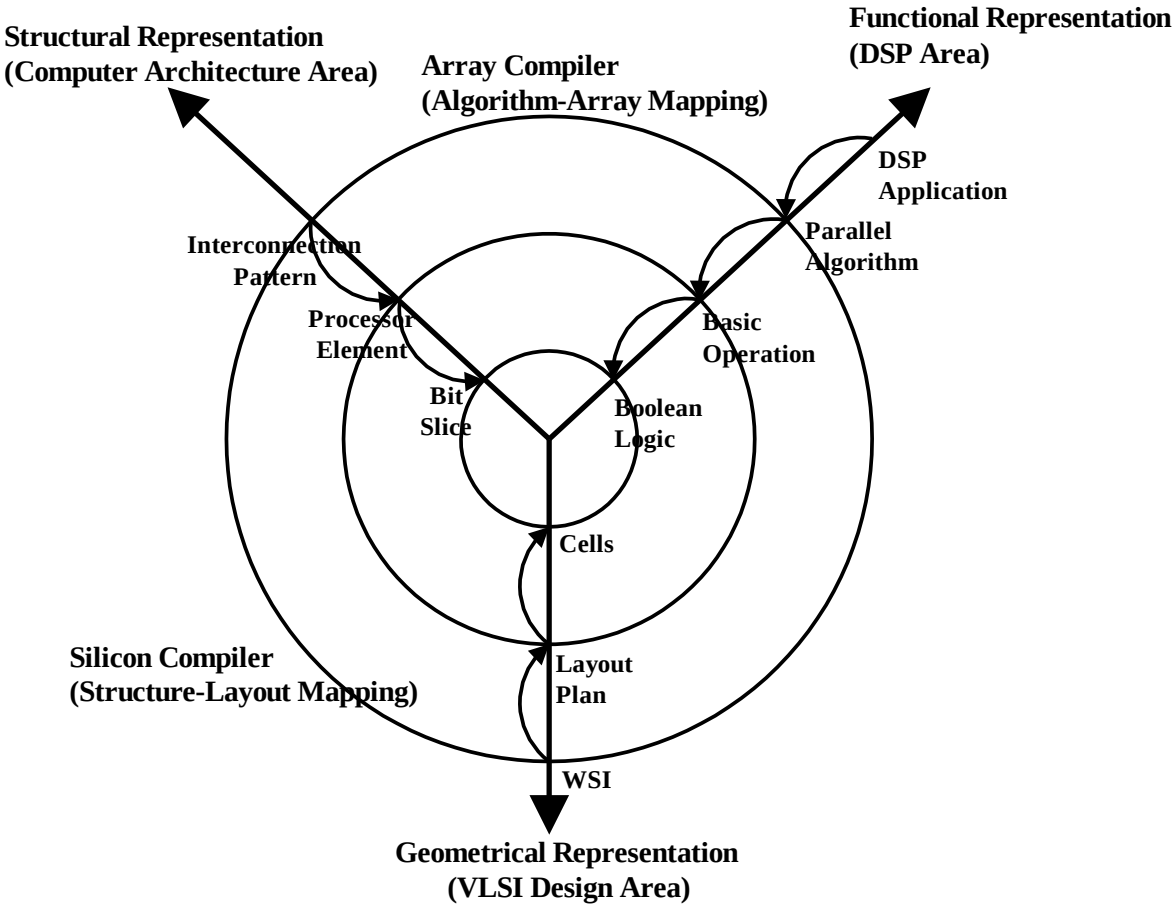
ΕΝΣΩΜΑΤΩΜΈΝΑ ΣΥΣΤΉΜΑΤΑ

ΑΛΓΟΡΙΘΜΙΚΟΊ ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΊ

Μετασχηματισμοί πηγαίου κώδικα

Y-CHART


Structural Representation
(Computer Architecture Area)
Array Compiler
(Algorithm-Array Mapping)
Functional Representation
(DSP Area)
DSP
Application
Parallel
Algorithm
Interconnection
Pattern
Processor
Element
Basic
Operation
Bit
Slice
Boolean
Logic
Cells
Silicon Compiler
(Structure-Layout Mapping)
Layout
Plan
WSI
Geometrical Representation
(VLSI Design Area)

Τύποι Εξαρτήσεων Δεδομένων
Types of Data Dependences

- **Εξάρτηση Ροής (Flow dependence)**
  - **S1:** a = c*10
  - **S2:** d = 2*a + c
- **Αντιεξάρτηση (Anti-dependence)**
  - **S1:** e = f*4 + g
  - **S2:** g = 2*h

Ανάλυση Εξαρτήσεων Βρόχου

```
do i1 = l1, u1
   do i2 = l2, u2
      ...
         do id = ld, ud
1           a[f1(i1, ..., id), ..., fm(i1, ..., id)] = ...
2           ...= a[g1(i1, ..., id), ..., gm(i1, ..., id)]
         end do
      ...
   end do
end do
```

```
      do i = 2, n
1       a[i] = a[i] + c
2       b[i] = a[i-1] * b[i]
      end do
```

Για τον υπολογισμό της πληροφορίας εξάρτησης σε φωλιασμένους για βρόχους, το βασικό πρόβλημα είναι η κατανόηση της χρήσης των πινάκων. Τα βαθμωτά δεδομένα είναι σχετικά εύκολο να διαχειριστούν. Για να παρακολουθήσετε τη συμπεριφορά του πίνακα, ο μεταγλωττιστής πρέπει να αναλύσει το δείκτη εκφράσεις σε κάθε αναφορά σε πίνακα.

Αλγοριθμικοί Μετασχηματισμοί

- **Μετασχηματισμοί με βάση τη ροή δεδομένων στο βρόχο**
- **Αναδιάταξη Βρόχων**
- **Αναδιάρθρωση Βρόχων**
- **Μετασχηματισμοί Αντικατάστασης Βρόχου**
- **Μετασχηματισμοί πρόσβασης στη μνήμη**
- **Μερική αξιολόγηση**
- **Εξάλειψη του πλεονασμού**
- **Μετασχηματισμοί για κλήσης συναρτήσεων/υποπρογραμμάτων**

Μετασχηματισμοί Αναδιάταξης Βρόχων
Loop Reordering Transformations

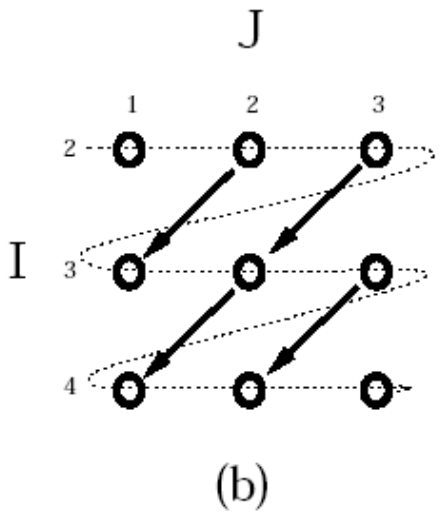
- Αλλαγή της σχετικής ακολουθίας εκτέλεσης των επαναλήψεων του/των φωλιασμένου/νων βρόχου/ων
  - Ανάδειξη παραλληλίας και βελτίωση της τοπικότητας στην μνήμη.

