# ΠΑΡΑΛΛΗΛΗ ΕΠΕΞΕΡΓΑΣΙΑ

**ΜΝΗΜΗ**

**Πρωτόκολλα Συνέπειας Μνήμης σε**

**Πολυεπεξεργαστικά Υπολογιστικά Συστήματα**

**ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ & ΠΛΗΡΟΦΟΡΙΚΗΣ**

**ΕΡΓΑΣΤΗΡΙΟ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΥΨΗΛΩΝ ΕΠΙΔΟΣΕΩΝ**

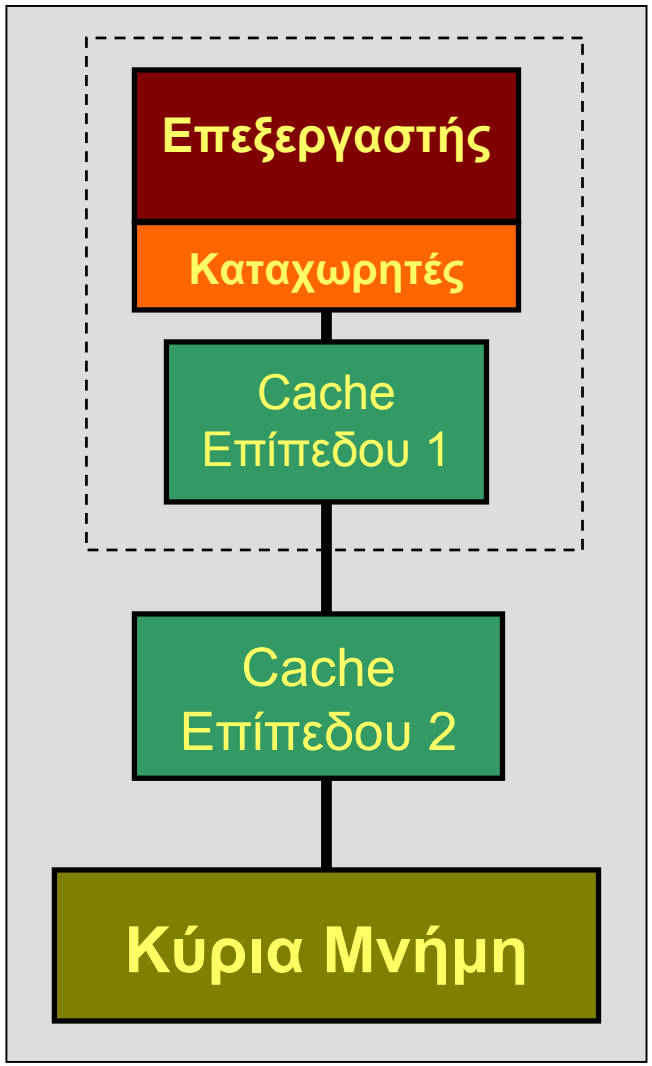
# Εισαγωγή

- Η **μνήμη** είναι ένα από τα βασικά στοιχεία του υλικού ενός υπολογιστή.
- Κύρια κριτήρια:
  - Κόστος / Μέγεθος.
  - Χρόνος Πρόσβασης.
- Η τεχνολογία των μνημών προχωράει πολύ αργά
- Οι γρήγορες μνήμες είναι πολύ ακριβές.
  - Πολύ μικρό χρόνο πρόσβασης.
  - Απαγορευτικό κόστος για μεγάλα μεγέθη.
- Αναγκαζόμαστε να κάνουμε **ιεραρχία μνήμης**.

# Ιεραρχία Μνήμης - I

- Η μνήμη χωρίζεται σε **πολλαπλά επίπεδα** και οργανώνεται έτσι ώστε:
  - Η μικρότερη και γρηγορότερη μνήμη να είναι **πιο κοντά** στον επεξεργαστή (υψηλότερα επίπεδα).
  - Καθώς απομακρυνόμαστε χρησιμοποιούμε αργότερες αλλά μεγαλύτερες μνήμες.
- Η ιεραρχία μνήμης βασίζεται στην έννοια της **τοπικότητας** (στον χώρο και στον χρόνο).
  - Τα δεδομένα που απαιτεί ο επεξεργαστής σε **γειτονικές χρονικές στιγμές**, βρίσκονται σε **γειτονικές διευθύνσεις** στην μνήμη.
- Τα δεδομένα μεταφέρονται μεταξύ των διαφόρων επιπέδων **κατά ομάδες** γειτονικών στοιχείων της μνήμης (**στοιχειώδες τμήμα**).

