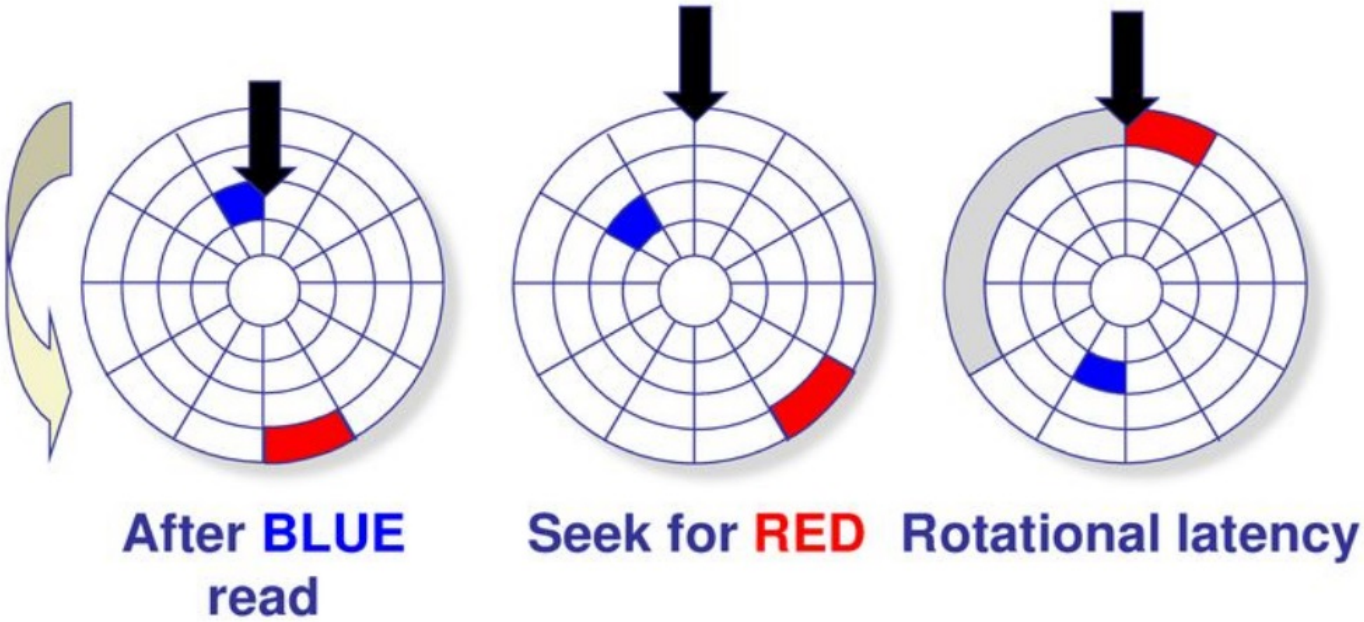


Red request scheduled next

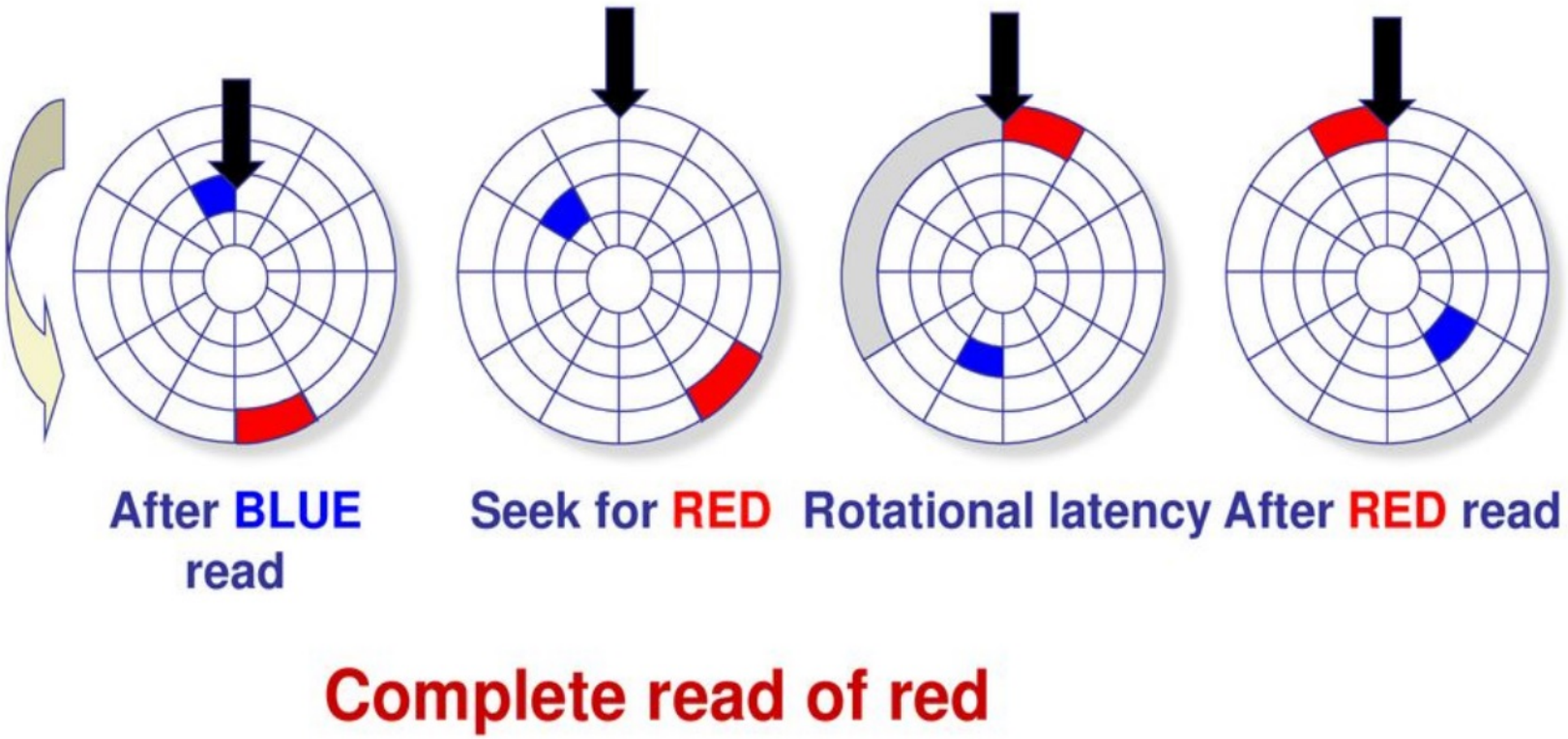
Παράδειγμα ανάγνωσης 2 sectors (4/7)



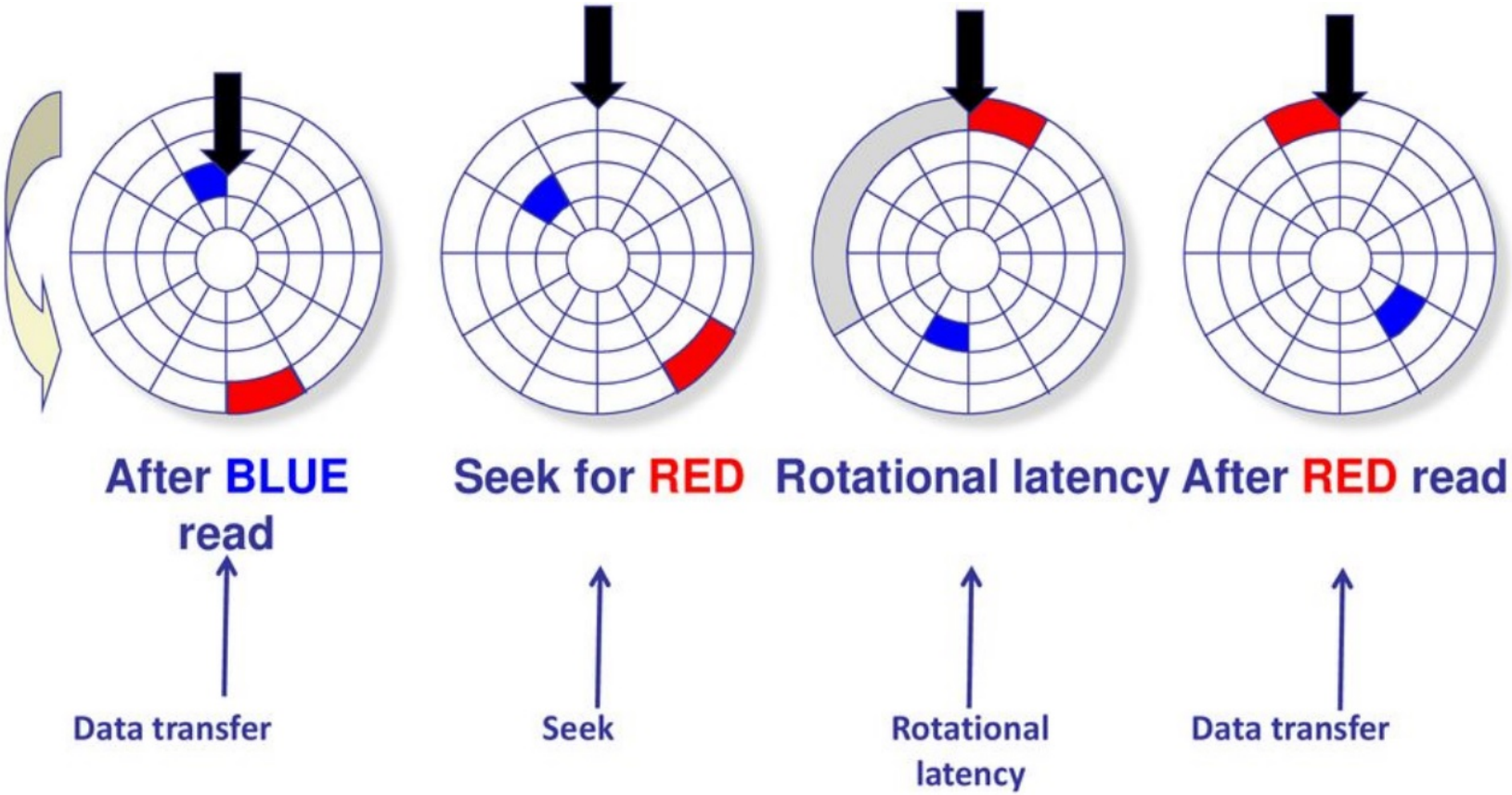
Παράδειγμα ανάγνωσης 2 sectors (5/7)



Παράδειγμα ανάγνωσης 2 sectors (6/7)



Παράδειγμα ανάγνωσης 2 sectors (7/7)



Μερικά αξιοσημείωτα

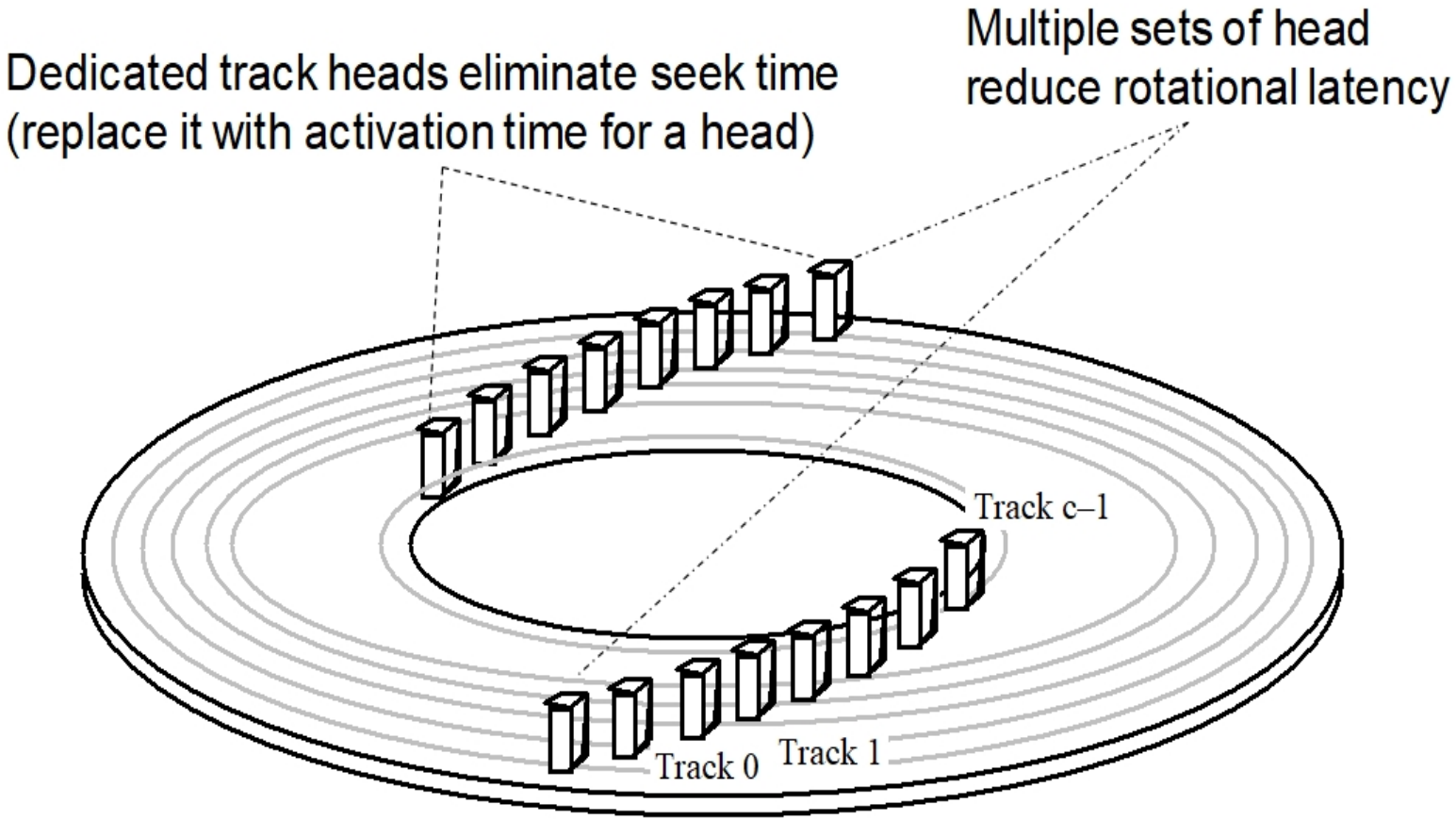
- Οι κυρίαρχες συνιστώσες του χρόνου προσπέλασης ενός δίσκου με μία κεφαλή ανά επιφάνεια είναι ο χρόνος αναζήτησης και ο χρόνος αναμονής.
- Η προσπέλαση του πρώτου bit ενός sector του δίσκου κοστίζει πολύ σε χρόνο, ενώ η προσπέλαση των υπολοίπων είναι δωρεάν.
- Στις μνήμες ισχύει:
  - Μπορώ να προσπελάσω 64 bit πληροφορίας από μια SRAM σε περίπου 4ns.
  - Μπορώ να προσπελάσω 64 bit πληροφορίας από μια DRAM σε περίπου 60ns.