Μετασχηματισμοί Αναδιάταξης Βρόχων (1)

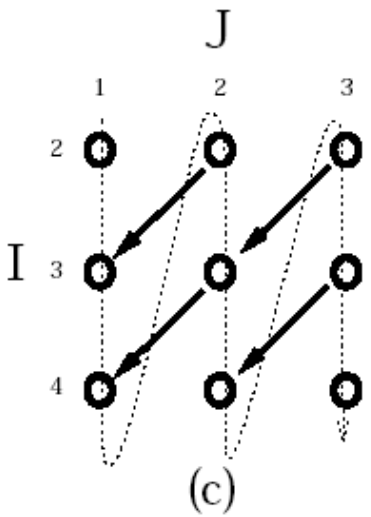
- **Εναλλαγή βρόχου (Loop Interchange)**

```
do i = 2, n
  do j = 1, n-1
    a[i,j] = a[i-1,j+1]
  end do
end do
```

(a)



(b)



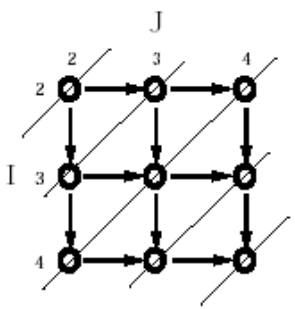
(c)

Μετασχηματισμοί Αναδιάταξης Βρόχων (2)

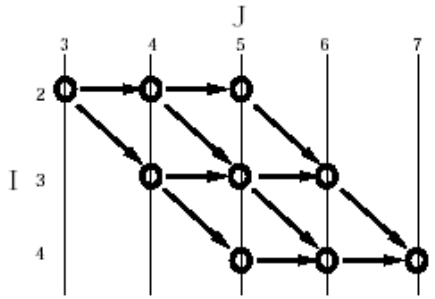
- Στρέβλωση Βρόχου (Loop Skewing) ➔ Στρέβλωση της εκτέλεσης των επαναλήψεων (skew iterations execution)
- Useful for Loop Interchange

```
do i = 2, n-1
  do j = 2, m-1
    a[i,j] = (a[i-1,j] + a[i,j-1] +
              a[i+1,j] + a[i,j+1])/4
  end do
end do
```

(a) original code: dependences $\{(1,0),(0,1)\}$



(b) original space



(c) skewed space

Parallel iterations

Skewing factor "i"

```
do i = 2, n-1
  do j = i+2, i+m-1
    a[i,j-i] = (a[i-1,j-i] + a[i,j-1-i] +
                a[i+1,j-i] + a[i,j+1-i])/4
  end do
end do
```

(d) skewed code: dependences $\{(1,1),(0,1)\}$

```
do j = 4, m+n-2
  do i = max(2,j-m+1), min(n-1,j-2)
    a[i,j-i] = (a[i-1,j-i] + a[i,j-1-i] +
                a[i+1,j-i] + a[i,j+1-i])/4
  end do
end do
```

(e) skewed and interchanged code: dependences $\{(1,0),(1,1)\}$

Partial Evaluation

- Partial evaluation refers to the general technique of performing part of a computation at compile time.
- Constant propagation is one of the most important optimizations that a compiler can perform and a good optimizing compiler will apply it aggressively.
- Programs typically contain many constants; by propagating them through the program, the compiler can do a significant amount of pre-computation.
- The propagation reveals many opportunities for other optimizations.

```
n = 64
c = 3
do i = 1, n
  a[i] = a[i] + c
end do
```

(a) original code

```
do i = 1, 64
  a[i] = a[i] + 3
end do
```

(b) after constant propagation

- Constant folding is a companion to constant propagation: when an expression contains an operation with constant values as operands, the compiler can replace the expression with the result.

Partial Evaluation (2)

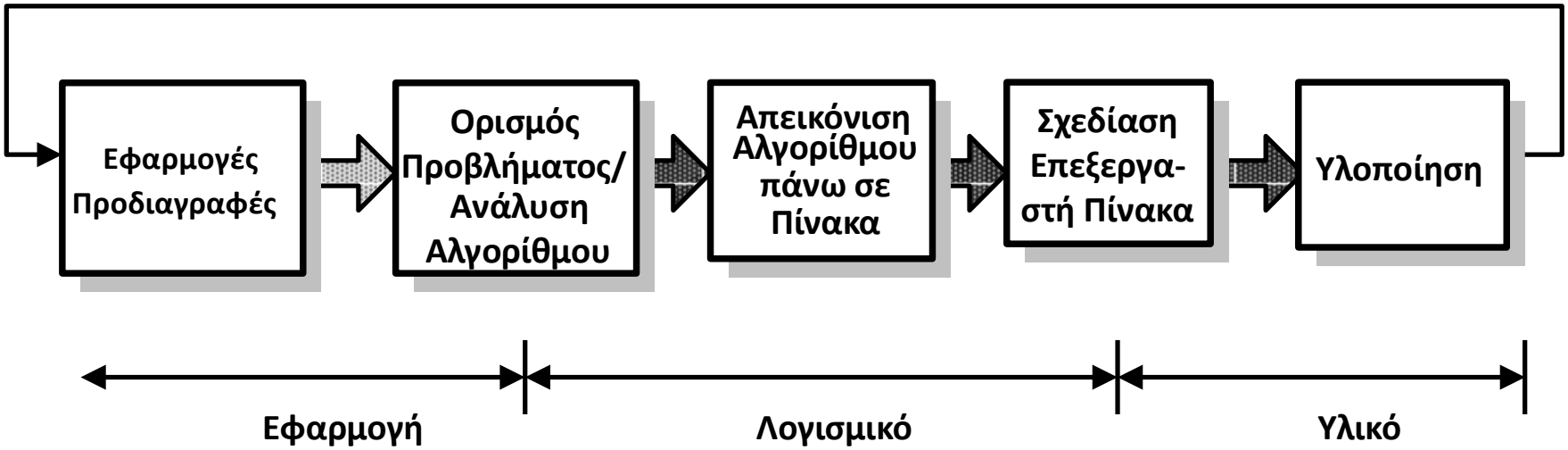
- Strength Reduction: replace an expensive operator with an equivalent less expensive operator.

Expression	Reduced Expr.	Datatypes
$x \times 2$	$x + x$	integer, real
x^2	$x \times x$	integer, real
$x^{c.5}$	$x^c \times \sqrt{x}$	real
$i \times 2^c$	$i \ll c$	integer
$(a, 0) + (b, 0)$	$(a + b, 0)$	complex
$\text{len}(s_1 \ \& \ s_2)$	$\text{len}(s_1) + \text{len}(s_2)$	string

Συστολικές αρχιτεκτονικές

SIMD Architecture

Μεθοδολογία Σχεδιασμού



Μεθοδολογία Σχεδιασμού που βασίζεται σε γράφο (Graph-based Methodology)

- **Φάση 1: Σχεδίαση Γράφου Εξάρτησης (Dependence Graph Design)**
- **Φάση 2: Σχεδίαση Γράφου Ροής Σήματος (Signal Flow Graph Design)**
- **Φάση 3: Σχεδίαση Επεξεργαστή Πίνακα (Array Processor Design)**

Φάση 1 Μεθοδολογίας

- **Σχεδίαση Εξαρτήσεων (Dependence Design (DG))**

  Για ένα συγκεκριμένο πρόβλημα, ο σχεδιαστής πρέπει να εντοπίσει ένα κατάλληλο αλγόριθμο που περιγράφεται από μια κατάλληλη έκφραση. Ένας αναδρομικός αλγόριθμος μπορεί εύκολα να μετατραπεί σε ένα DG με την χρήση χωρο-χρονικού γραμμικού χώρου χρησιμοποιώντας κατάλληλα βέλη για να δειχνούν οι εξαρτήσεις στο γραμμικό χώρο

Φάση 2 Μεθοδολογίας

- Γράφος ροής σήματος (SGF)

  Η έκφραση SGF αποτελείται κυρίως από κόμβους επεξεργασίας, ακμές επικοινωνίας και καθυστερήσεις. Ένας απλός τρόπος απεικόνησης ενός DG επάνω σε ένα SGF γίνεται με την βοήθεια προβολής, ηοποία εκχωρεί τις λειτουργίες από όλο υς τους κόμβους κατά μήκος μιας γραμμής σε ένα στοιχείο επεξεργασίας (PE).