# Ιεραρχία Μνήμης - II




| Μέγεθος | Ταχύτητα |
|---|---|
| 128 – 1024 bytes | 0.2 – 0.5 nsec |
| 16 – 128 Kb | 10 - 15 nsec |
| 256 Kb – 8 Mb | 20 - 40 nsec |
| 16 MB – 4 GB | 100 - 200 nsec |

# Μνήμες cache – Κρυφές μνήμες

- Τα δεδομένα **μεταφέρονται** από την κύρια μνήμη στην μνήμη κατά στοιχειώδη τμήματα.
- Όταν ο επεξεργαστής ζητάει κάποια διεύθυνση, τότε αυτή αναζητείται στα διάφορα επίπεδα.
  - Αν η ζητούμενη διεύθυνση βρίσκεται στην cache τότε έχουμε **επιτυχία - cache hit**.
  - Αν η ζητούμενη διεύθυνση δεν βρίσκεται στην cache τότε έχουμε **αποτυχία - cache miss**. Και η ζητούμενη διεύθυνση αποθηκεύεται στην cache.
- Μας ενδιαφέρει το μέγιστο **ποσοστό επιτυχίας – hit ratio**.

$$\text{Ποσοστό Επιτυχίας} = \frac{\text{Αριθμός cache hit}}{\text{Συνολικός αριθμός προσπελάσεων}}$$

# Σχεδιασμός ιεραρχημένης μνήμης - I

- Έχουμε **m επίπεδα μνήμης $M_1$, … $M_m$**. Ο επεξεργαστής αναζητάει μια διεύθυνση ξεκινώντας από το επίπεδο $M_1$.
- Το ποσοστό επιτυχίας στην μνήμη i είναι **$h_i$**. Ισχύει **$h_m = 1$**.
- Η πιθανότητα να βρεθεί το ζητούμενο στο επίπεδο i είναι:

$$f_i = (1 - h_1) \cdots (1 - h_{i-1}) h_1 \quad , i = 1, \cdots m$$

- Η $f_i$ καλείται **συχνότητα πρόσβασης** στην $M_i$.
- Ισχύει ότι: **$f_1 >> f_i >> \ldots >> f_m$** (τοπικότητα)
- Αν $t_i$ είναι ο **χρόνος πρόσβασης** στην $M_i$, τότε ισχύει ότι:

$$t_1 < t_2 < \ldots < t_m$$

- Τότε ο **μέσος χρόνος πρόσβασης** στην μνήμη είναι:

$$T_{eff} = \sum_{i=1}^{m} f_i t_i$$

# Σχεδιασμός ιεραρχημένης μνήμης - II

- Αν η μνήμη $M_i$ έχει χωρητικότητα $S_i$ μονάδες και κόστος $c_i$ τότε το συνολικό κόστος είναι:

$$C = \sum_{i=1}^{m} c_i s_i$$

# Πολιτικές αντιστοίχισης - I

- Οι **πολιτικές αντιστοίχησης** καθορίζουν τον τρόπο με τον οποίο οι διάφορες διευθύνσεις αποθηκεύονται στις μνήμες cache.
- Τόσο η cache όσο και η κύρια μνήμη χωρίζονται σε **ισομεγέθη** τμήματα.
  - **Στοιχειώδη τμήματα** για την κύρια μνήμη.
  - **Πλαίσια** για την cache.
- Τα πλαίσια ομαδοποιούνται σε **σύνολα πλαισίων**.
- Με βάση την πολιτική αντιστοίχισης κάθε στοιχειώδες τμήμα (της κύριας μνήμης) **αντιστοιχεί** σε κάποιο σύνολο πλαισίων (της cache) και μπορεί να τοποθετηθεί σε οποιοδήποτε από τα πλαίσια του συνόλου.

# Πολιτικές αντιστοίχισης - ΙΙ

Υπάρχουν 3 πολιτικές αντιστοίχισης:

- **Απευθείας αντιστοίχηση - direct mapping**:
  - Κάθε σύνολο πλαισίων αποτελείται από **ένα μόνο** πλαίσιο.
  - Τα στοιχειώδη τμήματα αντιστοιχίζονται **κυκλικά** στα αντίστοιχα πλαίσια.
  - Δεν απαιτείται **έλεγχος** μέσα στο σύνολο αφού μόνο ένα στοιχειώδες τμήμα μπορεί να υπάρχει εκεί.
  - Μπορεί να υλοποιηθεί με **μνήμη RAM**.
  - Δεν απαιτείται **αλγόριθμος αντικατάστασης**. Αν το ζητούμενο στοιχειώδες τμήμα δεν βρίσκεται μέσα στην cache τότε το υπάρχον εκεί τμήμα **διώχνεται** και αποθηκεύεται το νέο.
  - Παρουσιάζει **μειωμένο** ποσοστό επιτυχίας στην cache.