Άρα η προσπέλαση ενός δίσκου είναι 40.000 πιο αργή από τη προσπέλαση μιας SRAM και 2.500 φορές πιο αργή από μιας DRAM.

Μείωση του χρόνου προσπέλασης

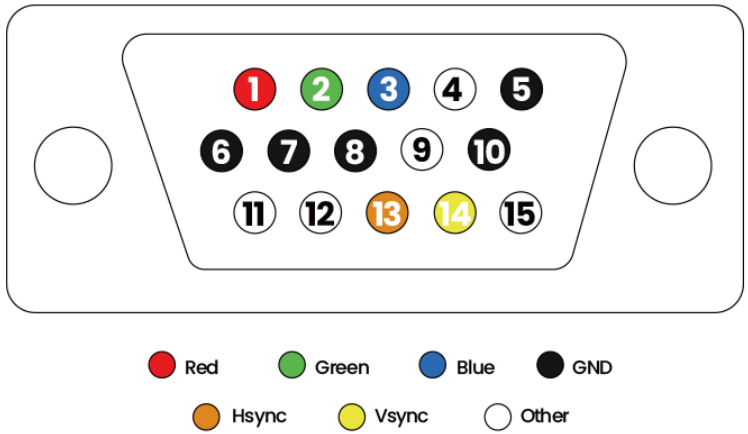
- Ο χρόνος αναζήτησης μπορεί να:
  - Μειωθεί με την ύπαρξη δύο ή περισσότερων κεφαλών.
  - Εξαλειφθεί με την ύπαρξη μιας κεφαλής / track (fixed head disks)
- Ο χρόνος αναμονής μπορεί να μειωθεί με την ύπαρξη δύο συνόλων κεφαλών τοποθετημένων αντιδιαμετρικά.
- Ο χρόνος ανάγνωσης μπορεί να μειωθεί με την παράλληλη ανάγνωση από όλους τους sectors ενός κυλίνδρου.

Δίσκος με 2 σετ σταθερών κεφαλών



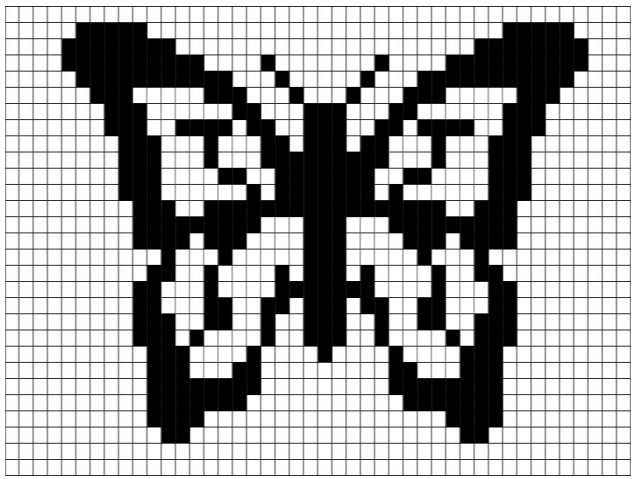
Γενικά

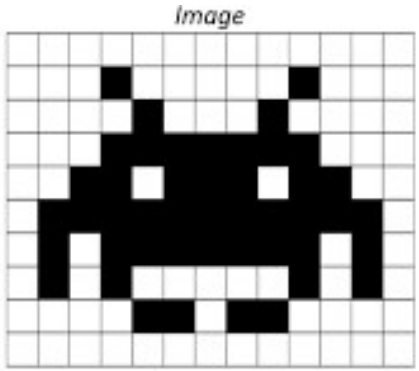
- Η οθόνη είναι μια συσκευή εξόδου.
- Παρότι υπάρχουν διάφορα πρωτόκολλα για την οδήγηση μιας οθόνης, εμείς θα επικεντρώσουμε στο Video Graphics Adapter (VGA).
- Παρουσιάστηκε το 1987 από την IBM και χρησιμοποιεί έν μέρει ψηφιακά (ΟΧΙ δυαδικά) σήματα.
- Η σύνδεση γίνεται με βύσμα 15 ακροδεκτών:

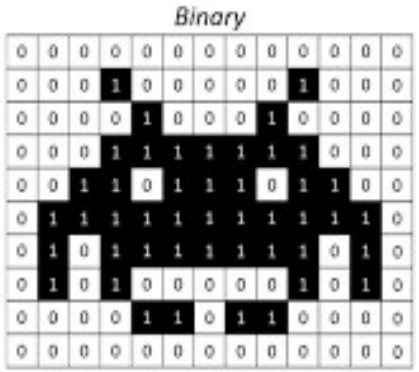


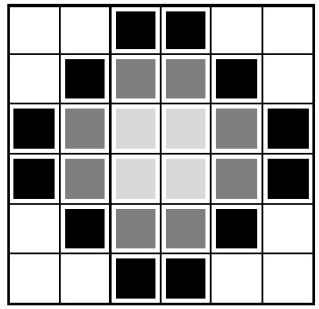
Απεικονιζόμενη εικόνα

- Η απεικονιζόμενη εικόνα (frame - καρέ) ανανεώνεται με ρυθμό τουλάχιστον 30 φορές το δευτερόλεπτο ώστε το ανθρώπινο μάτι να έχει την αίσθηση της ομαλούς μετάβασης και κίνησης.
- Η εικόνα αποτελείται από πολύ μικρές κουκίδες, τα εικονοστοιχεία (picture elements - pixels).
- Αν η εικόνα μας είναι ασπρόμαυρη (ή μονοχρωματική) - binary image για κάθε pixel αρκεί ένα δυαδικό ψηφίο για να καθορίσει το εάν αυτό θα είναι στη μία κατάσταση ή στην άλλη.
- Για αποχρώσεις του γκρι, χρειαζόμαστε $\lceil \log_2(\text{διαβαθμίσεις του γκρι}) \rceil$



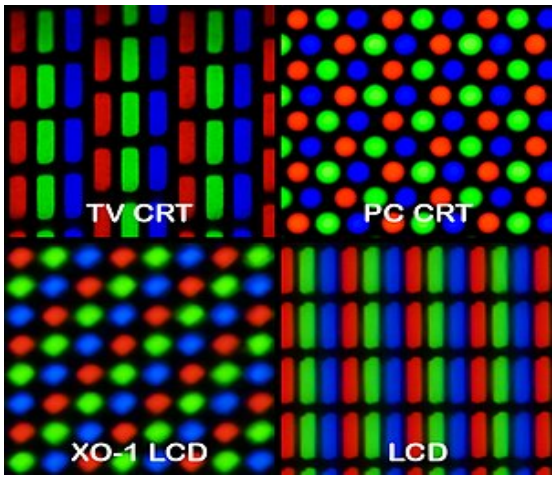


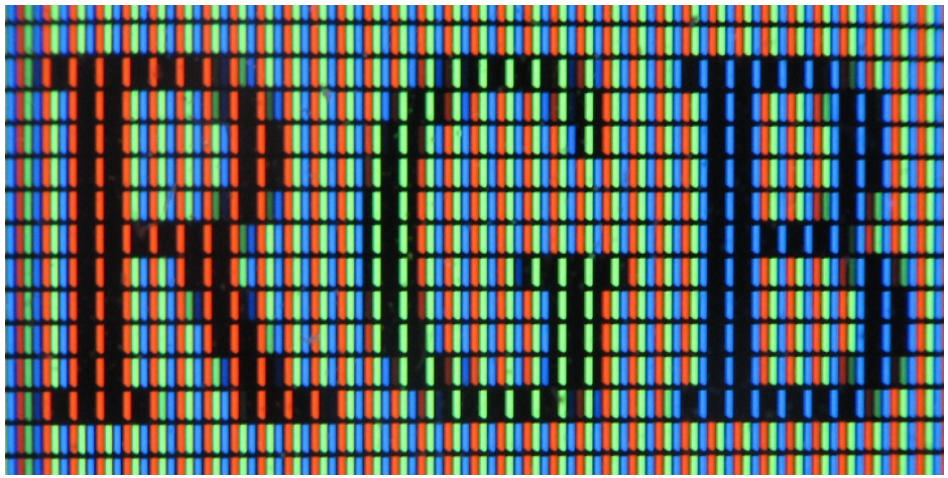




```
11 11 00 00 11 11
11 00 01 01 00 11
00 01 10 10 01 00
00 01 10 10 01 00
11 00 01 01 00 11
11 11 00 00 11 11
```

Χρώμα

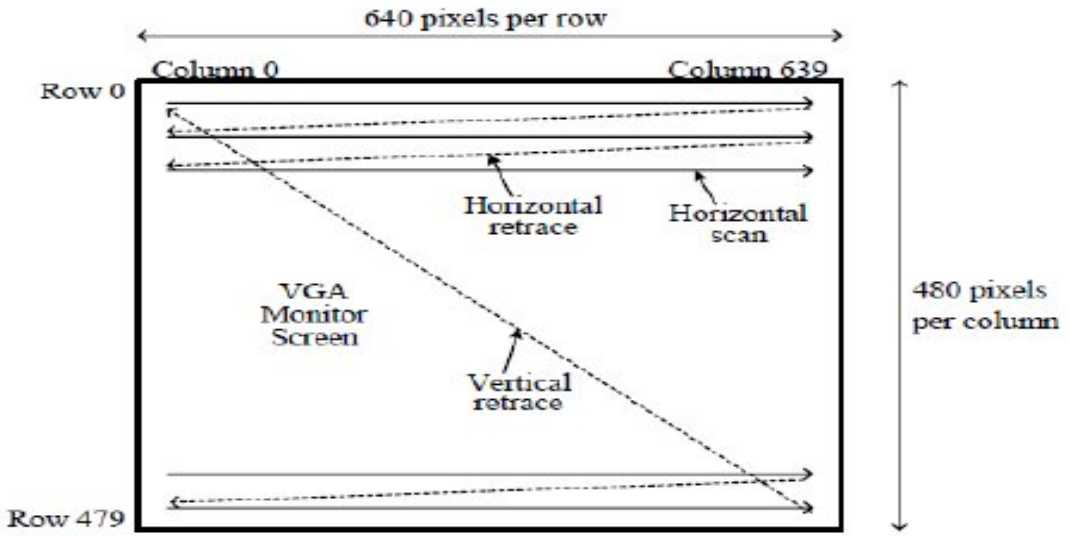


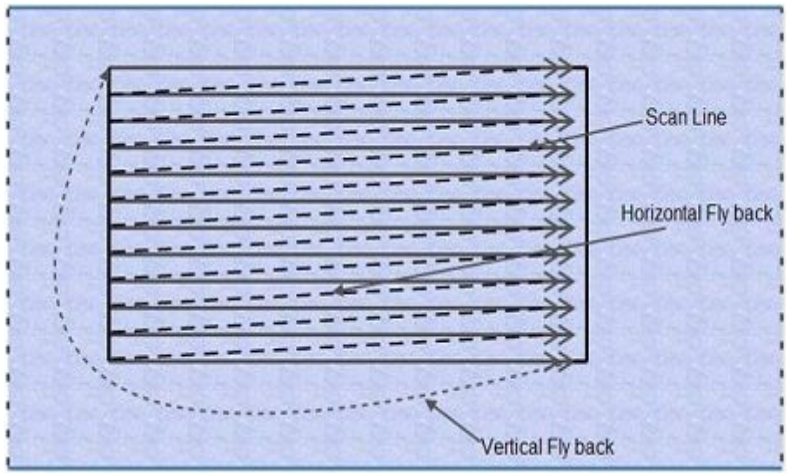


- Για την απεικόνιση χρώματος, οι σύγχρονες οθόνες (τηλεόρασης & υπολογιστών) διαθέτουν για κάθε pixel τρεις μικροσκοπικές λυχνίες (tubes). Η διάταξή τους διαφέρει σε κάθε συσκευή.
- Κάθε μία είναι ικανή να αναπαράγει ένα από τα χρώματα του συνόλου βάσης {Red, Green, Blue} (RGB).
- Η σύνθεση αυτών των χρωματικών συνιστωσών είναι το απεικονιζόμενο χρώμα.
- Οι προσφερόμενες διαβαθμίσεις των βασικών χρωμάτων καθορίζουν το βάθος χρώματος (color depth). Για παράδειγμα αν υπάρχουν 256 διαβαθμίσεις κάθε βασικού χρώματος (ισοδύναμα, αν για κάθε βασική χρωματική συνιστώσα διατίθενται 8 bit) το βάθος χρώματος είναι 24 bit και τα πιθανά χρώματα για κάθε εικονοστοιχείο είναι $2^{24} = 16.777.216_{10}$