Ανάθεση και Χρονοδρομολόγηση Επεξεργαστή
Processor Assignment & Scheduling (1)

- **Δύο είναι οι βασικές θεωρήσεις για την απεικόμιση ενός DG σε SFG:**

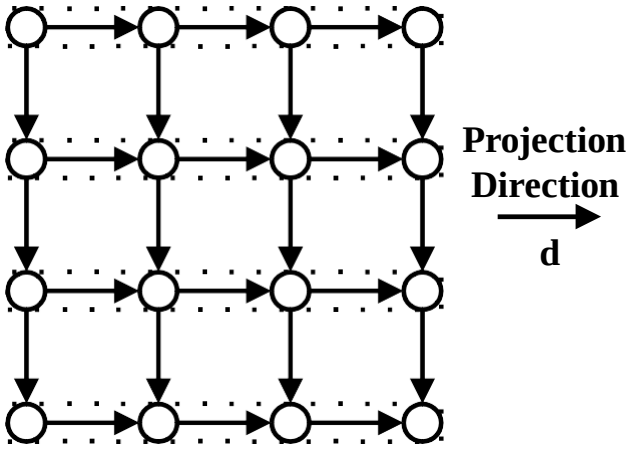
  Σε ποιούς επεξεργαστές θα πρέπει να απεικονιστούν οι λειτουργίες;

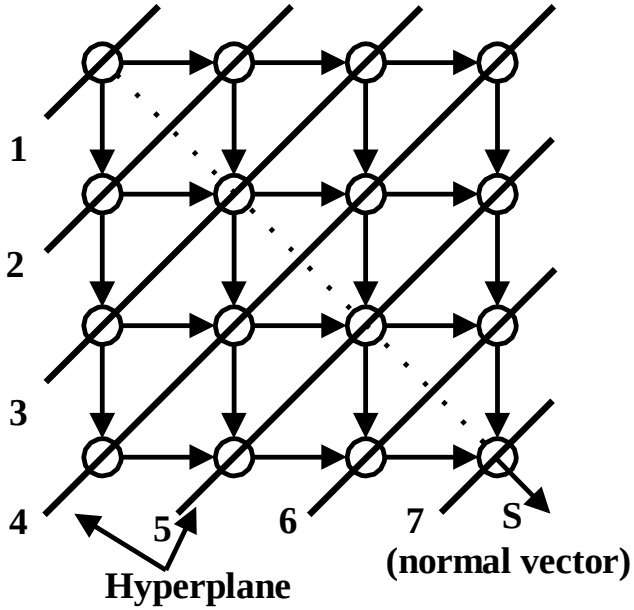
  Ανάθεση Επεξεργαστή (*Processor Assignment)*

  Με ποια σειρά θα εκτελεστούν οι πράξεις που θα εκχωρηθούν σε ένα επεξεργαστή; processor?

  Χρονοδρομολόγηση (*Scheduling)*

Ανάθεση και Χρονοδρομολόγηση Επεξεργαστή
Processor Assignment & Scheduling (2)





Υπερεπίπεδο

Αποδεκτές Γραμμικές χρονοδρομολογήσεις
Permissible Linear Schedules (2)

Η χρονοδρομολόγηση
είναι αποδεκτή εάν και μόνο εάν: (1) όλα τα βέλη εξάρτησης που ρέουν προς την ίδια κατεύθυνση κατά πλάτος του υπερεπιπέδου και (2) τα υπερεπιπέδα δεν είναι παράλληλα με το διάνυσμα προβολής

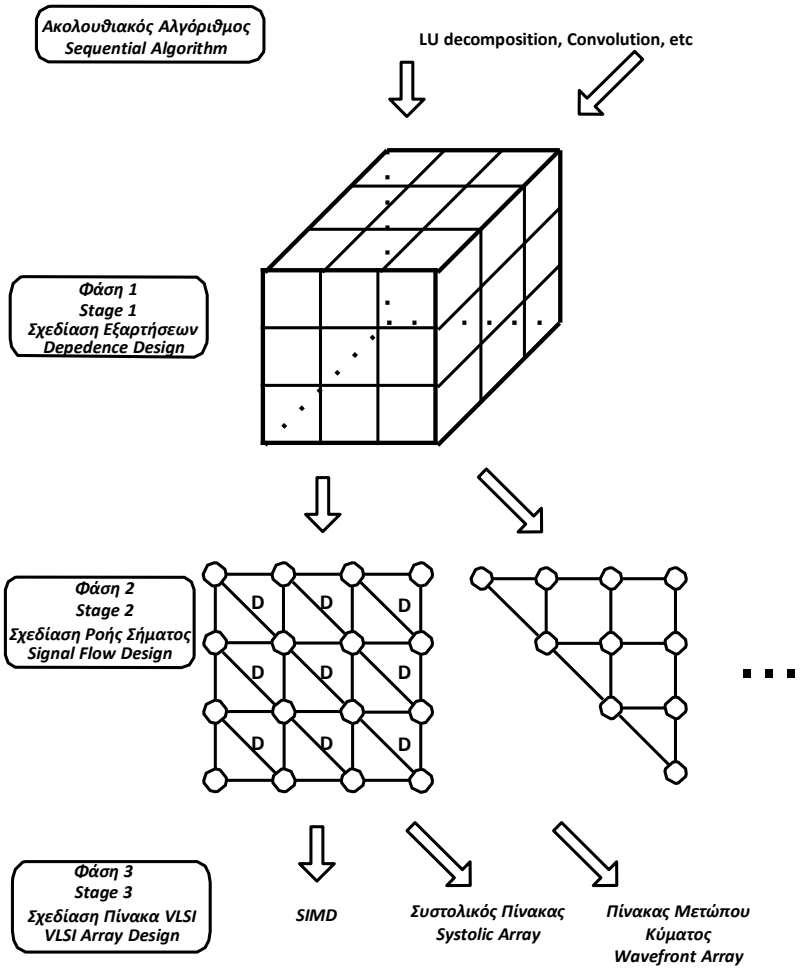
$$\vec{s}^{\,T}\,\vec{e} \geq 0 \quad (1) \qquad\qquad \vec{s}^{\,T}\,\vec{d} \geq 0 \quad (2)$$

METHODOLOGY STAGE (3)

- **Σχεδίαση Πίνακα Επεξεργαστή (Array Processor Design)**

  Το SFG από την Φάση 2 μπορεί να απεινονιστεί σε SIMD, συστολικό πίνακα, πίνακα μετώπου κύματος, κλπ. Για παράδειγμα, για την μετατροπή ενός πίνακα SFG σε συστολικό πίνακα array, εφαρμόζεται η διαδικασία του επαναχρονισμού (retiming)

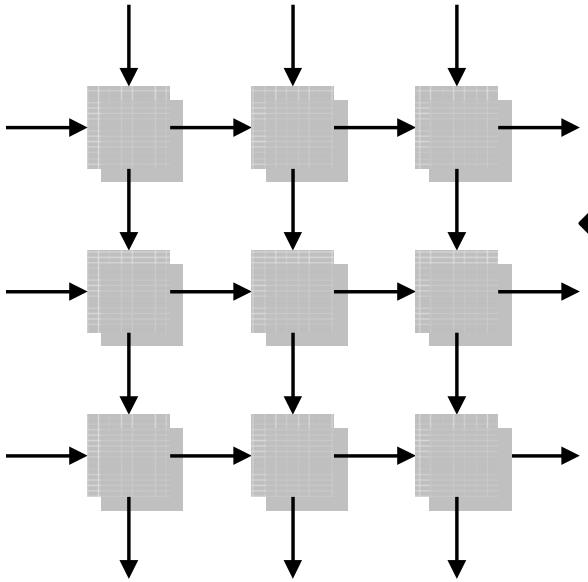
Μεθοδολογία Σχεδιασμού που βασίζεται σε γράφο (Graph-based Methodology) (2)



Γιατί Συστολικές Αρχιτεκτονικές
WHY SYSTOLIC ARCHITECTURES?