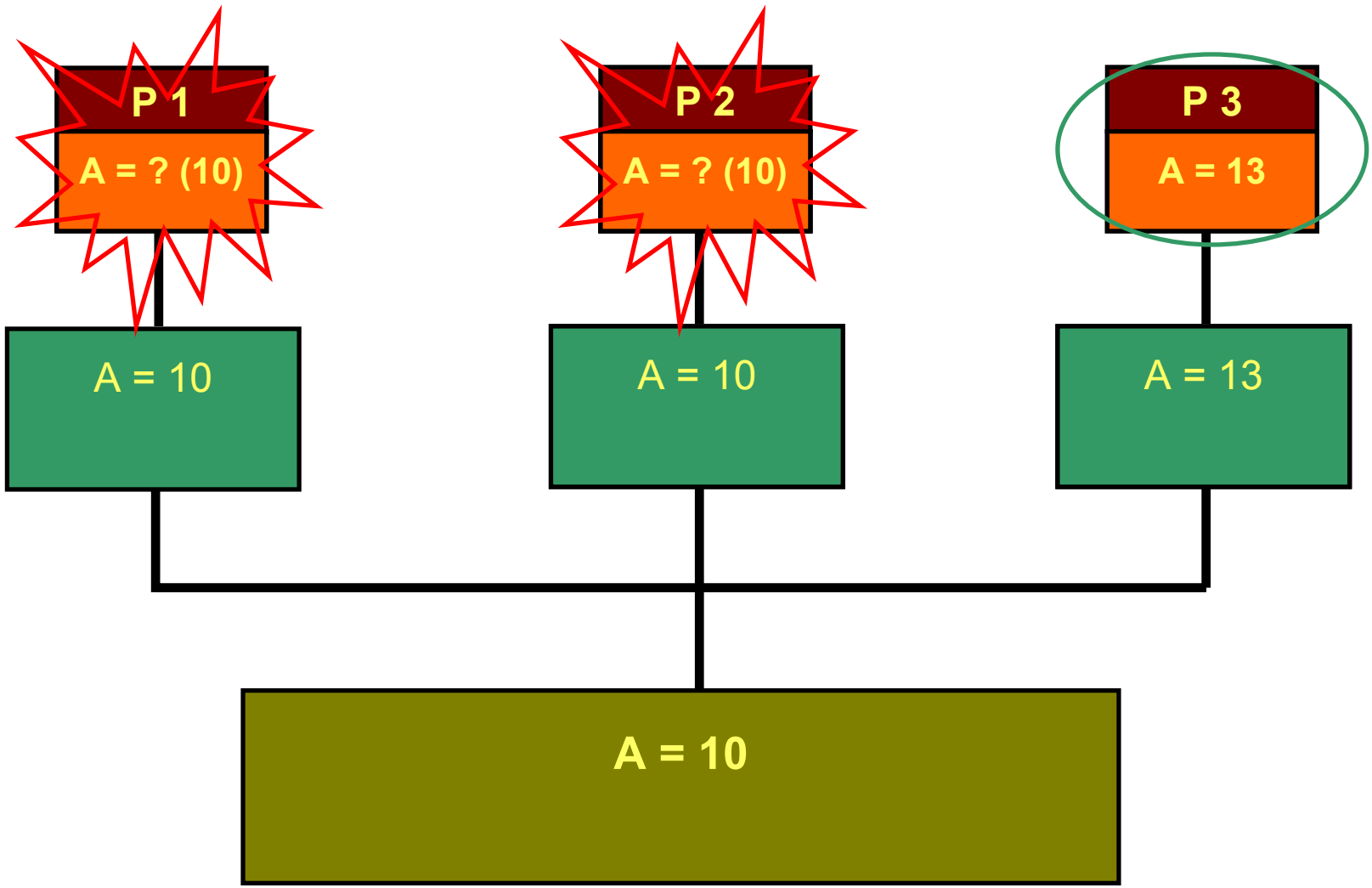
# Πολιτικές αντιστοίχισης - ΙΙΙ

- **Πλήρως προσεταιριστική - fully associative**:
  - **Όλα** τα πλαίσια της cache ανήκουν στο **ίδιο** σύνολο.
  - Οποιοδήποτε στοιχειώδες τμήμα μπορεί να τοποθετηθεί σε οποιοδήποτε πλαίσιο της cache.
  - Σε περίπτωση αστοχίας το νέο στοιχειώδες τμήμα μπορεί να τοποθετηθεί σε οποιοδήποτε πλαίσιο.
  - **Απαιτείται** αλγόριθμος αντικατάστασης.
  - **Απαιτείται έλεγχος** μέσα στο σύνολο με βάση το περιεχόμενο.
  - Θα πρέπει να υλοποιηθεί με **μνήμη CAM**.
  - Παρουσιάζει **πολύ υψηλό** ποσοστό επιτυχίας στην cache, ανάλογα με τον αλγόριθμό αντικατάστασης.
- **Προσεταιριστική κατά σύνολα - set associative**:
  - Συνδυασμός των δύο προηγούμενων.