Σάρωση (scan)

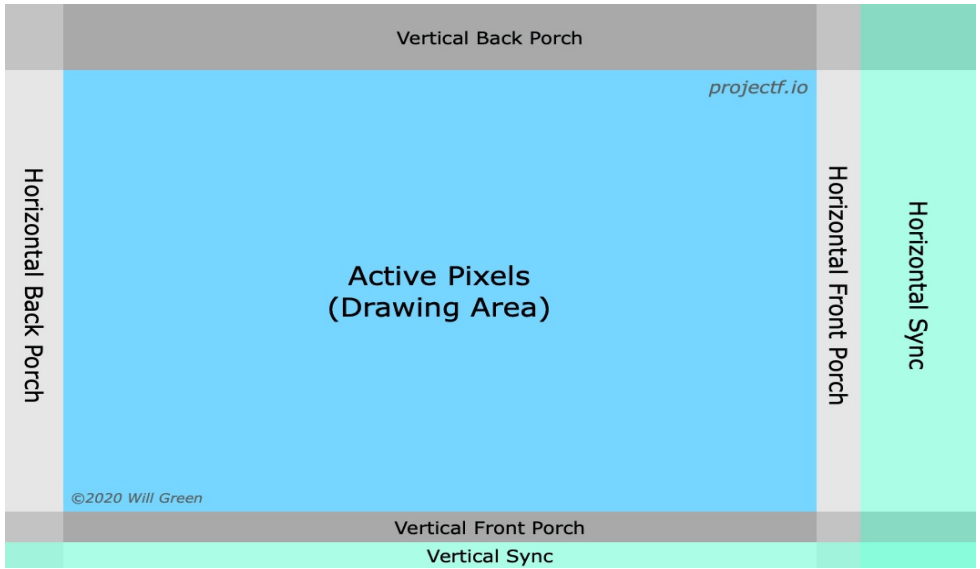
- Κάθε μικροσκοπική λυχνία ανάβει μέσω μιας ή περισσοτέρων δεσμών ηλεκτρονίων που σαρώνουν (scan) την περιοχή απεικόνισης.
- Θα πρέπει να ολοκληρώνονται τουλάχιστον 30 σαρώσεις καρέ το δευτερόλεπτο.
- Σημερινές τυπικές τιμές του ρυθμού ανανέωσης (refresh rate): 60Hz / 100Hz.





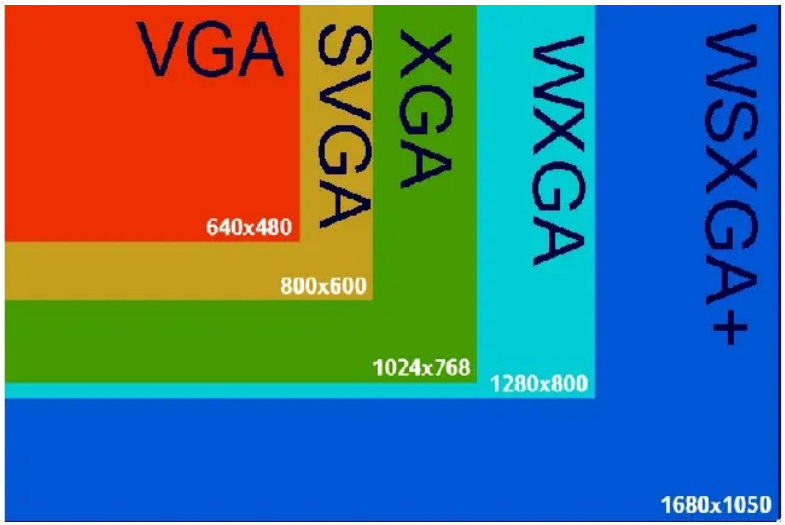
Αμαύρωση (Blanking)

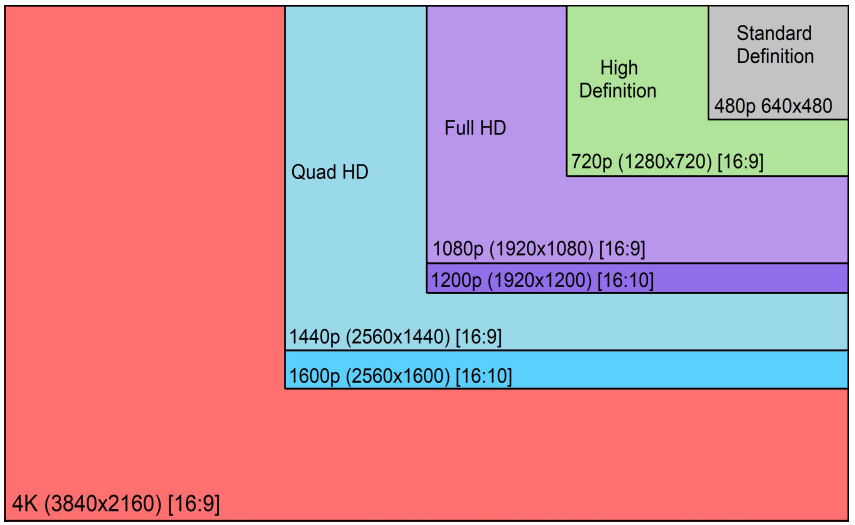
- Πριν τη σάρωση, απαιτείται απομάκρυνση του υπολειπόμενου φορτίου από την προηγούμενη σάρωση.
- Αυτό γίνεται με την αμαύρωση (blanking) που γίνεται πριν τη σάρωση μιας γραμμής ή ενός καρέ.
- Κατά τον ίδιο χρόνο η δέσμη σάρωσης μετακινείται στην επόμενη γραμμή ή στην κορυφή της εικόνας (retrace / fly back).
- Συνεπώς δε σαρώνονται μόνο τα ορατά pixels ανά γραμμή και οι ορατές γραμμές ανά καρέ αλλά ένα υπερσύνολο αυτών.



Ανάλυση (Resolution)

- Ο αριθμός των pixels που απαρτίζουν την ορατή εικόνα ονομάζεται ανάλυση (resolution)
- Κάθε οθόνη και κάρτα VGA θα πρέπει να υποστηρίζει ένα σύνολο πιθανών αναλύσεων.

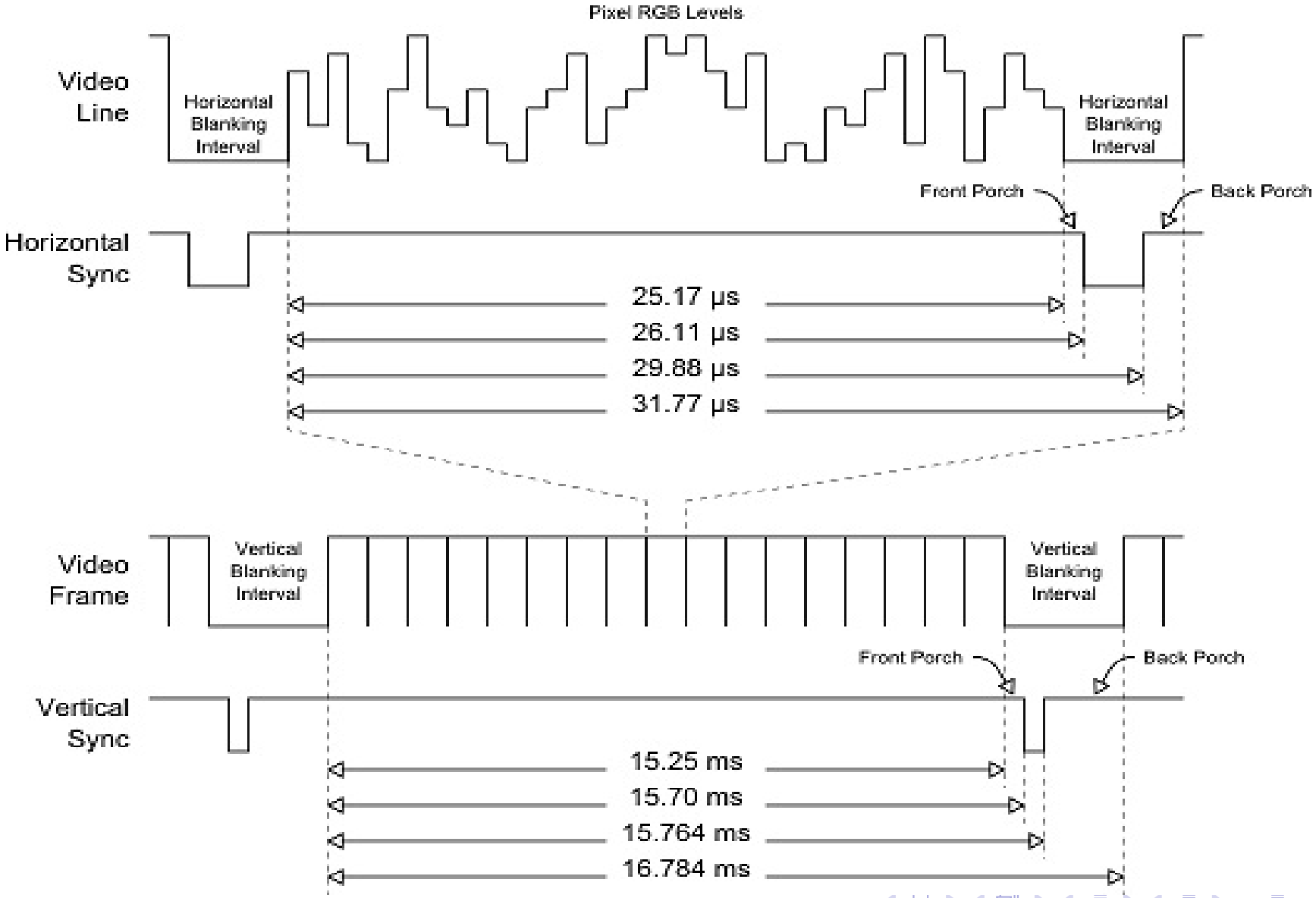




Οριζόντιος & Κάθετος συγχονισμός

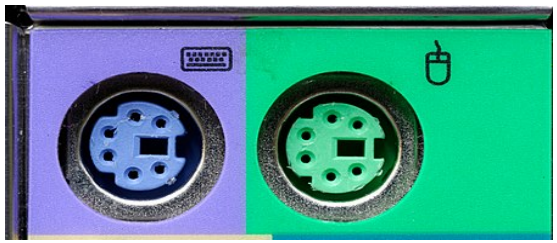
- Για τη σάρωση ενός καρέ απαιτούνται:
  - Ένα σήμα οριζόντιου συγχρονισμού (HSYNC) που υποδεικνύει το πέρας σάρωσης μιας γραμμής. Μετριέται σε περιόδους pixels.
  - Ένα σήμα κάθετου συγχρονισμού (VSYNC) που υποδεικνύει το πέρας σάρωσης ενός καρέ. Μετριέται σε περιόδους γραμμών απεικόνισης.
- Στη διάρκεια και την πολικότητα των παλμών αυτών των σημάτων μπορούμε να κωδικοποιήσουμε και την ανάλυση του καρέ που θα ακολουθήσει.