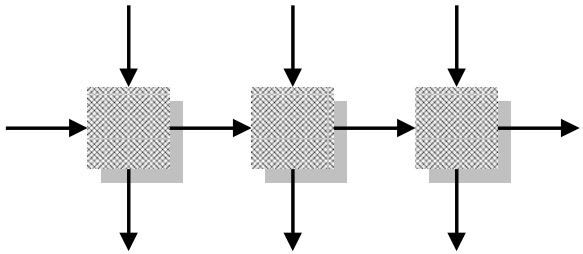
- **Οι κυριότεροι παράγοντες για την χρήση των συστολικών αρχιτεκτονικών για εφαρμογές επεξεργασίας σήματος:**
  - **Απλή και Κανονική Σχεδίαση**
  - **Ταυτοχρονισμός και Επικοινωνία**
  - **Εξισορρόπηση Υπολογισμού και Εισόδου/Εξόδου**

Συστολικές Αρχιτεκτονικές
Systolic Architectures



Αρχιτεκτονική Πίνακα 2-Διαστάσεων

Γραμμική Αρχιτεκτονική Πίνακα



Εφαρμογή: Πολλαπλασιασμός Πίνακα (1)

- Μαθηματική Έκφραση:

$$c_{ij} = \sum_{k=1}^{N} a_{ik}\, b_{kj}$$

- Ακολουθιακός Κώδικας:

```
for i  from 1 to N1
for  j  from 1 to N2
for k  from 1 to N3
c(i, j, k)=c(i, j, k−1)+a(i, k)b(k, j)
```

Εφαρμογή: Πολλαπλασιασμός Πίνακα (2)

- Κώδικας Μοναδικής Καταχώρησης (Single Assignment Code) ή Ομοιόμορφες Επαναληπτικές Εξισώσεις (Uniform Recurrent equation)

for i ***from*** 1 ***to*** N_1

for j ***from*** 1 ***to*** N_2

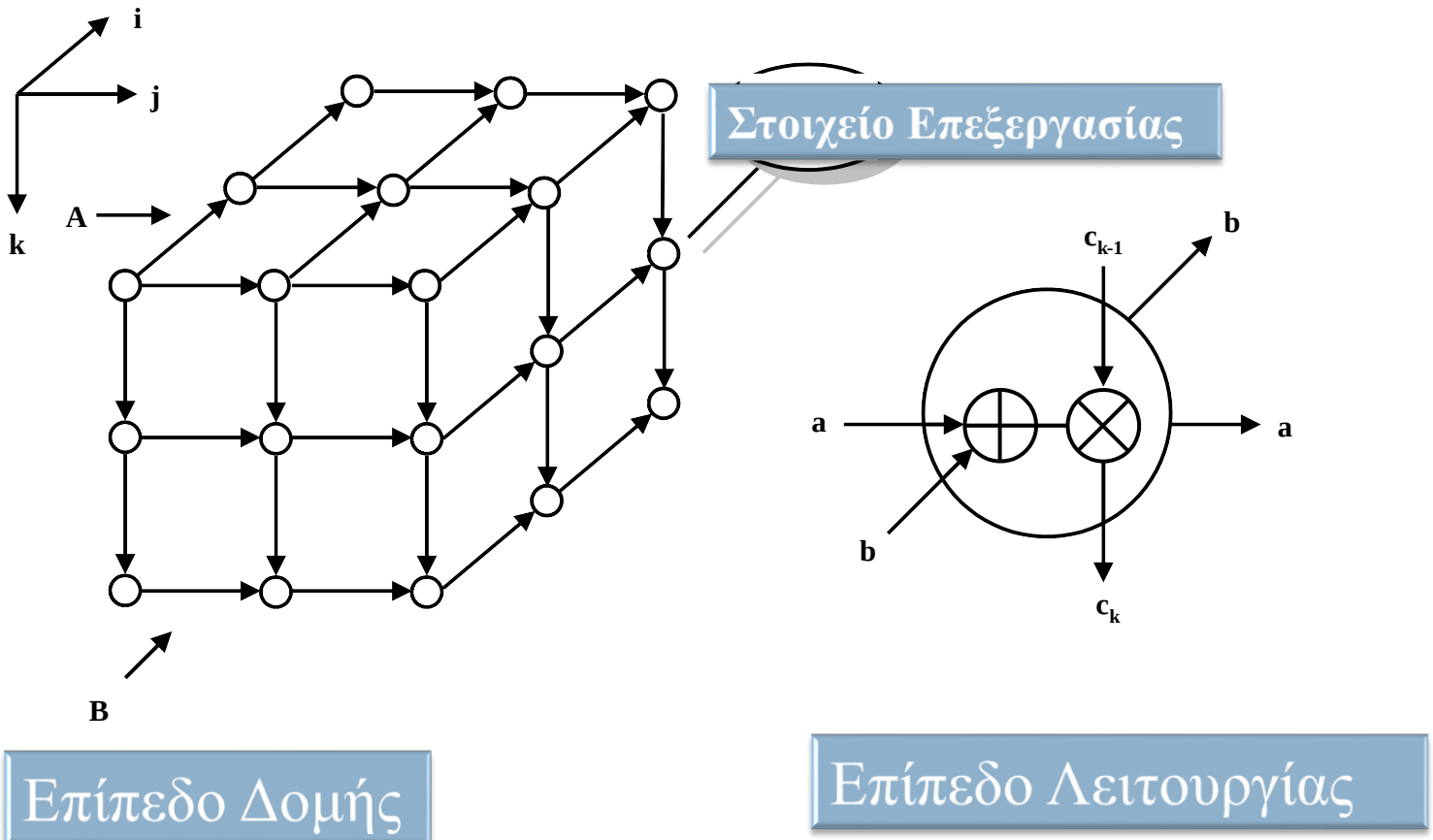
for k ***from*** 1 ***to*** N_3

$$a(i,j,k)=a(i,j-1,k)$$

$$b(i,j,k)=b(i-1,j,k)$$

$$c(i,j,k)=c(i,j,k-1)+a(i,j,k)b(i,j,k)$$

Γράφος Εξάρτησης: Πολλαπλασιασμός Πίνακα
Dependence Graph: Matrix multiplication



ΕΠΙΠΡΌΣΘΕΤΕΣ ΠΛΗΡΟΦΟΡΊΕΣ

Algebraic Approach for SFG projection (1)

- **Node mapping: This mapping assigns the node activities in the DG to processors. The index set of nodes of the SFG are represented by mapping $P : R \rightarrow I^{n-1}$ where R is the index set of the nodes of the DG, and I^{n-1} is the Cartesian product of (n-1) integers. The mapping of a computation c in the SFG is found by:**

$$n = P^T c$$

where the *processor basis P*, denoted by an n x (n-1) matrix, is orthogonal to $\vec{d}$. Mathematically,

$$P^T \vec{d} = 0$$

Algebraic Approach for SFG projection (2)

- **Arc mapping: This mapping maps the arcs of the DG to the edges of the SFG. The set of edges *e* into each node of the SFG and number of delays D(e) on every edge are derived from the set of dependence edges b at each point in the DG by:**

$$\begin{bmatrix} D(\vec{e}) \\ \cdots \\ \vec{e} \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \cdots \\ P^T \end{bmatrix} \begin{bmatrix} \vec{b} \end{bmatrix}$$

Algebraic Approach for SFG projection (3)

- **I/O mapping: The SFG node position, *n*, and time, *t*(*c*), of an input of the DG computation *c* is derived as:**

$$\begin{bmatrix} t(\mathbf{c}) \\ \cdots \\ \mathbf{n} \end{bmatrix} = \begin{bmatrix} \vec{s}^T \\ \cdots \\ P^T \end{bmatrix} [\mathbf{c}]$$

- **A similar mapping applies to output nodes**

Αναφορές

- K. Parhi, “VLSI DIGITAL SIGNAL PROCESSING SYSTEMS: DESIGN AND IMPLEMENTATION”, John Wiley, 1999
- S.Y. Kung, “VLSI Array Processors”, Prentice Hall, 1988
- H.T. Kung, “Why Systolic Architectures?”, IEEE Computers Magazine, vol. 15, pp. 37-45, Jan. 1982.