# Το Πρόβλημα της Συνέπειας ή Συνοχής

- Εμφανίζεται σε παράλληλες μηχανές **κοινής μνήμης**.
- Όταν πολλοί επεξεργαστές προσπελαύνουν τα ίδια δεδομένα.
- Όταν περισσότερες από μία μνήμες cache διαφορετικών επεξεργαστών περιέχουν **αντίγραφα** των τιμών κάποιας θέσης μνήμης
- Οι επεξεργαστές εργάζονται (γενικά) με τα αντίγραφα που υπάρχουν στις μνήμες cache τους.
- Δεν υπάρχει εγγύηση ότι η ανάγνωση από τη μνήμη θα επιστρέψει την **«πιο πρόσφατη» τιμή.**

# Παράδειγμα



# Λύση στο πρόβλημα

- Πρωτόκολλα τα οποία διασφαλίζουν τη συνέπεια των αντιγράφων δεδομένων που υπάρχουν σε διαφορετικές cache και την κύρια μνήμη
- Πολιτικές για την ενημέρωση των άλλων cache:
  - Εγγραφής – Ακύρωσης
  - Εγγραφής – Ενημέρωσης
- Πολιτικές για την ενημέρωση της κύριας μνήμης:
  - Έμμεση Ενημέρωση (Write – Through)
  - Άμεση Ενημέρωση (Write – Back ή Copy – Back)

# Snoopy Πρωτόκολλα - I

- Κάθε επεξεργαστής **παρακολουθεί** (snoops) τις αλληλεπιδράσεις των υπολοίπων επεξεργαστών με τη μνήμη, πάνω από τον **δίαυλο επικοινωνίας - bus**.
- Ορίζεται μια **κατάσταση** για κάθε αντίγραφο στοιχειώδους τμήματος που υπάρχει στην cache και κανόνες μετάβασης από την μια κατάσταση στην άλλη.
  - Χρησιμοποιούνται μηχανές πεπερασμένων καταστάσεων.
  - Ανάλογα με την λειτουργία που κάνει κάποιος επεξεργαστής σε κάποιο στοιχειώδες τμήμα οι υπόλοιποι ενημερώνονται αντίστοιχα, αλλάζοντας καταστάσεις στα αντίγραφα που έχουν τοπικά.
- Απαιτείται υποστήριξη από τον ελεγκτή της cache και τον ελεγκτή του διαύλου.

# Snoopy Πρωτόκολλα - ΙΙ

- Μειονέκτημα:
  - Ανάγκη για δίαυλο ώστε ο κάθε επεξεργαστής να μπορεί να παρακολουθεί τις αλληλεπιδράσεις των άλλων με τη μνήμη.
- Αλληλεπίδραση μεταξύ μεγέθους cache και μεγέθους στοιχειώδους τμήματος:
  - Το ποσοστό επιτυχίας αυξάνει  όσο αυξάνει το μέγεθος του στοιχειώδους τμήματος (μέχρι κάποιο σημείο -> **σημείο μόλυνσης** των δεδομένων).
  - Όσο αυξάνει το μέγεθος της cache τόσο μεγαλύτερο είναι το μέγεθος στοιχειώδους τμήματος που αντιστοιχεί στο σημείο μόλυνσης.

# Πρωτόκολλα καταλόγου - Ι

- Σε κάθε μνήμη cache υπάρχει ένας **ιδιωτικός κατάλογος** που περιέχει πληροφορίες σχετικά με το ποια στοιχειώδη τμήματα βρίσκονται στην συγκεκριμένη cache.
- Υπάρχει και ένας **κεντρικός κατάλογος** που περιέχει αντίγραφα των ιδιωτικών καταλόγων.
- Όταν μεταβάλλεται τοπικά κάποιο τμήμα:
  - Ο ελεγκτής cache βάσει του κεντρικού καταλόγου βρίσκει σε ποιες μνήμες cache υπάρχει κάποιο στοιχειώδες τμήμα.
  - Ενημερώνει έπειτα τους άλλους επεξεργαστές και τους καταλόγους τους.