Παράδειγμα χρονισμού

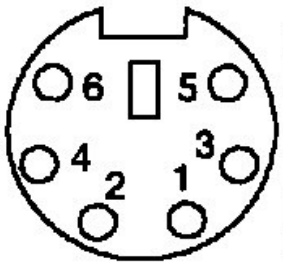


Γενικά

- Το πληκτρολόγιο είναι μια συσκευή εισόδου.
- Το ποντίκι είναι μια συσκευή εισόδου & εξόδου.
- Παρότι υπάρχουν διάφορα πρωτόκολλα για τη διασύνδεση αυτών των συσκευών, εμείς θα επικεντρώσουμε στο PS/2.
- Παρουσιάστηκε το 1987 από την IBM ως θύρα του Personal System/2 (δεύτερης γενιάς προσωπικοί υπολογιστές).
- Αντικατέστησε το σειριακό πρωτόκολλο (RS-232).
- Η σύνδεση γίνεται με βύσμα 6 ακροδεκτών, εκ των οποίων χρησιμοποιούνται μόνο οι 4:



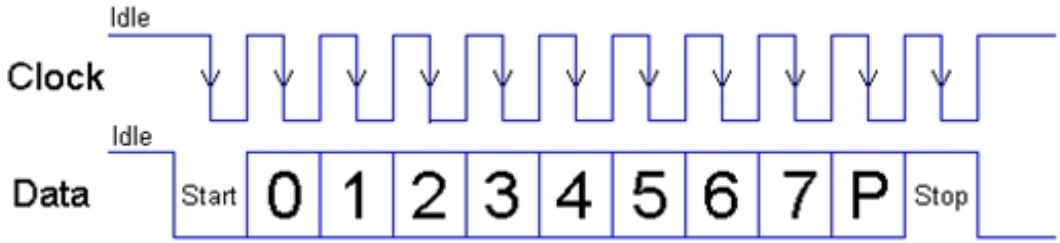
Σήματα & Πρωτόκολλο



Pinout on Female Port

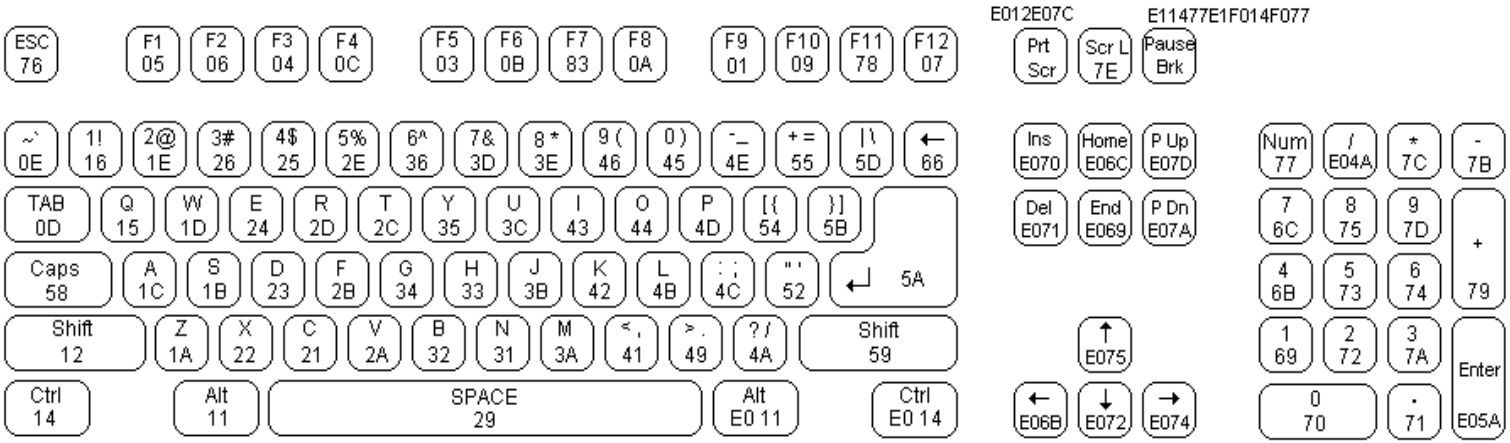
Pin Number	*Description*
1	Data
2	No Connection
3	Ground
4	+5V
5	Clock
6	No Connection

- Πρόκειται για ένα σειριακό, σύγχρονο, half duplex πρωτόκολλο.
- Τα σήματα που χρησιμοποιούνται για την υλοποίηση του πρωτοκόλλου είναι τα Data και Clock.
- Το ρολόι είναι της τάξης των 20-30 KHz
- Κάθε πακέτο μετάδοσης είναι των 11 δυαδικών ψηφίων (1 byte δεδομένων + 1 parity bit + start + stop bit)
- Κάθε bit μεταδίδεται στην ανοδική ακμή του ρολογιού.
- Όταν δε μεταδίδεται τίποτε οι γραμμές είναι idle.



Πληκτρολόγιο

- Το πληκτρολόγιο είναι μια συσκευή εισόδου.
- Κάθε πάτημα ενός πλήκτρου στο πληκτρολόγιο δημιουργεί έναν κωδικό (scan code).
- Ο κωδικός αντιστοιχεί στη στήλη και τη γραμμή του πλήκτρου σε μια QWERTY διάταξη.
- Ο κωδικός θα συνεχίσει να παράγεται και να μεταδίδεται όσο το πλήκτρο είναι πατημένο.
- Όταν αφεθεί το πλήκτρο, παράγεται το <F0>+<scan code>.



Ποντίκι

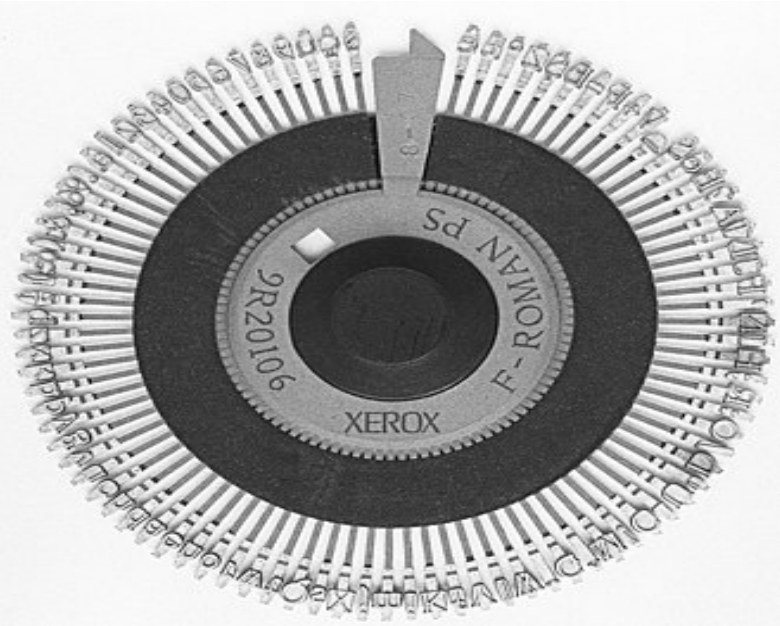
- Το ποντίκι είναι μια συσκευή εισόδου & εξόδου.
- Η επικοινωνία ξεκινά από τον υπολογιστή (host / master) που στέλνει κάποια εντολή {Reset, Self test, Resend last byte, Start, Stop transmitting movement packets}.
- Το ποντίκι (slave) αποκρίνεται με {Acknowledge, Self test passed or failed, Device ID}
- Το ποντίκι έχει εσωτερικά δύο μετρητές που καταγράφουν κίνηση ως προς τους δύο άξονες.
- Μεταδίδει 3 πακέτα του 1 byte που καταγράφουν τη διαφορά των μετρητών ως προς την προηγούμενη θέση.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Y overflow	X overflow	Y sign	X sign	Always 1	Middle button	Right button	Left button
Byte 1	X movement							
Byte 2	Y movement							

Μονάδες εκτύπωσης

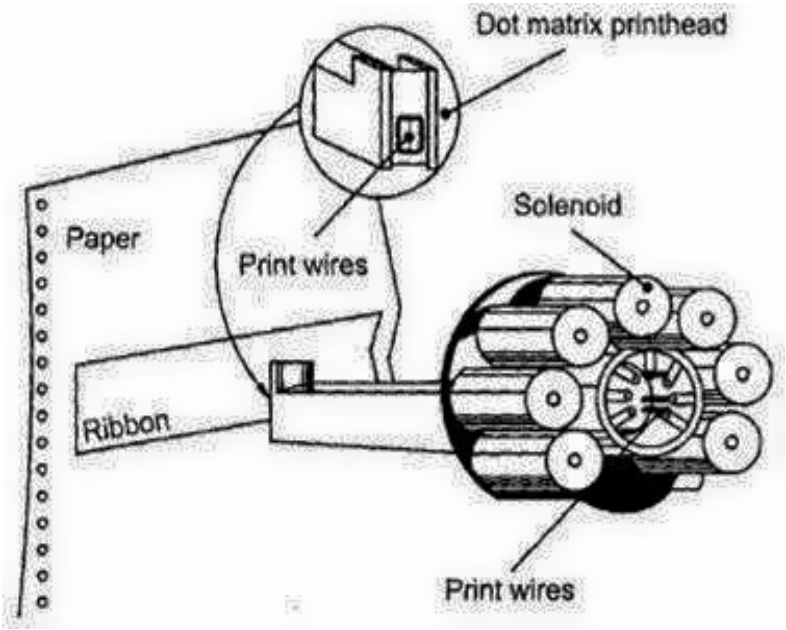
- Για την εκτύπωση, χρησιμοποιήθηκαν / χρησιμοποιούνται εκτυπωτές:
  - Αλυσίδας / Μαργαρίτας (Daisy chain)
  - Ακίδων (Dot-matrix)
  - Εκτόξευσης μελάνης (Inkjet)
  - Laser

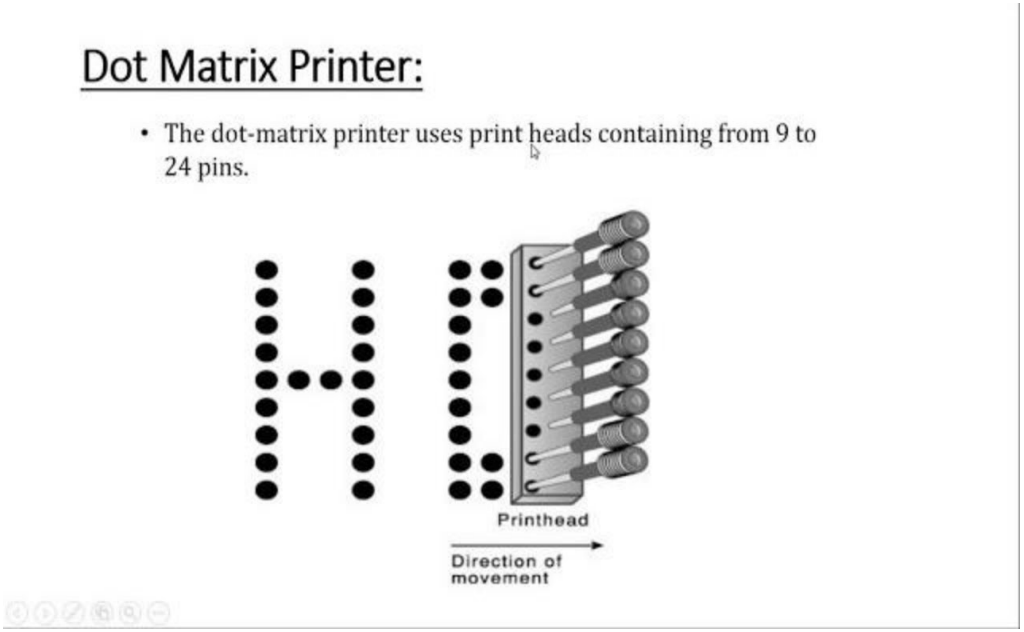
Daisy-Chain Printers



- Ανάγλυφοι χαρακτήρες τοποθετούνται πάνω στις ακτίνες ενός δίσκου ή κατά μήκος μιας ταινίας.
- Με περιστροφή επιλέγεται ένας ο οποίος χτυπώντας πάνω στη μελανοταινία αποτυπώνεται στο χαρτί.
-
  - \+ : Εκτύπωση σε πολλαπλά αντίγραφα με καρμπόν.
  - \- : Προαποφασισμένο σύνολο χαρακτήρων, αδυναμία εκτύπωσης εικόνων / γραφικών, θόρυβος, χαμηλή ταχύτητα.