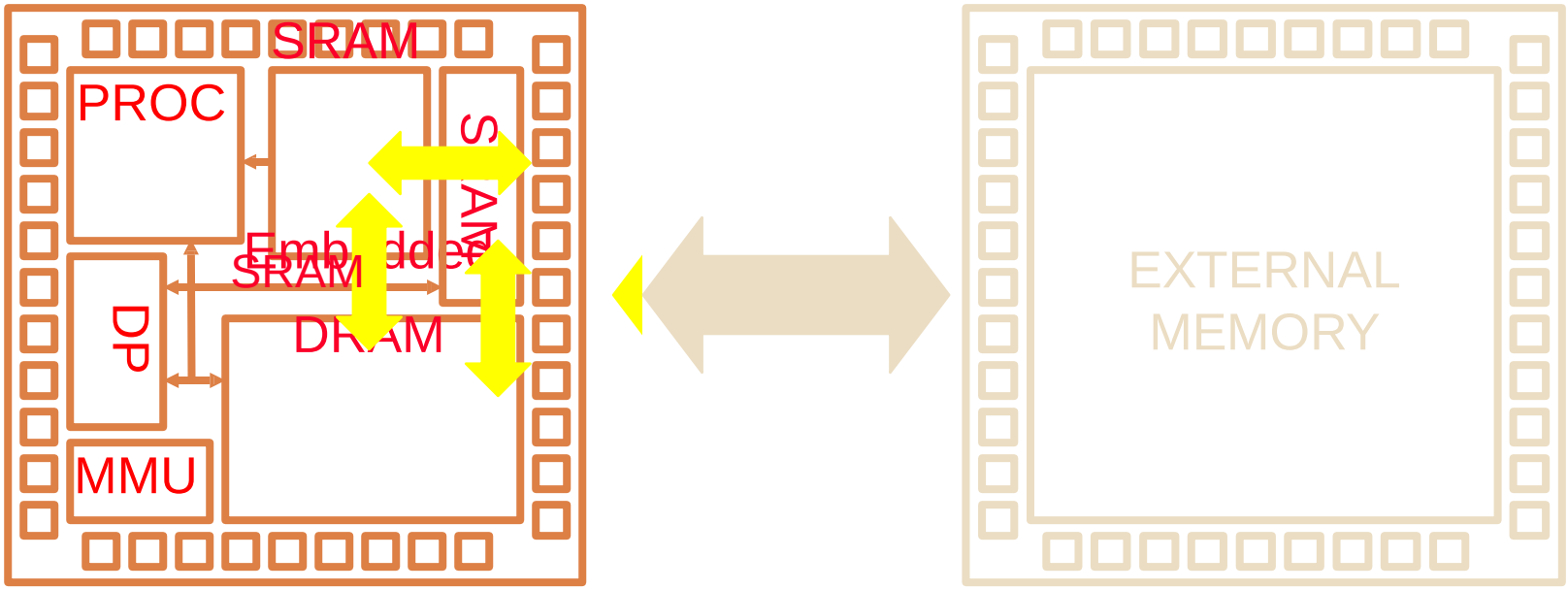
ΕΝΣΩΜΑΤΩΜΈΝΑ ΣΥΣΤΉΜΑΤΑ

Στατική Διαχείριση Μνήμης:
Αρχές και Εφαρμογές

Ορισμός Προβλήματος

- Οι εφαρμογές πολυμέσων (multimedia) (π.χ., mpeg-4, jpeg, h.264) χαρακτηρίζονται από μεγάλο αριθμό προσπελάσεων στην μνήμη
- Οι προσπελάσεις στην μνήμη είναι το μεγαλύτερο ποσοστό της συνολικής κατανάλωσης ισχύος (με άμεση επίδραση στην ταχύτητα εκτέλεσης)
- Για την λύση του προβλήματος αυτού απαιτείται η ανάπτυξη συστηματικής μεθοδολογίας

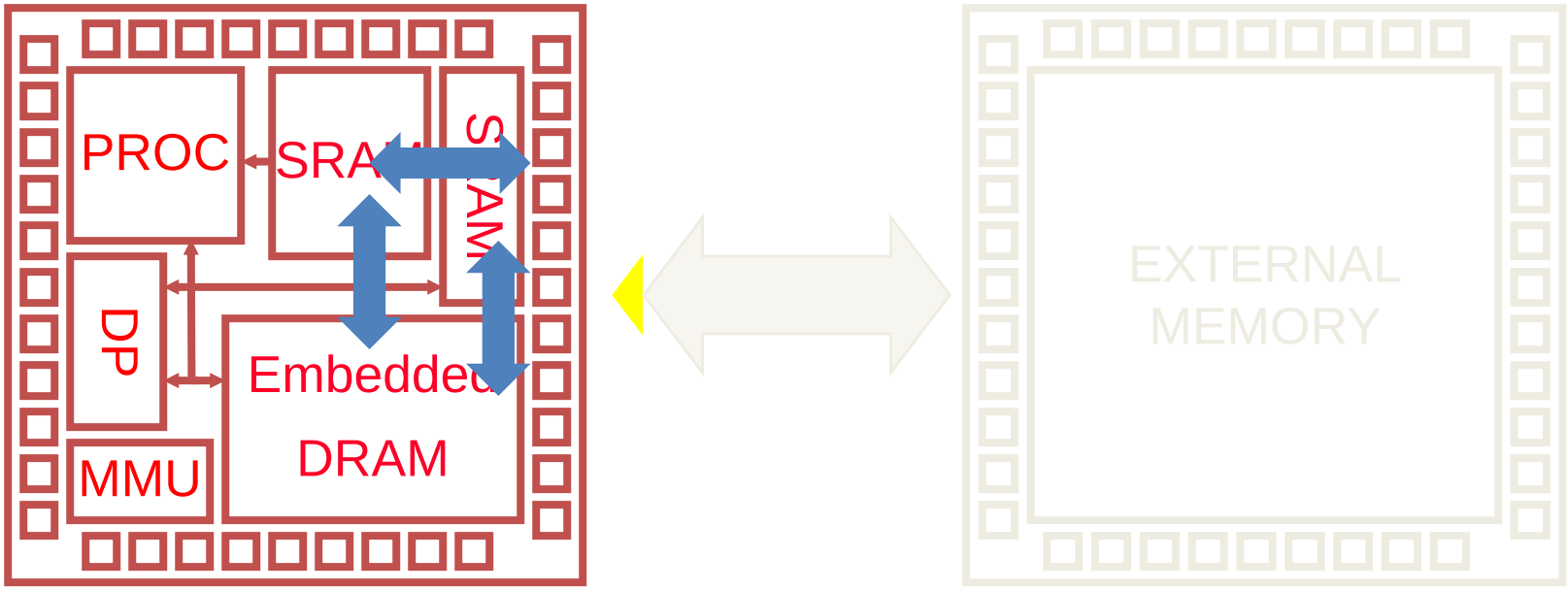
Embedded Systems



P(Ext. Access) = typ. 30 x P(Arithmetic Operations)

P(Int. Memory) = typ. 40 % - 60 % P(Chip)

Ενσωματωμένα Συστήματα (Embedded Systems)



P(Ext. Access) = typ. 30 x P(Arithmetic Operations)

P(Int. Memory) = typ. 40 % - 60 % P(Chip)

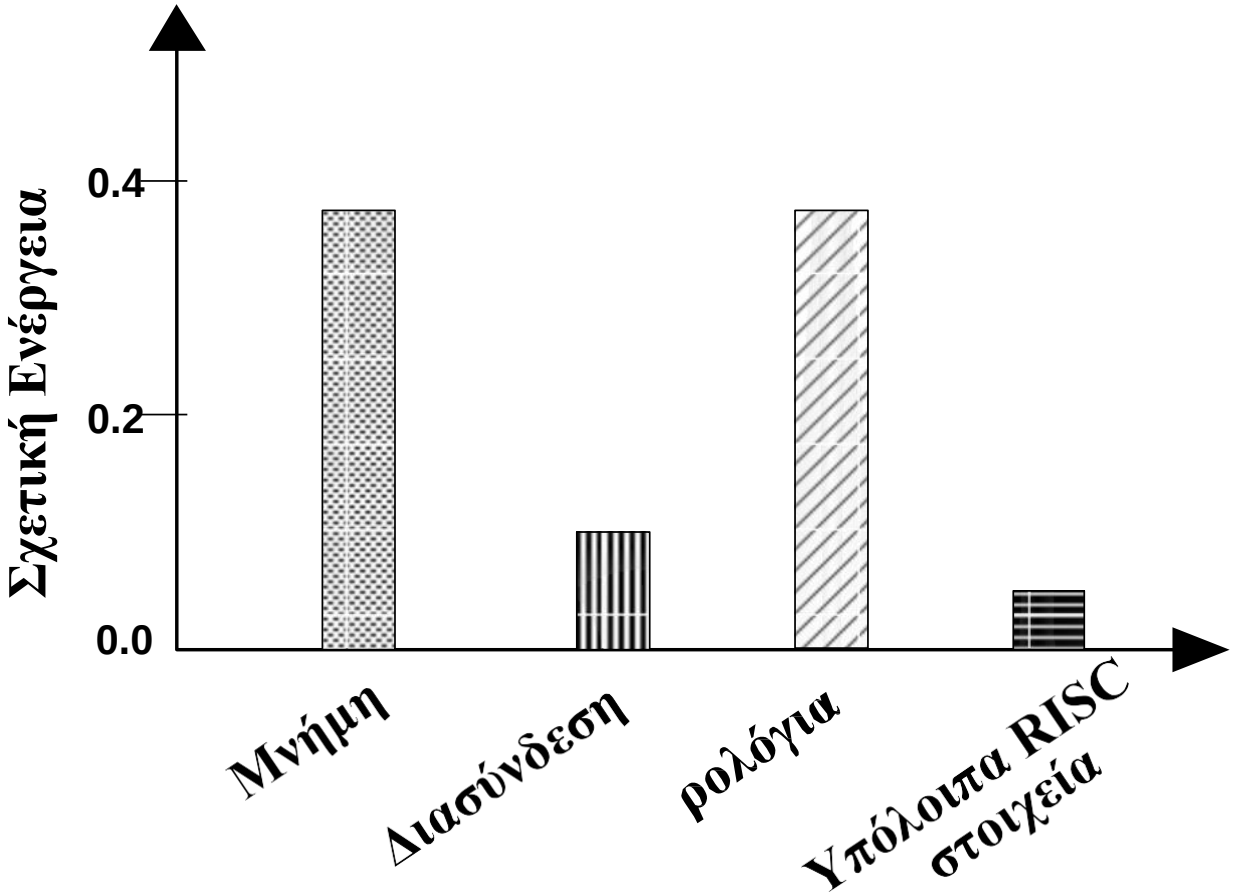
Κατανάλωση ενέργειας στην μνήμη για ενσωματωμένα συστήματα