# Πρωτόκολλα καταλόγου - ΙΙ

- Εναλλακτική προσέγγιση: **Πλήρες σχήμα**
  - Σε κάθε στοιχειώδες τμήμα αντιστοιχεί ένα **διάνυσμα παρουσίας** που δείχνει το ποιοι επεξεργαστές έχουν αντίγραφο αυτού του στοιχειώδους τμήματος.
  - Ταχύτερη αναζήτηση και ενημέρωση όταν έχουμε αλλαγές.
  - Δημιουργεί σημαντική επιβάρυνση μνήμης, επειδή σε κάθε διάνυσμα απαιτούνται τόσα bits όσα και οι επεξεργαστές.
- Πλεονέκτημα πρωτοκόλλων καταλόγου
  - Αίρεται η ανάγκη για δίαυλο.
  - Κλιμακωσιμότητα, για μεγάλο αριθμό επεξεργαστών.

# Πολύπλεξη μνήμης - I

- Η πολύπλεξη μνήμης χρησιμοποιείται όταν έχουμε **πολλαπλά διαμερίσματα μνήμης**.
- Μπορούμε να πετύχουμε **παράλληλη πρόσβαση** στα δεδομένα.
- Εξετάζουμε **δύο τύπους** πολύπλεξης της μνήμης

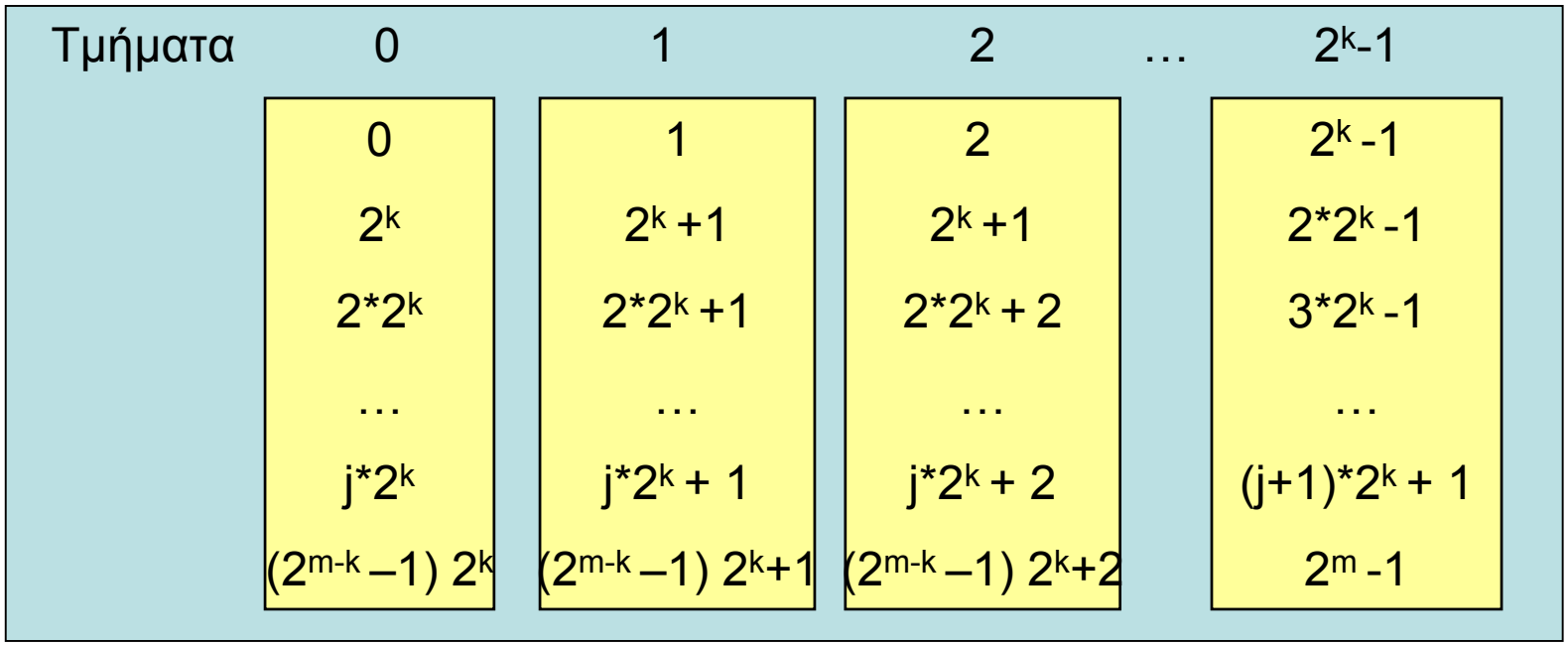
# Πολύπλεξη μνήμης - II

- **Αδρή πολύπλεξη**: Τα δεδομένα αποθηκεύονται σε κάθε τμήμα μνήμης διαδοχικά.
  - Εύκολα επεκτάσιμη.
  - Η προσπέλαση σε διαδοχικές θέσεις γίνεται ακολουθιακά.