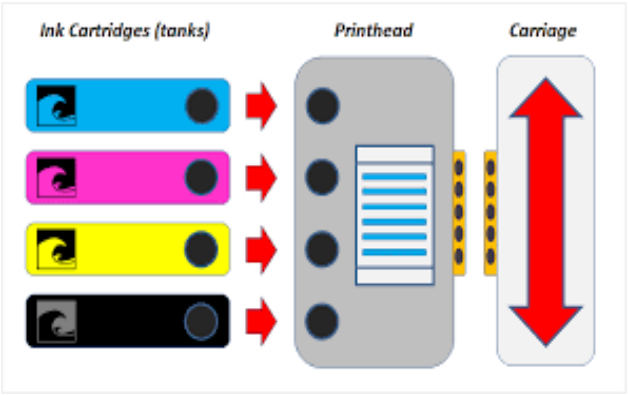
Dot-matrix εκτυπωτές

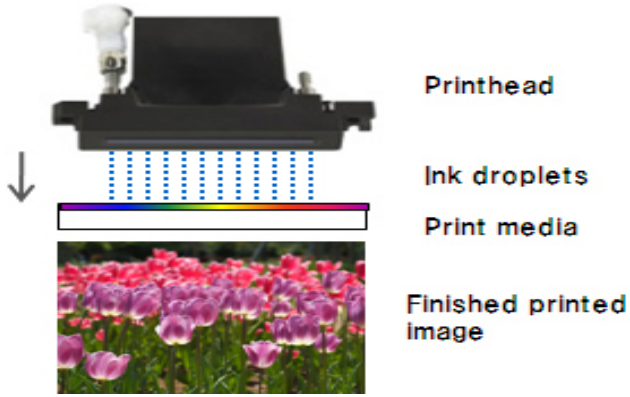




- Οι ανάγλυφοι χαρακτήρες αντικαταστάθηκαν με μια κεφαλή με μικροσκοπικά σφυριά - ακίδες.
- Κάποια εξ αυτών ενεργοποιούμενα, χτυπούν πάνω στη μελανοταινία και αποτυπώνονται αντίστοιχα στίγματα στο χαρτί.
- Ο αριθμός των ακίδων ποικίλει από 9 έως 24.
-
  - + : Εκτύπωση σε πολλαπλά αντίγραφα με καρμπόν.
  - - : Θόρυβος, χαμηλή ταχύτητα.

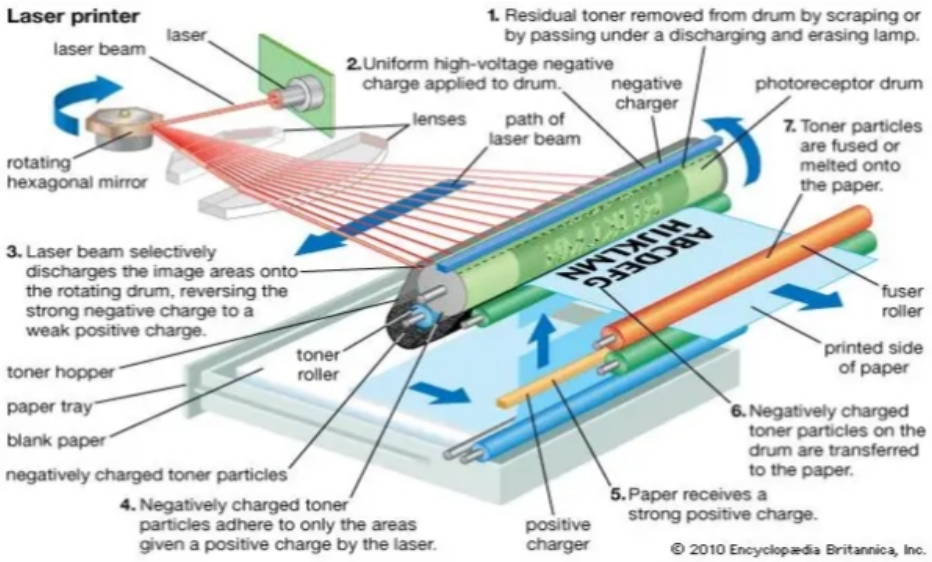
Inkjet εκτυπωτές

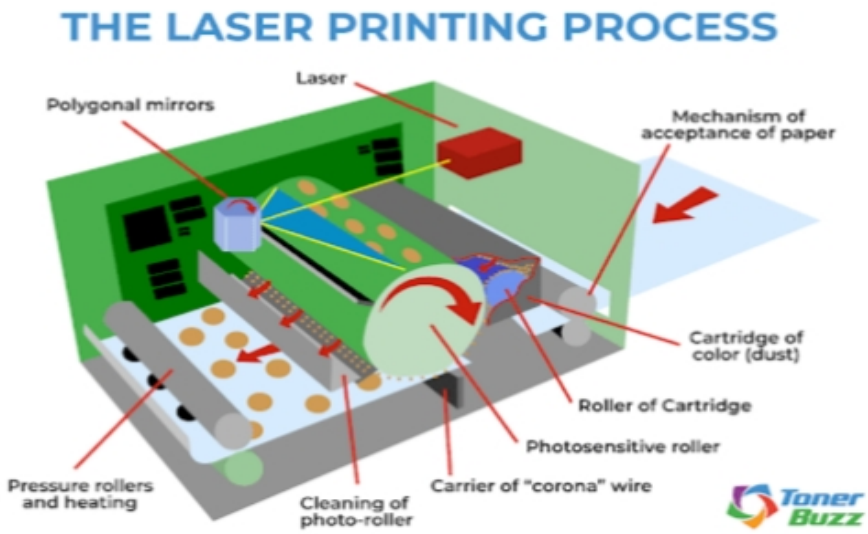




- Αντί για πρόσκρουση σε μελανοταινία, οι ακίδες αντικαταστάθηκαν με μικροσκοπικούς εκτοξευτές μελάνης.
- Με την πρόσμιξη διαφόρων χρωμάτων μελανιών αποκτούμε την ικανότητα έγχρωμης εκτύπωσης.
- Ο αριθμός των εκτοξευτών ποικίλει από 15 έως 80+.
-
  - \+ : Έγχρωμη εκτύπωση.
  - \- : Ανικανότητα παράλληλης εκτύπωσης σε πολλαπλά αντίγραφα, χαμηλή ταχύτητα, υψηλό κόστος μελανιών.

Laser εκτυπωτές





- Μια ακτίνα κι ένα σύστημα καθρεπτών αποφορτίζει σημεία του αρχικά αρνητικά φορτισμένου τυμπάνου (drum).
- Τα μη αποφορτισμένα σημεία απορροφούν από το toner γραφίτη και ακολούθως εναποτίθενται πάνω στο χαρτί.
- Με θέρμανση του χαρτιού, η αποτύπωση γίνεται μόνιμη.
-
  - + : Ταχύτητα, χαμηλό κόστος εκτύπωσης.
  - - : Υψηλό αρχικό κόστος ειδικά για έγχρωμη εκτύπωση.

Network - Node - Link

- Ένα δίκτυο (network) είναι ένα σύνολο από συσκευές - κόμβους (nodes) που διασυνδέονται με συνδέσμους επικοινωνίας (links).

Network - Node - Link

- Ένα δίκτυο (network) είναι ένα σύνολο από συσκευές - κόμβους (nodes) που διασυνδέονται με συνδέσμους επικοινωνίας (links).
- Οι κόμβοι ενός δικτύου μπορεί να είναι:
  - Υπολογιστές

Network - Node - Link

- Ένα δίκτυο (network) είναι ένα σύνολο από συσκευές - κόμβους (nodes) που διασυνδέονται με συνδέσμους επικοινωνίας (links).
- Οι κόμβοι ενός δικτύου μπορεί να είναι:
  - Υπολογιστές
  - Εκτυπωτές

Network - Node - Link

- Ένα δίκτυο (network) είναι ένα σύνολο από συσκευές - κόμβους (nodes) που διασυνδέονται με συνδέσμους επικοινωνίας (links).
- Οι κόμβοι ενός δικτύου μπορεί να είναι:
  - Υπολογιστές
  - Εκτυπωτές
  - Συσκευές απεικόνισης

Network - Node - Link

- Ένα δίκτυο (network) είναι ένα σύνολο από συσκευές - κόμβους (nodes) που διασυνδέονται με συνδέσμους επικοινωνίας (links).
- Οι κόμβοι ενός δικτύου μπορεί να είναι:
  - Υπολογιστές
  - Εκτυπωτές
  - Συσκευές απεικόνισης
  - Τηλέφωνα

Network - Node - Link

- Ένα δίκτυο (network) είναι ένα σύνολο από συσκευές - κόμβους (nodes) που διασυνδέονται με συνδέσμους επικοινωνίας (links).
- Οι κόμβοι ενός δικτύου μπορεί να είναι:
  - Υπολογιστές
  - Εκτυπωτές
  - Συσκευές απεικόνισης
  - Τηλέφωνα
  - Προσαρμοστές υποδικτύων (subnetworks)

Network - Node - Link

- Ένα δίκτυο (network) είναι ένα σύνολο από συσκευές - κόμβους (nodes) που διασυνδέονται με συνδέσμους επικοινωνίας (links).
- Οι κόμβοι ενός δικτύου μπορεί να είναι:
  - Υπολογιστές
  - Εκτυπωτές
  - Συσκευές απεικόνισης
  - Τηλέφωνα
  - Προσαρμοστές υποδικτύων (subnetworks)
  - Οποιαδήποτε άλλη συσκευή ικανή να στείλει ή / και να δεχθεί δεδομένα που γεννούν οι υπόλοιποι κόμβοι.

Network - Node - Link

- Ένα δίκτυο (network) είναι ένα σύνολο από συσκευές - κόμβους (nodes) που διασυνδέονται με συνδέσμους επικοινωνίας (links).
- Οι κόμβοι ενός δικτύου μπορεί να είναι:
  - Υπολογιστές
  - Εκτυπωτές
  - Συσκευές απεικόνισης
  - Τηλέφωνα
  - Προσαρμοστές υποδικτύων (subnetworks)
  - Οποιαδήποτε άλλη συσκευή ικανή να στείλει ή / και να δεχθεί δεδομένα που γεννούν οι υπόλοιποι κόμβοι.
- Οι σύνδεσμοι επικοινωνίας μπορεί να είναι:
  - Καλώδια

Network - Node - Link

- Ένα δίκτυο (network) είναι ένα σύνολο από συσκευές - κόμβους (nodes) που διασυνδέονται με συνδέσμους επικοινωνίας (links).
- Οι κόμβοι ενός δικτύου μπορεί να είναι:
  - Υπολογιστές
  - Εκτυπωτές
  - Συσκευές απεικόνισης
  - Τηλέφωνα
  - Προσαρμοστές υποδικτύων (subnetworks)
  - Οποιαδήποτε άλλη συσκευή ικανή να στείλει ή / και να δεχθεί δεδομένα που γεννούν οι υπόλοιποι κόμβοι.
- Οι σύνδεσμοι επικοινωνίας μπορεί να είναι:
  - Καλώδια
  - Οπτικές ίνες

Network - Node - Link

- Ένα δίκτυο (network) είναι ένα σύνολο από συσκευές - κόμβους (nodes) που διασυνδέονται με συνδέσμους επικοινωνίας (links).
- Οι κόμβοι ενός δικτύου μπορεί να είναι:
  - Υπολογιστές
  - Εκτυπωτές
  - Συσκευές απεικόνισης
  - Τηλέφωνα
  - Προσαρμοστές υποδικτύων (subnetworks)
  - Οποιαδήποτε άλλη συσκευή ικανή να στείλει ή / και να δεχθεί δεδομένα που γεννούν οι υπόλοιποι κόμβοι.
- Οι σύνδεσμοι επικοινωνίας μπορεί να είναι:
  - Καλώδια
  - Οπτικές ίνες
  - Ραδιοσήματα / ηλεκτρομαγνητικά σήματα

Σκοπός

- Σκοπός της δημιουργίας ενός δικτύου είναι η διαμοίραση των πόρων (resources)

Σκοπός

- Σκοπός της δημιουργίας ενός δικτύου είναι η διαμοίραση των πόρων (resources)
- Πιθανοί πόροι μπορεί να είναι:
  - Αρχεία

Σκοπός

- Σκοπός της δημιουργίας ενός δικτύου είναι η διαμοίραση των πόρων (resources)
- Πιθανοί πόροι μπορεί να είναι:
  - Αρχεία
  - Ολόκληρες συλλογές αρχείων - ντοσιέ (folders)

Σκοπός

- Σκοπός της δημιουργίας ενός δικτύου είναι η διαμοίραση των πόρων (resources)
- Πιθανοί πόροι μπορεί να είναι:
  - Αρχεία
  - Ολόκληρες συλλογές αρχείων - ντοσιέ (folders)
  - Εκτυπωτές

Σκοπός

- Σκοπός της δημιουργίας ενός δικτύου είναι η διαμοίραση των πόρων (resources)
- Πιθανοί πόροι μπορεί να είναι:
  - Αρχεία
  - Ολόκληρες συλλογές αρχείων - ντοσιέ (folders)
  - Εκτυπωτές
  - Δίσκοι

Σκοπός

- Σκοπός της δημιουργίας ενός δικτύου είναι η διαμοίραση των πόρων (resources)
- Πιθανοί πόροι μπορεί να είναι:
  - Αρχεία
  - Ολόκληρες συλλογές αρχείων - ντοσιέ (folders)
  - Εκτυπωτές
  - Δίσκοι
  - Οποιαδήποτε συσκευή συνδέεται σε ένα υπολογιστικό σύστημα

Πλεονεκτήματα

- Τα πιθανά πλεονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:

Πλεονεκτήματα

- Τα πιθανά πλεονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Συνδεσιμότητα και επικοινωνία

Πλεονεκτήματα

- Τα πιθανά πλεονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Συνδεσιμότητα και επικοινωνία
  - Διαμοίραση δεδομένων

Πλεονεκτήματα

- Τα πιθανά πλεονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Συνδεσιμότητα και επικοινωνία
  - Διαμοίραση δεδομένων
  - Διαμοίραση συσκευών υλικού

Πλεονεκτήματα

- Τα πιθανά πλεονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Συνδεσιμότητα και επικοινωνία
  - Διαμοίραση δεδομένων
  - Διαμοίραση συσκευών υλικού
  - (Διαμοιραζόμενη) πρόσβαση στο διαδίκτυο

Πλεονεκτήματα

- Τα πιθανά πλεονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Συνδεσιμότητα και επικοινωνία
  - Διαμοίραση δεδομένων
  - Διαμοίραση συσκευών υλικού
  - (Διαμοιραζόμενη) πρόσβαση στο διαδίκτυο
  - Ασφάλεια δεδομένων και αποτελεσματική διαχείρισή τους

Πλεονεκτήματα

- Τα πιθανά πλεονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Συνδεσιμότητα και επικοινωνία
  - Διαμοίραση δεδομένων
  - Διαμοίραση συσκευών υλικού
  - (Διαμοιραζόμενη) πρόσβαση στο διαδίκτυο
  - Ασφάλεια δεδομένων και αποτελεσματική διαχείρισή τους
  - Αυξημένη υπολογιστική ισχύς (παράλληλη επεξεργασία) και εξομάλυνση του φόρτου

Πλεονεκτήματα

- Τα πιθανά πλεονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Συνδεσιμότητα και επικοινωνία
  - Διαμοίραση δεδομένων
  - Διαμοίραση συσκευών υλικού
  - (Διαμοιραζόμενη) πρόσβαση στο διαδίκτυο
  - Ασφάλεια δεδομένων και αποτελεσματική διαχείρισή τους
  - Αυξημένη υπολογιστική ισχύς (παράλληλη επεξεργασία) και εξομάλυνση του φόρτου
  - Διασκέδαση

Μειονεκτήματα

- Τα πιθανά μειονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:

Μειονεκτήματα

- Τα πιθανά μειονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Κόστος συσκευών υλικού

Μειονεκτήματα

- Τα πιθανά μειονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Κόστος συσκευών υλικού
  - Κόστος λογισμικού

Μειονεκτήματα

- Τα πιθανά μειονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Κόστος συσκευών υλικού
  - Κόστος λογισμικού
  - Κόστος διαχείρισης υλικού και λογισμικού

Μειονεκτήματα

- Τα πιθανά μειονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Κόστος συσκευών υλικού
  - Κόστος λογισμικού
  - Κόστος διαχείρισης υλικού και λογισμικού
  - Μη επιθυμητή διαμοίραση πόρων

Μειονεκτήματα

- Τα πιθανά μειονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Κόστος συσκευών υλικού
  - Κόστος λογισμικού
  - Κόστος διαχείρισης υλικού και λογισμικού
  - Μη επιθυμητή διαμοίραση πόρων
  - Παράνομη ή μη επιθυμητή συμπεριφορά μεταξύ των μελών του δικτύου

Μειονεκτήματα

- Τα πιθανά μειονεκτήματα της κατασκευής και χρήσης ενός δικτύου είναι:
  - Κόστος συσκευών υλικού
  - Κόστος λογισμικού
  - Κόστος διαχείρισης υλικού και λογισμικού
  - Μη επιθυμητή διαμοίραση πόρων
  - Παράνομη ή μη επιθυμητή συμπεριφορά μεταξύ των μελών του δικτύου
  - Προβλήματα ασφάλειας διαμοιραζόμενων δεδομένων

LAN – MAN – WAN

- Ανάλογα με τη γεωγραφική περιοχή που καλύπτουν, τα δίκτυα συνήθως ταξινομούνται σε 3 κατηγορίες:

LAN – MAN – WAN

- Ανάλογα με τη γεωγραφική περιοχή που καλύπτουν, τα δίκτυα συνήθως ταξινομούνται σε 3 κατηγορίες:
  - Τοπικά δίκτυα (Local Area Networks - LANs)

LAN – MAN – WAN

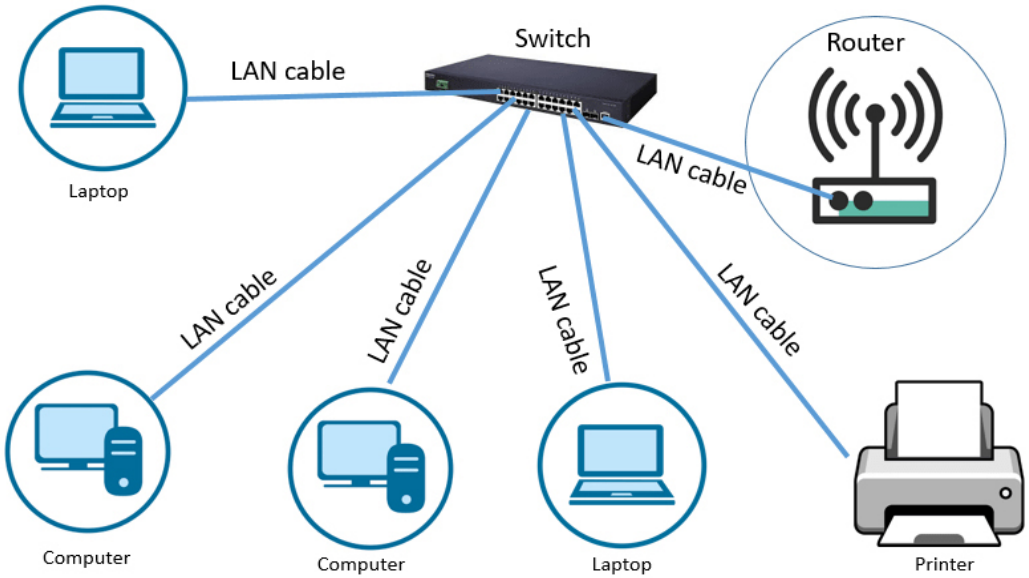
- Ανάλογα με τη γεωγραφική περιοχή που καλύπτουν, τα δίκτυα συνήθως ταξινομούνται σε 3 κατηγορίες:
  - Τοπικά δίκτυα (Local Area Networks - LANs)
  - Μητροπολιτικά δίκτυα (Metropolitan Area Networks - MANs), και

LAN – MAN – WAN

- Ανάλογα με τη γεωγραφική περιοχή που καλύπτουν, τα δίκτυα συνήθως ταξινομούνται σε 3 κατηγορίες:
  - Τοπικά δίκτυα (Local Area Networks - LANs)
  - Μητροπολιτικά δίκτυα (Metropolitan Area Networks - MANs), και
  - Δίκτυα Ευρείας Περιοχής (Wide Area Networks - WANs).

LAN

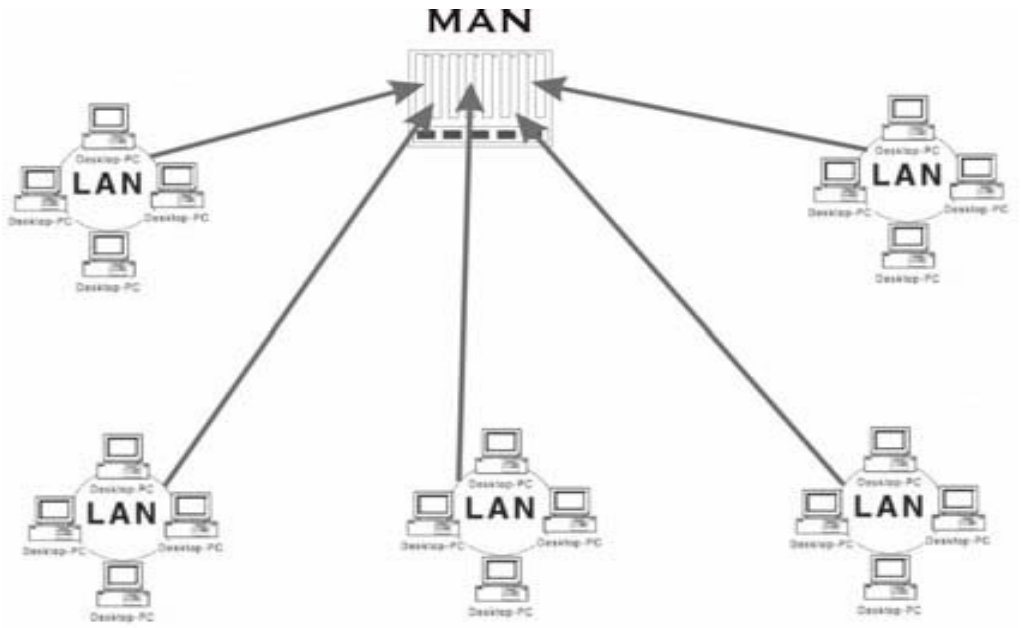
- Ένα τοπικό δίκτυο (LAN) καλύπτει μια μικρή γεωγραφική περιοχή, όπως ένα σπίτι, ένα γραφείο ένα κτήριο ή μια ομάδα γραφείων ή κτηρίων



Local Area Network

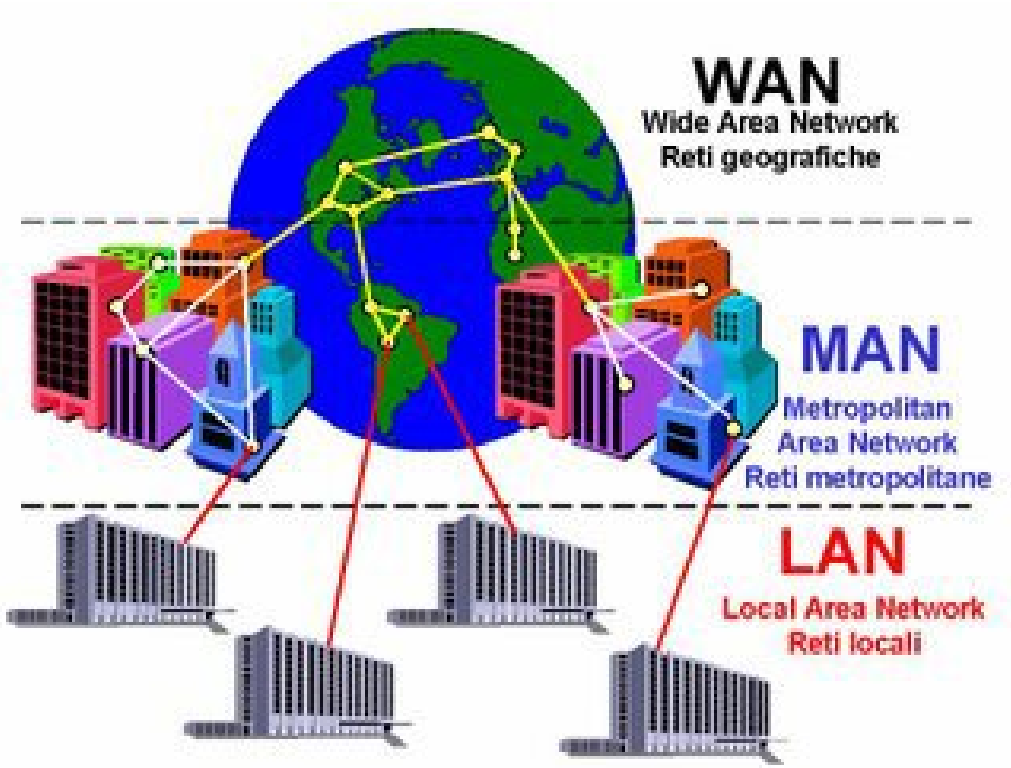
MAN

- Ένα μητροπολιτικό δίκτυο (MAN) συνδέει συσκευές και χρήστες μιας γεωγραφικής περιοχής μεγαλύτερης ακόμα και από αυτήν ενός μεγάλου LAN αλλά μικρότερης από ένα WAN. Συνήθως ο όρος (MAN) χρησιμοποιείται για τη διασύνδεση των δικτύων μιας πόλης σε ένα μεγαλύτερο δίκτυο, ή πολλών LANs σε ένα δίκτυο υποστήριξης (π.χ. δίκτυο πανεπιστημιούπολης).



WAN

- Ένα δίκτυο που καλύπτει μια πολύ μεγάλη περιοχή (π.χ. ξεπερνάει τα όρια μιας πόλης, ενός νομού ή μιας χώρας είναι ένα δίκτυο ευρείας περιοχής (WAN).
- Το μεγαλύτερο και πιο γνωστό WAN είναι το διαδίκτυο (Internet).