- Ενα τυπικό ενσωματωμένο σύστημα πολυμέσων περιλαμβάνει δύο τύπους μνημών:
  - Μνήμη Δεδομένων (Data Memory)
  - Μνήμη Εντολών (Instruction Memory)

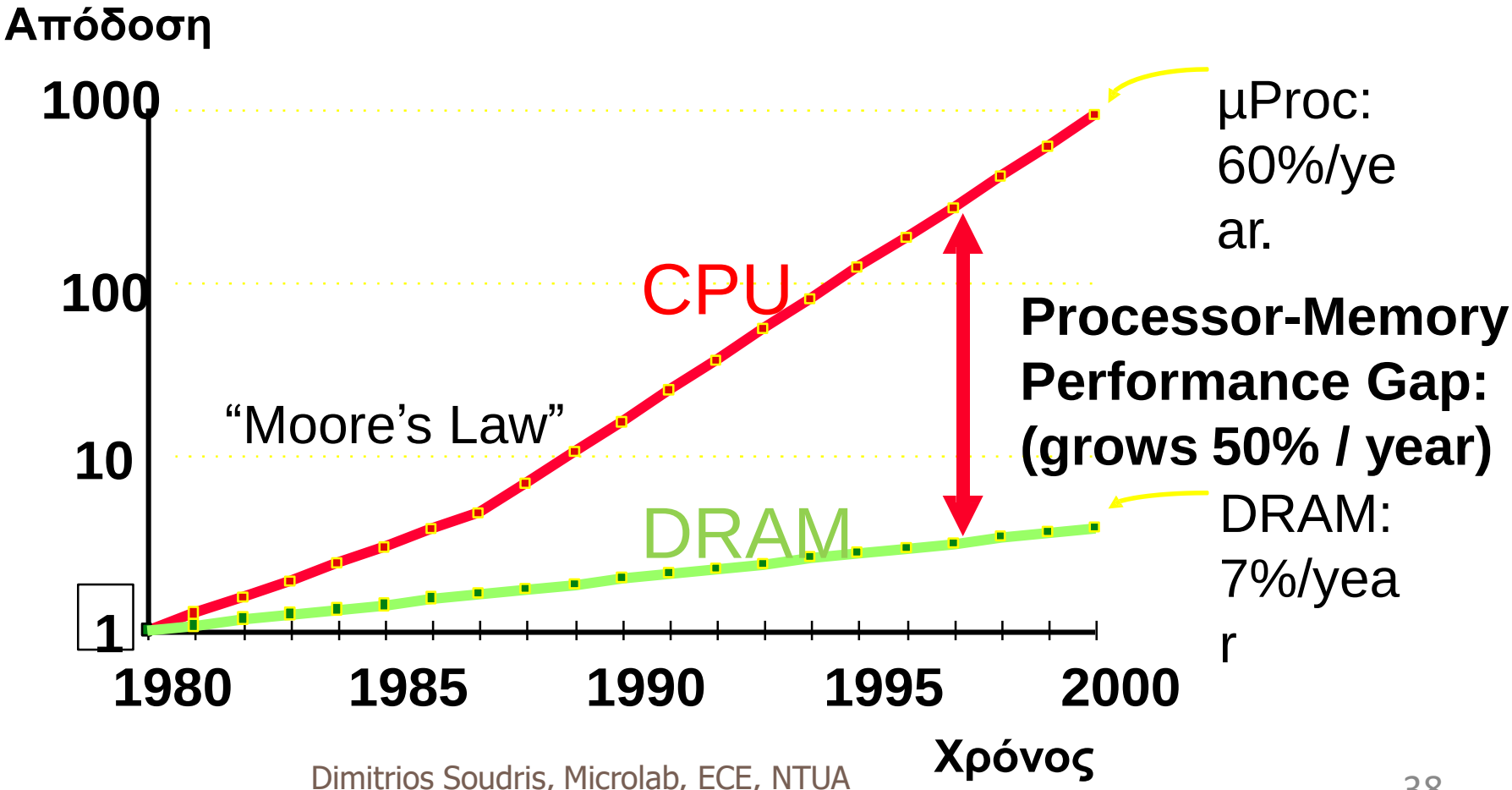
Στόχος: Ελαχιστοποίηση της συνολικής κατανάλωσης ισχύος, P_{total}, και των δύο συνιστωσών, P_{data} και $P_{instruction}$:

- $P_{total} = P_{data} + P_{instruction}$

Κατανομή Ενέργειας



Μνήμη= Το εμπόδιο στην απόδοση
Memory = Performance Bottleneck



Ενσωματωμένα Συστήματα Πολυμέσων

- Συστήματα Πολυμέσων είναι εφαρμογές πραγματικού χρόνου
- Παραδείγματα
  - Φωνή
  - Βίντεο (>30 fps)
  - Τηλεδιάσκεψη (10-15 fps)

Ανάλυση Βίντεο

30 frames /second

Παράδειγμα:

Ένα RGB video με 176 X 144 pixels έχει:

Μεταφορά Δεδομένων=

3*M*N*Bits*frame-rate =

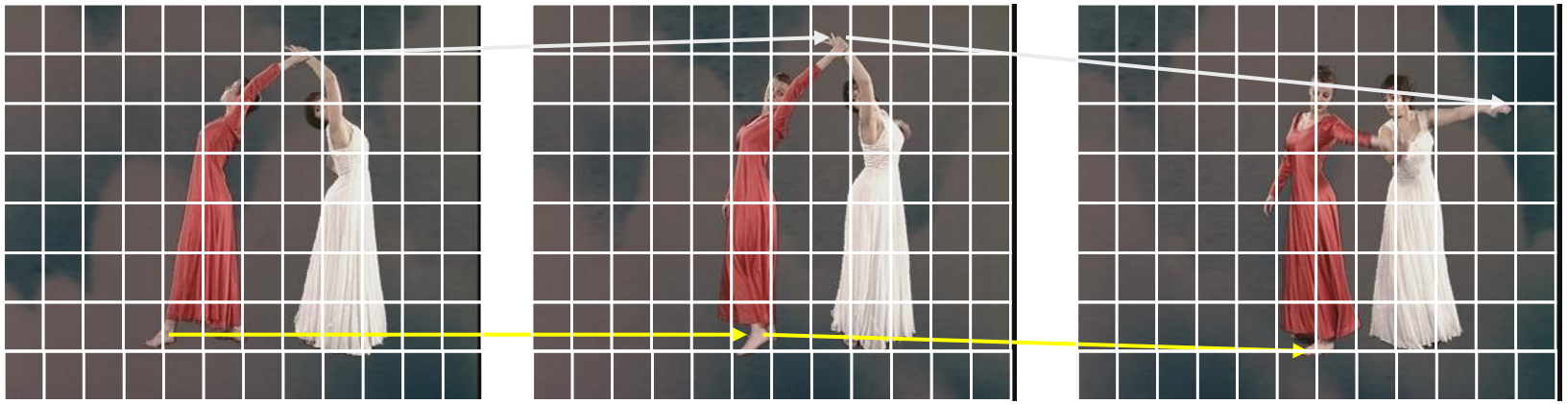
3*176*144*8*30 =

18.247.680 Bits/sec =

17.4 Mbits/sec

Τι μπορούμε να κάνουμε;