| Τμήματα | 0 | 1 | 2 | … | $2^k-1$ |
|---|---|---|---|---|---|
| | 0 | $2^{m-k}$ | $2*2^{m-k}$ | | $(2^k-1)*2^{m-k}$ |
| | 1 | $2^{m-k}+1$ | $2*2^{m-k}+1$ | | $(2^k-1)*2^{m-k}+1$ |
| | 2 | $2^{m-k}+2$ | $2*2^{m-k}+2$ | | $(2^k-1)*2^{m-k}+2$ |
| | … | … | … | | … |
| | j | $2^{m-k}+j$ | $2*2^{m-k}+j$ | | $(2^k-1)*2^{m-k}+j$ |
| | $2^{m-k}-1$ | $2*2^{m-k}-1$ | $3*2^{m-k}-1$ | | $2^m-1$ |

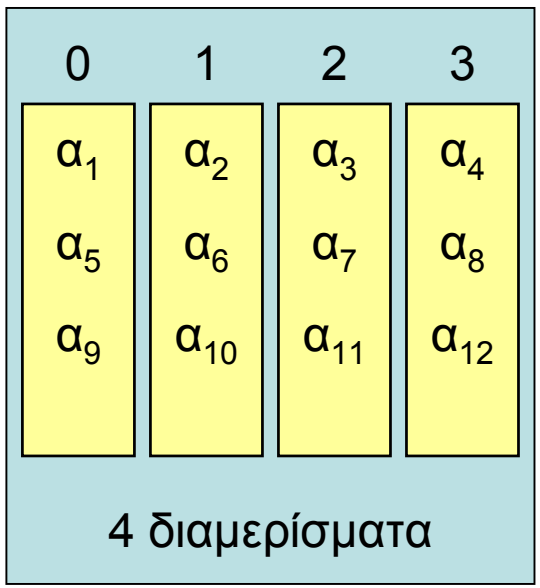
# Πολύπλεξη μνήμης - III

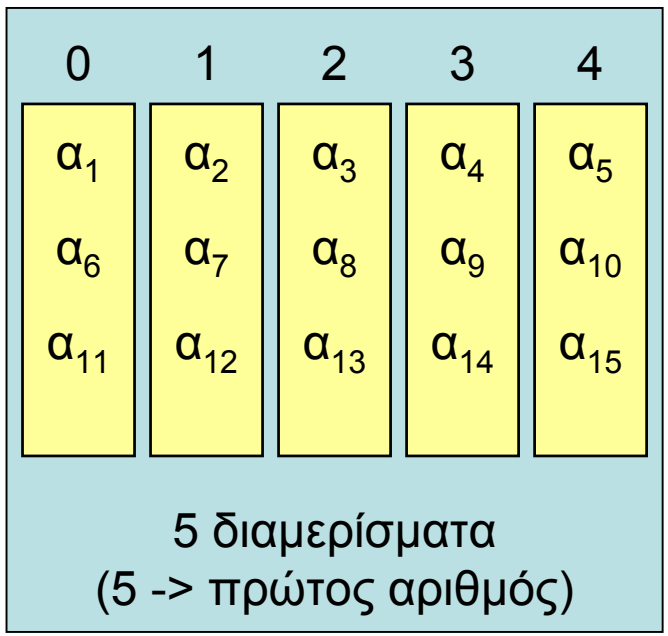
- **Λεπτή πολύπλεξη**: Τα διαδοχικά δεδομένα αποθηκεύονται σε διαδοχικά τμήματα μνήμης.
  - Η προσπέλαση σε διαδοχικές θέσεις γίνεται παράλληλα.
  - Τα k LS bits δείχνουν το αντίστοιχο τμήμα μνήμης.
  - Δυσκολίες στην επέκταση.