Συνοπτικές Συγκρίσεις

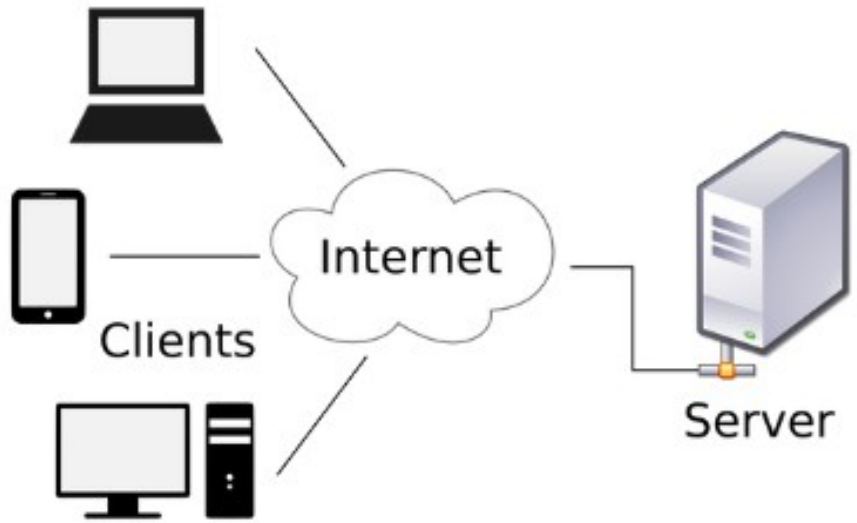
Comparison	LAN	MAN	WAN
Full Name	Local Area Network	Metropolitan Area Network	Wide Area Network
Meaning	A network that connects a group of computers in a small geographical area	It covers relatively large region such as cities, towns	It spans large locality & connects countries together. e.g. Internet
Ownership of Network	Private	Private or Public	Private or Public (VPN)
Design and Maintenance	Easy	Difficult	Difficult
Propagation Delay	Short	Moderate	Long
Speed	High	Moderate	Low
Equipment Used	NIC, Switch, Hub	Modem, Router	Microwave, Radio Transmitter & Receiver
Range(Approximately)	1 to 10 km	10 to 100 km	Beyond 100 km
Used for	College, School, Hospital	Small towns, City	State, Country, Continent

Peer-to-peer & Client-Server Δίκτυα

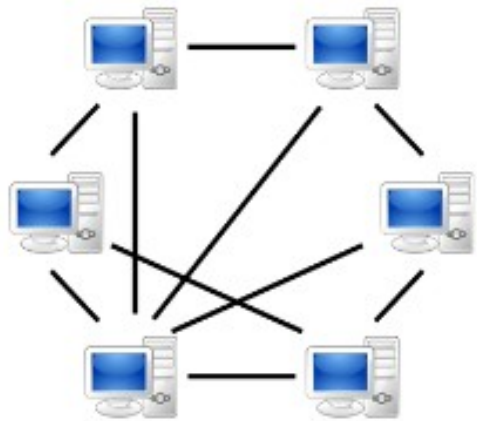
Ανάλογα με το ρόλο των κόμβων, τα δίκτυα ταξινομούνται σε δίκτυα ομοτίμων (peer-to-peer - p2p) και σε δίκτυα πελατών - εξυπηρετητών (client - server)

- p2p δίκτυα
  - Οι υπολογιστές είναι παράλληλα σταθμοί εργασίας αλλά και εξυπηρετητές.
  - Κάθε κόμβος έχει πόρους που διαμοιράζονται με οποιοδήποτε άλλο κόμβο.
  - Εξαιρετική επιλογή για μικρά, απλά και χαμηλού κόστους δίκτυα.
- Client-server δίκτυα
  - Ένας μικρός αριθμός από κόμβους χαρακτηρίζονται ως κεντρικοί εξυπηρετητές (π.χ. αρχείων, εκτυπώσεων, σύνδεσης με το διαδίκτυο).
  - Παρέχουν τις υπηρεσίες τους σε ένα μεγάλο αριθμό από πολύ χαμηλότερου κόστους κόμβους - πελάτες.

Peer-to-peer & Client-Server Δίκτυα



Client-Server

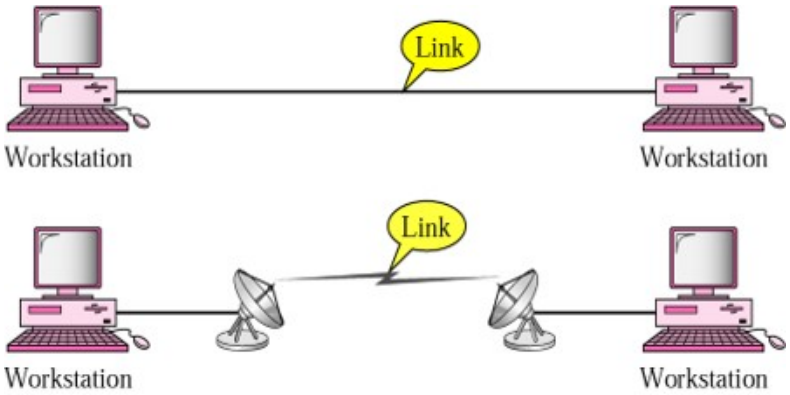


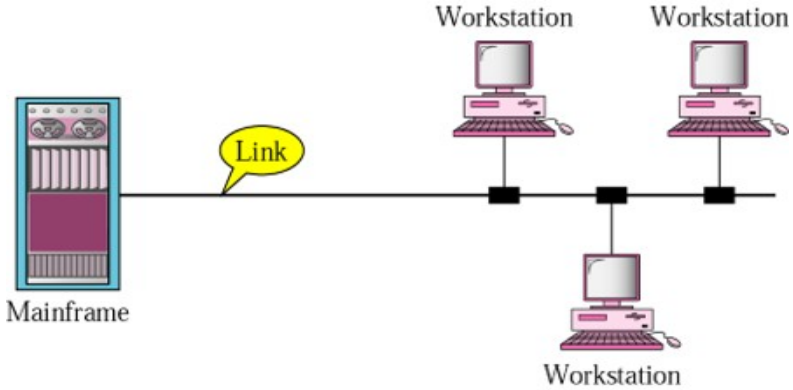
Peer-to-Peer

Point to point vs Multipoint

Ανάλογα με τον αριθμό των συνδεόμενων κόμβων σε κάθε σύνδεση, τα δίκτυα ταξινομούνται σε:

- Point to Point, στα οποία διατίθεται μια αποκλειστική σύνδεση μεταξύ κάθε δύο κόμβων, και
- Multipoint, στην οποία περισσότεροι τών δύο κόμβων μοιράζονται την ίδια σύνδεση. Προφανώς απαιτεί σχήμα διαιτησίας (token / CSMA).

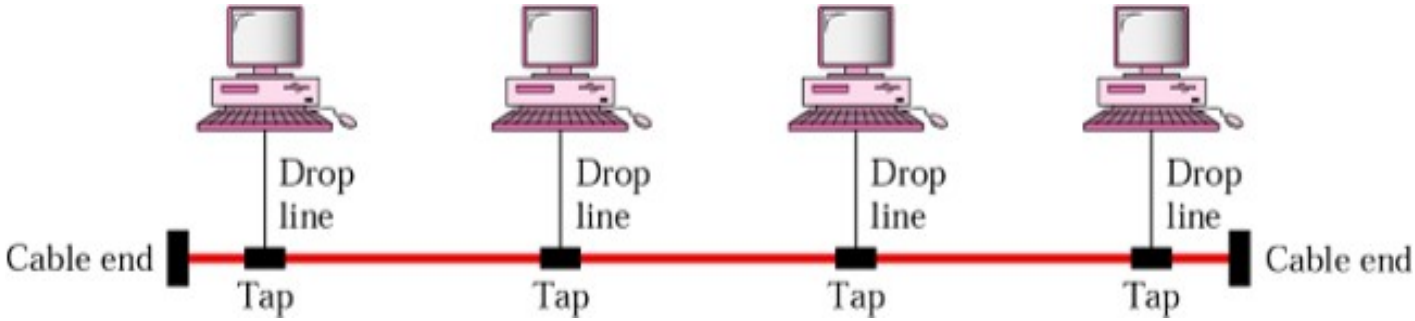




Τοπολογίες

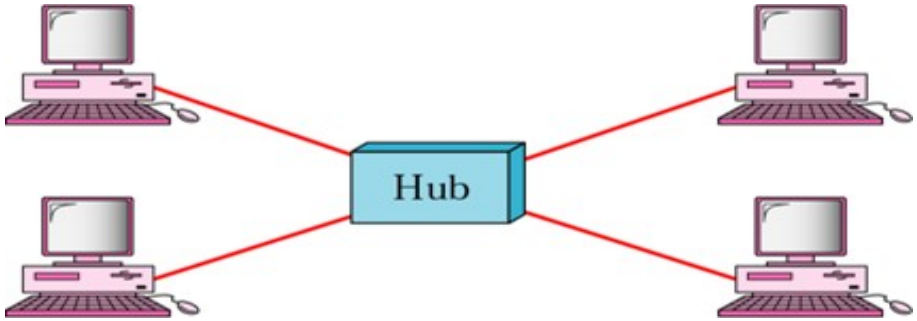
- Με τον όρο τοπολογία ενός δικτύου εννοούμε τον τρόπο με τον οποίο έχει δομηθεί. Συνήθως αυτή αποτυπώνεται σε ένα διάγραμμα κόμβων και συνδέσεων.
- Ωστόσο, θα πρέπει να είμαστε προσεκτικοί, καθώς υπάρχουν δύο διαφορετικές τοπολογίες:
  - Η φυσική τοπολογία η οποία περιγράφει τον τρόπο με τον οποίο έχουν δομηθεί τα φυσικά μέσα (καλώδια, ίνες, σταθμοί αναμετάδοσης, ...), και
  - Η λογική τοπολογία η οποία περιγράφει το δρόμο με τον οποίο δρομολογούνται τα μηνύματα μέσα στο δίκτυο.
- Υπάρχουν οι εξής τοπολογίες :
  - Αρτηρίας (Bus)
  - Αστέρα (Star) Ορίζεται μόνο ως φυσική τοπολογία.
  - Δακτυλίου (Ring), και
  - Πλέγματος (Mesh)

Τοπολογία Αρτηρίας



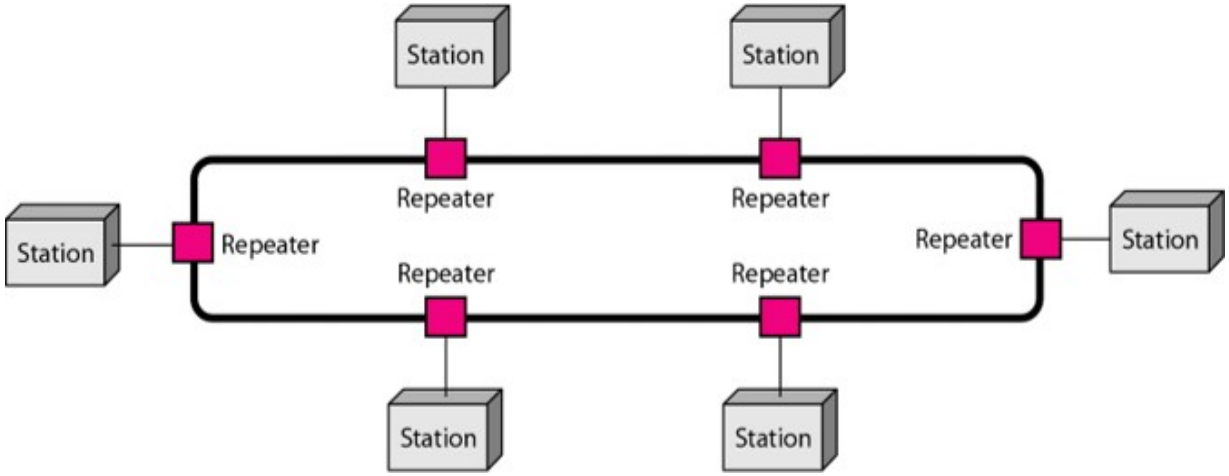
- Multipoint. Όλοι οι κόμβοι συνδέονται σε μια γραμμή backbone.
- Για τη σύνδεσή τους οι κόμβοι χρησιμοποιούν γραμμές διακλάδωσης (drop lines) και διακλαδώσεις (taps).
- +: Εύκολη εγκατάσταση
- -: Δύσκολη απομόνωση σφάλματος και επανασύνδεση. Πρόβλημα στη κεντρική γραμμή βγάζει το δίκτυο εκτός λειτουργίας.