Τι είναι η Εκτίμηση Κίνησης (Motion Estimation);



Αλγόριθμοι Εκτίμηση Κίνησης (Motion Estimation Algorithms)

- Full Search (FS)
- Hierarchical Search (HS)
- 3 Step Logarithmic (3STEP)
- Parallel Hierarchical One Dimensional Search (PHODS)

Detailed info: https://link.springer.com/article/10.1023/A:1008192719838

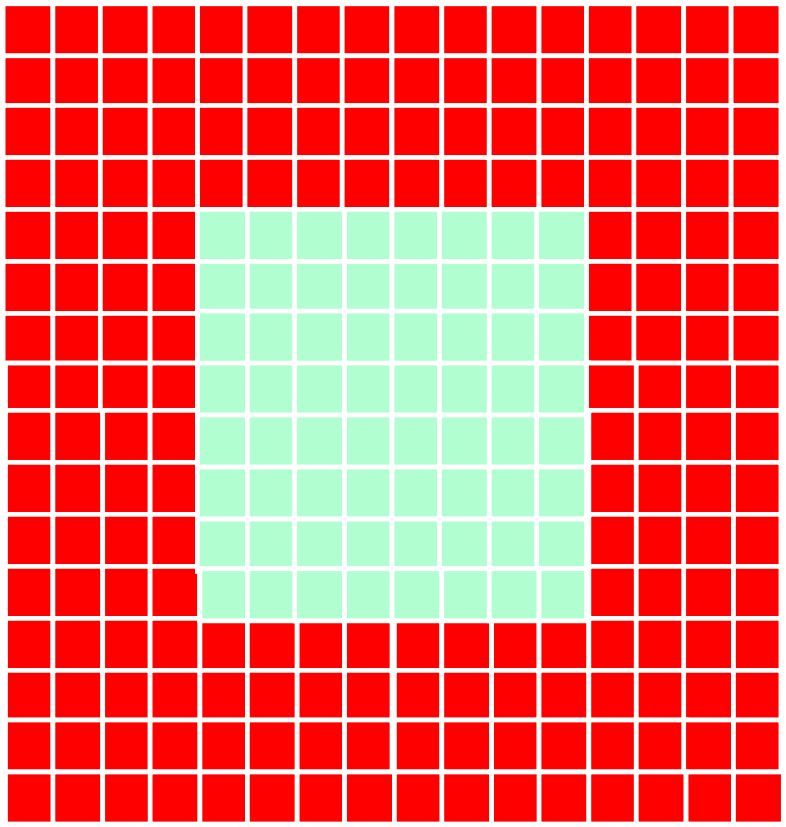
Υπολογιστική Πολυπλοκότητα

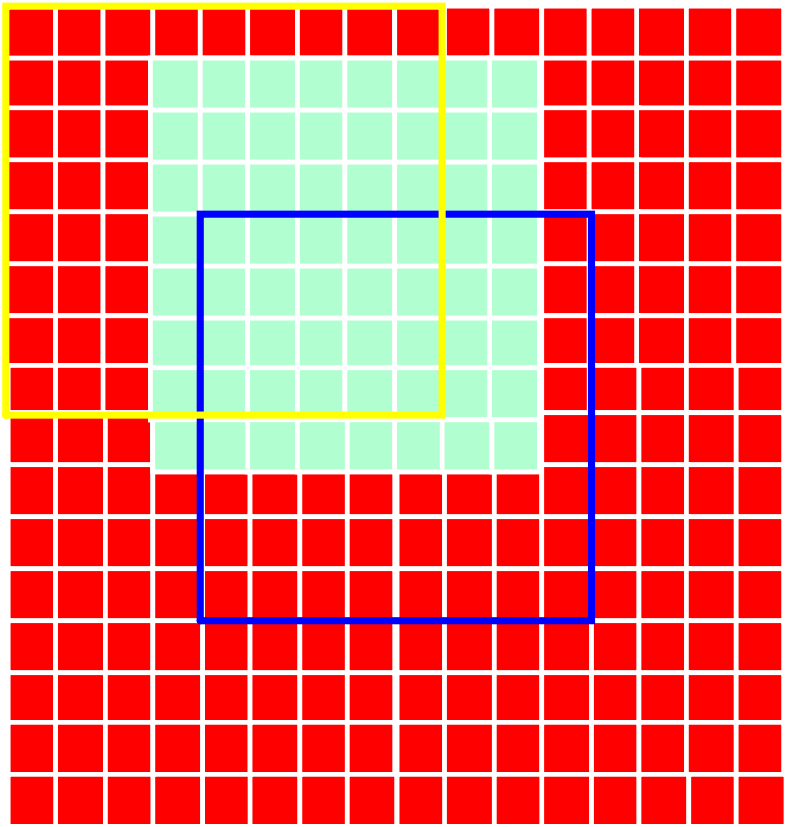
- Η υπολογιστική πολυπλοκότητα των αλγορίθμων Εκτίμηση Κίνησης είναι τεράστια
- Εξαρτάται από τον αλγόριθμο
- Παράδειγμα: το Full Search έχει πολυκλότητα 513,216 MOPS για εικόνες VGA (680X480)
- Πολύ μεγάλο αριθμό προσπελάσεων στην Μνήμη Δεδομένων (γύρω 300.000 προσπελάσεις ανά frame)

Full Search Algorithm


```
for(x = 0; x < N/B ; x++)    /* For all the rows */
 for(y = 0; y < M/B ; y++)  /*For all the columns */
   {
    for(i = -p; i < p+1; i++)  /* with i translocation on y */
     for(j = -p; j < p+1; j++) /* with j translocation on y */
      {
       for(k = 0; k < B; k++)   /* for all pixels of the block */
         for(l = 0; l < B; l++)  /* for all pixels of the block */
           {
             if((B*x+i+k) < 0 || (B*x+i+k) > N-1 ||
              (B*y+j+l)<0 ||       (B*y+j+l)>M-1)
             /*calculations*/
           }
       }
    }
```

Matching…





Block matching

Previous Frame **Current Frame**

$$MAE(dx,dy) = \sum_{m=x}^{x+N-1} \sum_{n=y}^{y+N-1} \left| I_k(m,n) - I_{k-1}(m+dx, n+dy) \right|$$

$$(MV_x, MV_y) = \min_{(dx,dy)\in R^2} MAE(dx,dy)$$

Μετασχηματισμοί Ροής Δεδομένων

Αρχικός Αλγόριθμος

```
for(x = 0; x < N/B ; x++)
 for(y = 0; y < M/B ; y++)
  {
   while(S>0)
    {
     for(i=-S;i<S+1;i+=S)
      {
       for(k = 0; k < B; k++)
        for(l = 0; l < B; l++)
         {
          calculations on X dimension
         }
      }
     for(i=-S;i<S+1;i+=S)
      {
       for(k = 0; k < B; k++)
        for(l = 0; l < B; l++)
         {
          calculations on Y dimension
         } }
     S=S/2
  }}
```