# Παράλληλη προσπέλαση για διανύσματα - I

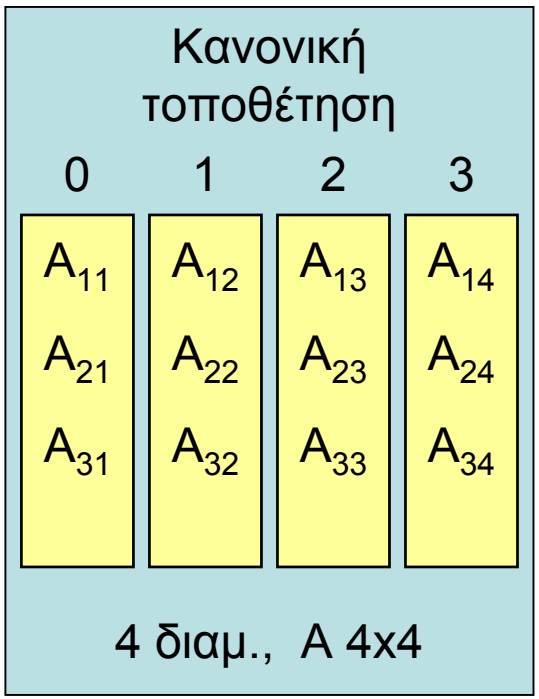
- Πολλά προγράμματα απαιτούν την προσπέλαση ενός διανύσματος με κάποιον συγκεκριμένο τρόπο.
  - Π.χ. Διάβασε τα k πρώτα στοιχεία του διανύσματος ή διάβασε τα στοιχεία του διανύσματος που βρίσκονται στις μονές θέσεις.

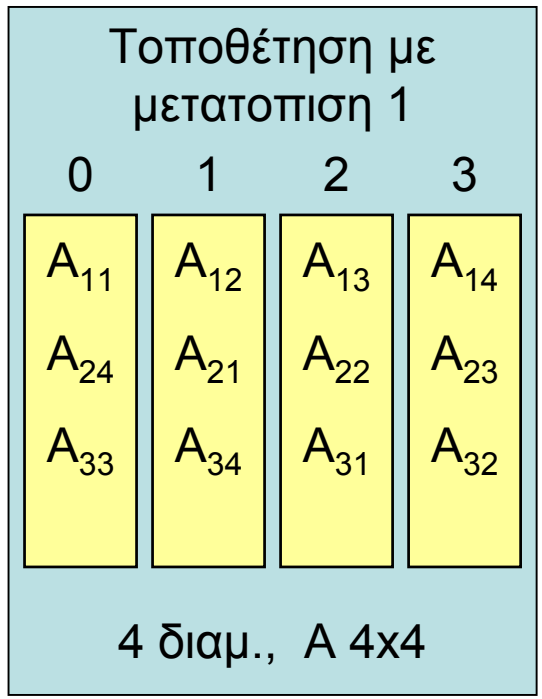


# Παράλληλη προσπέλαση για πίνακες

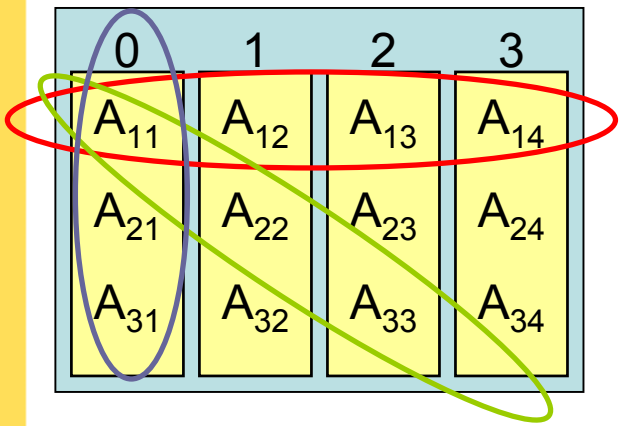
- Πώς θα προσπελάσουμε τα στοιχεία του πίνακα **A[n][n]** με διάφορους κανονικούς τρόπους.
  - Π.χ. προσπέλαση κατά στήλες, γραμμές ή διαγώνιούς.
- Με την προϋπόθεση ότι ο αριθμός των διαμερισμάτων **δ >= n**.



# Παράλληλη προσπέλαση για πίνακες - I

- Πώς θα προσπελάσουμε τα στοιχεία του πίνακα **A[n][n]** με διάφορους κανονικούς τρόπους ?
  - Π.χ. προσπέλαση κατά στήλες, γραμμές ή διαγώνιούς.
- Με την προϋπόθεση ότι ο αριθμός των διαμερισμάτων **δ >= n**.
- Τοποθετούμε τα στοιχεία με τον κλασσικό τρόπο.