Τοπολογία Αστέρα



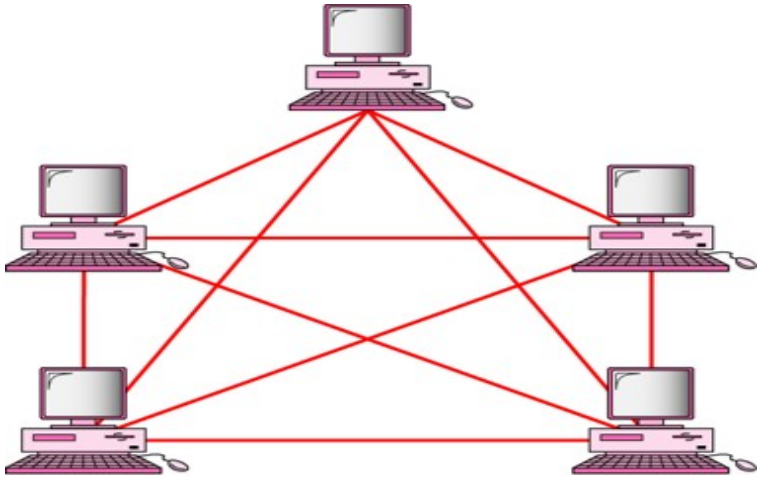
- Κάθε κόμβος επικοινωνεί με μια point-to-point σύνδεση με έναν κεντρικό ελεγκτή, που ονομάζεται hub. Δεν υπάρχει άμεση επικοινωνία μεταξύ κόμβων.
- +: Χαμηλό κόστος και ευκολία εγκατάστασης και αναδιάρθρωσης. Πολύ εύρωστο.
- -: Αξιοπιστία ίση με αυτήν του hub (Single point of failure).

Τοπολογία Δακτυλίου



- Κάθε κόμβος έχει δύο point-to-point συνδέσεις μόνο με τους κομβους που βρίσκονται αριστερά και δεξιά του. Κάθε μήνυμα διαδίδεται προς τη μία κατεύθυνση από κόμβο σε κόμβο μέχρι να φθάσει στον κόμβο προορισμού του. Κάθε κόμβος περιέχει έναν αναμεταδότη.
- +: Εύκολη εγκατάσταση και αναδιάρθρωση. Εύκολη απομόνωση βλάβης.
- -: Χαμηλή ταχύτητα λόγω μηνυμάτων συγχρονισμού και κυριότητας. Μονοκατευθυντήρια μετάδοση.

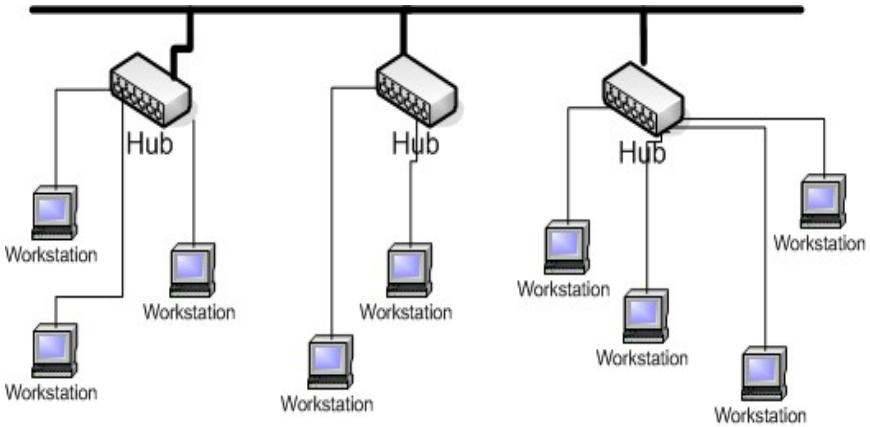
Τοπολογία Πλέγματος

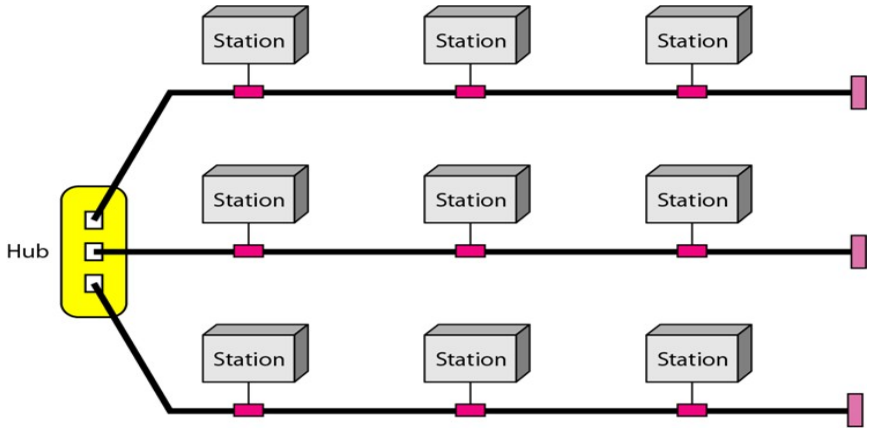


- Κάθε κόμβος έχει ξεχωριστή point-to-point σύνδεση με κάθε άλλο κόμβο πάνω στην οποία διαδίδονται τα μηνύματα μόνο των δύο κόμβων που διασυνδέει.
- Ένα πλήρες πλέγμα διαθέτει $\frac{n(n-1)}{2}$ φυσικές συνδέσεις, ενώ κάθε κόμβος πρέπει να έχει $(n-1)$ θύρες επικοινωνίας.
- +: Γρήγορο, ασφαλές, εύκολη συντήρηση.
- -: Ακριβό, απαιτεί πολλούς πόρους (θύρες, μέσα διασύνδεσης).

Υβριδικές Τοπολογίες

- Στις μέρες μας όλα τα δίκτυα είναι μια διασύνδεση μικρότερων δικτύων, καθένα με μια δική του τοπολογία. Έτσι προκύπτουν υβριδικές τοπολογίες.





Πρωτόκολλα - Standards - Οργανισμοί

- Θα ήταν παντελώς ανόητο κάθε εταιρεία να αναπτύσσει δίκτυα αποκλειστικά για δική της χρήση.
- Η διασύνδεση συνεπώς πέραν από το φυσικό μέσο, περιλαμβάνει κάποια πρωτόκολλα τα οποία υπακούν σε κάποια standards.
- Υπενθυμίζεται ότι:
  - Πρωτόκολλο: Ένα σύνολο κανόνων.
  - Standard: Κανόνες στους οποίους συμφωνούν όλοι.
- Τα standards εκδίδονται από επιστημονικές επιτροπές οργανισμών μετά από μελέτη προτάσεων και υπαρχόντων πρωτοκόλλων.
- Γνωστοί οργανισμοί: ISO (International Standards Organization), ITU-T (International Telecommunication Unit), ANSI (American National Standards Institute), IEEE (Institute of Electrical and Electronics Engineers), EIA (Electronic Industries Association).
- Στα δίκτυα ακολουθείται το standard OSI model (Open Systems Interconnect) του ISO.

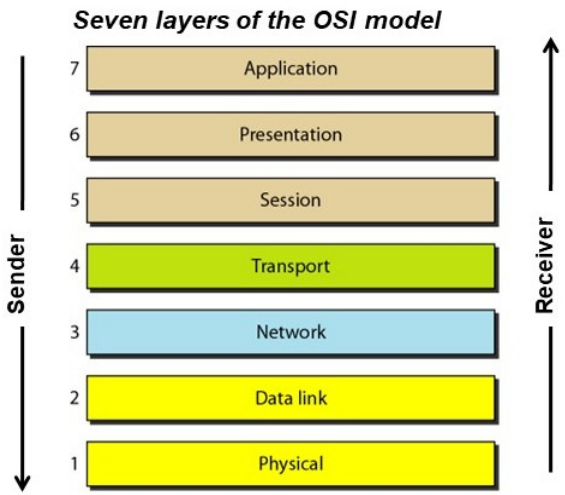
OSI: Ένα πολυεπίπεδο μοντέλο

- Το μοντέλο αποτελείται από 7 επίπεδα.
- Κάθε επίπεδο υλοποιεί ένα υποσύνολο των απαιτούμενων συναρτήσεων επικοινωνίας βασιζόμενο στις πιο πρωταρχικές συναρτήσεις που του παρέχει το πιο κάτω.
- Κάθε επίπεδο παρέχει έτοιμες συναρτήσεις εξυπηρέτησης (services) στο πιο πάνω επίπεδο.
- Αλλαγές σε κάποιο επίπεδο δεν απαιτούν αλλαγές στα υπόλοιπα.
- Δύο κόμβοι στην πραγματικότητα επικοινωνούν μόνο στο κάτω - κάτω επίπεδο, ενώ δίνουν την ψευδαίσθηση ότι επικοινωνούν στο πιο πάνω, το πιο κοντινό στον άνθρωπο.

Τα επίπεδα του OSI

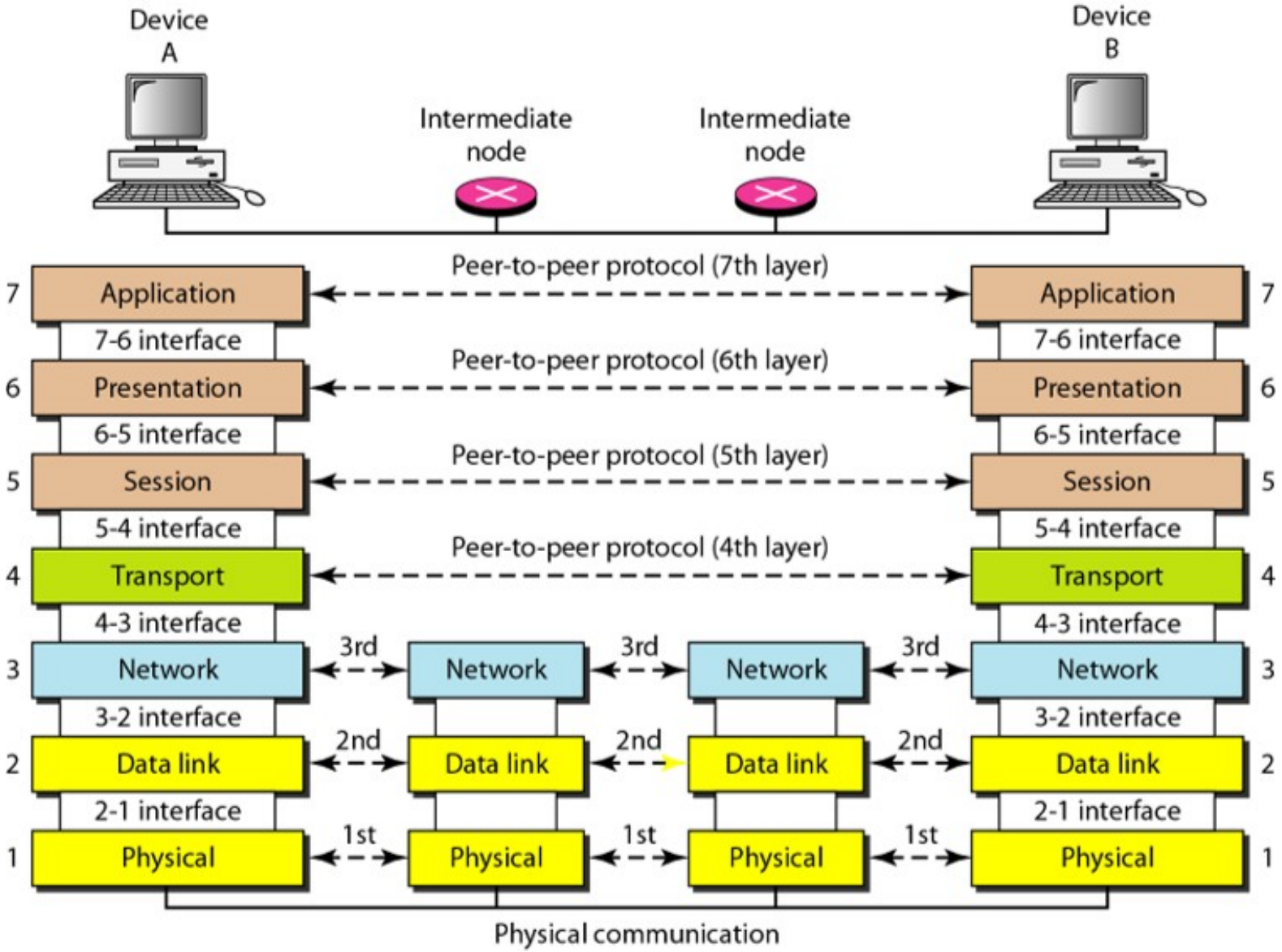
Layers

Layer 7. Application
Layer 6. Presentation
Layer 5. Session
Layer 4. Transport
Layer 3. Network
Layer 2. Data Link
Layer 1. Physical



- Η επικοινωνία κατεβαίνει τα επίπεδα στην πλευρά του αποστολέα, κι αφού περάσει από το κανάλι επικοινωνίας ανεβαίνει τα επίπεδα στον αποστολέα.
- Κάθε επίπεδο του αποστολέα προσθέτει τη δική του πληροφορία ελέγχου στο μήνυμα.
- Κάθε επίπεδο του παραλήπτη αφαιρεί από το μήνυμα την πληροφορία που προορίζεται γι αυτό και παραδίδει το νέο πακέτο στο επόμενο προς τα πάνω επίπεδο.

Ιδεατή και πραγματική επικοινωνία



Ένα πακέτο επικοινωνίας στο OSI