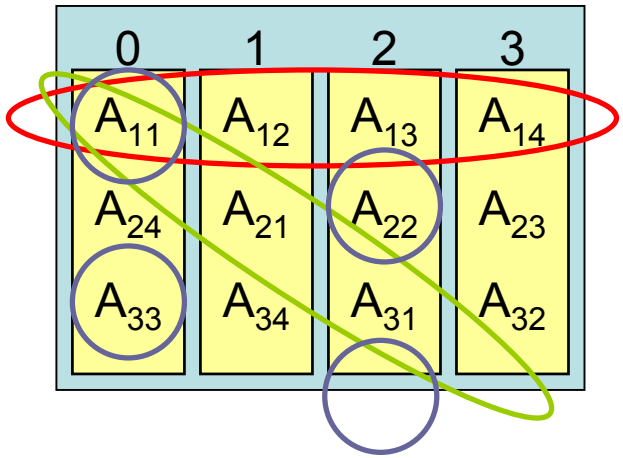



γραμμές διαγώνιους

στήλες

# Παράλληλη προσπέλαση για πίνακες - ΙΙ

- Τοποθετούμε τα στοιχεία χρησιμοποιώντας **μετατόπιση 1** για κάθε γραμμή.




γραμμές στήλες

διαγώνιους

- Χρησιμοποιώντας μετατόπιση 1 λύσαμε το πρόβλημα για τις στήλες αλλά όχι για τις διαγώνιους.
- Γενικά ισχύει:

ότι για **n άρτιο και δ = n δεν μπορούμε** να έχουμε παράλληλη προσπέλαση κατά γραμμές, στήλες και διαγώνιους ταυτόχρονα.

# Παράλληλη προσπέλαση για πίνακες
# Γενική Λύση

- Έστω το διάνυσμα $\boldsymbol{\alpha_1, \alpha_2,\ldots,\alpha_n}$. Τοποθετούμε το $\alpha_1$ στην θέση **s**. Και τα υπόλοιπα στοιχεία **ανά d θέσεις**.
- Το διάνυσμα καταλαμβάνει τις θέσεις s, s+d, s+2d,… με κυκλική επαναφορά στα πρώτα διαμερίσματα. Άρα το στοιχείο $\alpha_i$ τοποθετείται στην θέση:

$$(d(i-1)+s) \bmod (\delta)$$

- Αποδεικνύεται ότι για να είναι δυνατή η παράλληλη πρόσβαση στο διάνυσμα αρκεί να ισχύει:

$$\delta \geq n \cdot \text{ΜΚΔ}(d,\delta)$$

- Για την περίπτωση ενός πίνακα έχουμε το **«σχήμα($\boldsymbol{\delta_1}$, $\boldsymbol{\delta_2}$)»**. Όπου έχουμε $d = \delta_1$ για τις γραμμές, $d = \delta_2$ για τις στήλες και $d = \delta_1 + \delta_2$ για τις διαγώνιους.

# Παράδειγμα 4.11-1

Για ένα πίνακα Α 4x4, να βρεθεί ο ελάχιστος αριθμός διαμερισμάτων δ και να τοποθετηθούν τα στοιχεία του πίνακα:

1.Για παράλληλη προσπέλαση κατά στήλες και διαγώνιες.

2.Για παράλληλη προσπέλαση κατά γραμμές, στήλες και διαγώνιες.

## ΛΥΣΗ

1. Δοκιμάζουμε d=1 για στήλες και διαγώνιες. Αρκεί:

$$\delta >= n*ΜΚΔ(1,\delta) = 4*1=4, \text{ άρα } \delta = 4$$

Δηλαδή χρειαζόμαστε 4 διαμερίσματα.

| | | | |
|---|---|---|---|
| $A_{11}$ | $A_{21}$ | $A_{31}$ | $A_{41}$ |
| $A_{12}$ | $A_{22}$ | $A_{32}$ | $A_{42}$ |
| $A_{13}$ | $A_{23}$ | $A_{33}$ | $A_{43}$ |
| $A_{14}$ | $A_{24}$ | $A_{34}$ | $A_{44}$ |

2. Δοκιμάζουμε σχήμα (1,1), δηλαδή d=1 για γραμμές και στήλες.

Αρκεί:

$$\delta >= n*\text{ΜΚΔ}(1,\delta) = 4*1=4, \text{ για γραμμές και στήλες}$$

και

$$\delta >= n*\text{ΜΚΔ}(2,\delta) = 8, \text{ αν } \delta \text{ είναι ζυγός}$$

$$= 4, \text{ αν } \delta \text{ είναι μονός}$$

Άρα η ελάχιστη τιμή για το δ είναι **δ = 5**.

Δηλαδή χρειαζόμαστε 5 διαμερίσματα.