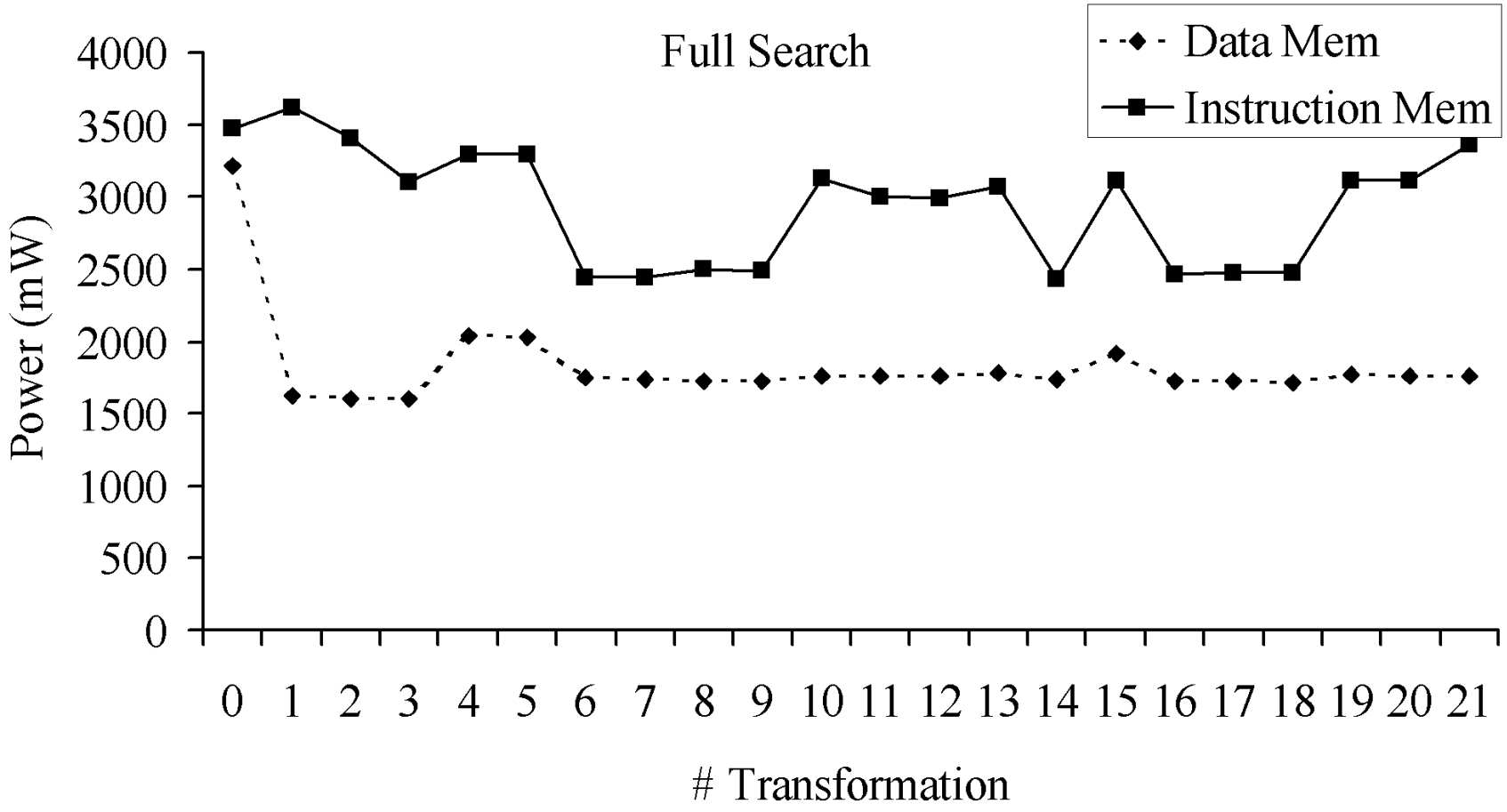
Βελτιστποίηση

```
for(x = 0; x < N/B ; x++)
 for(y = 0; y < M/B ; y++)
  {
   while(S>0)
    {
     for(i=-S;i<S+1;i+=S)
     {
      for(k = 0; k < B; k++)
       for(l = 0; l < B; l++)
        {
         calculations on X & Y dimensions
        }
     }
    S=S/2
    }

  }
```

Αλγόριθμος Εκτίμησης Κίνησης

Συνολική Κατανάλωση Ισχύος στο Full Search



Μετασχηματισμοί Επαναχρησιμοποίησης Δεδομένων
Data Reuse Transformations

- Επαναχρησιμοποίηση των δεδομένων στα διάφορα επίπεδα ιεραρχίας μνημών
- Δεν χρειάζεται να διαβάζονται όλα τα δεομένα από τα ψηλότερα επίπεδα μνήμης
- Λιγότερες συνολικά προσπελάσεις στην μνήμη

Τελικά συμπεράσματα

- Σωστός συνδυασμός ιεραρχίας μνημών δεδομένων και μετασχηματισμών επαναχρησιμοποίησης δεδομένων απαιτείται για να προκύψει μια καλή λύση
- Οι μετασχηματισμοί ροής απαιτούνται σε εφαρμογές πολυμέσων, καθώς και εφαρμογές πραγματικού χρόνου
- Η κατανάλωση ισχύος της μνήμης εντολών είναι μεγάλο τμήμα της συνολικής κατανάλωσης ισχύος
- Ο αριθμός επεξεργαστών επιδρά σημαντικά στην ισχύ, την απόδοση και το εμβαδόν πυριτίου

Μονάδες υλικού, Ιεραρχία Μνήμης

Μονάδες Μνήμης

Μονάδα	Ακριβές πλήθος byte	Προσέγγιση
Kilobyte (KB)	2^{10} (1.024) byte	10^3 byte
megabyte (MB)	2^{20} (1.048.576) byte	10^6 byte
gigabyte (GB)	2^{30} (1.073.741.824) byte	10^9 byte
Terabyte (TB)	2^{40} byte	10^{12} byte
Petabyte (PB)	2^{50} byte	10^{15} byte
Exabyte (EB)	2^{60} byte	10^{18} byte

Ταξινόμηση Ημιαγωγικών Μνημών

Read-Write Memory		Non-Volatile Read-Write Memory	Read-Only Memory
Random Access	Non-Random Access	EPROM E^2PROM FLASH	Mask-Programmed Programmable (PROM)
SRAM DRAM	FIFO LIFO Shift Register CAM		

Τύποι Μνήμης

- RAM **μνήμη τυχαίας προσπέλασης** (Random Access Memory)
  - μπορεί να αναγνωστεί και να εγγραφεί από το χρήστη
  - είναι "πτητική", όταν διακόπτεται η τροφοδοσία του ρεύματος, οι πληροφορίες (πρόγραμμα ή δεδομένα) διαγράφονται
- ROM **μνήμη μόνο για ανάγνωση** (Read-Only Memory)
  - ο χρήστης μπορεί να διαβάσει τη ROM αλλά όχι και να γράψει σε αυτή
  - είναι μη πτητική
  - χρησιμοποιείται για προγράμματα ή δεδομένα που δεν πρέπει να διαγραφούν ή να μεταβληθούν ακόμα και όταν ο υπολογιστής κλείνει

Μνήμες Ανάγνωσης-Εγγραφής (RAM)

- **Στατική RAM (Static RAM - SRAM)**
  - Τα δεδομένα αποθηκεύονται όσο υπάρχει τάση
  - Το κελί μνήμης έχει 6 τρανζίστορ
  - Γρήγορη
  - Εφαρμογές: π.χ., Κρυφές Μνήμες (cache memory), μνήμες εντός ολοκληρωμένου

- **Δυναμική RAM (Dynamic RAM - DRAM)**
  - Απαιτεί περιοδικό φρεσκάρισμα περιεχομένου
  - Μικρό μέγεθος (1-3 τρανζίστορ ανά κελί)
  - Αργή μνήμη
  - Εφαρμογές: π.χ., μνήμες εκτός ολοκληρωμένου

Τύποι Μνήμης RAM

- SRAM **στατική RAM** (Static RAM)
  - χρησιμοποιεί για την αποθήκευση δεδομένων τις παραδοσιακές πύλες φλιπ-φλοπ
  - διατηρεί την κατάστασή της (0 ή 1), δηλαδή τα δεδομένα διατηρούνται αποθηκευμένα όσο υπάρχει τροφοδοσία ρεύματος χωρίς να χρειάζονται ανανέωση.
  - είναι γρήγορη αλλά ακριβή

Τύποι Μνήμης RAM

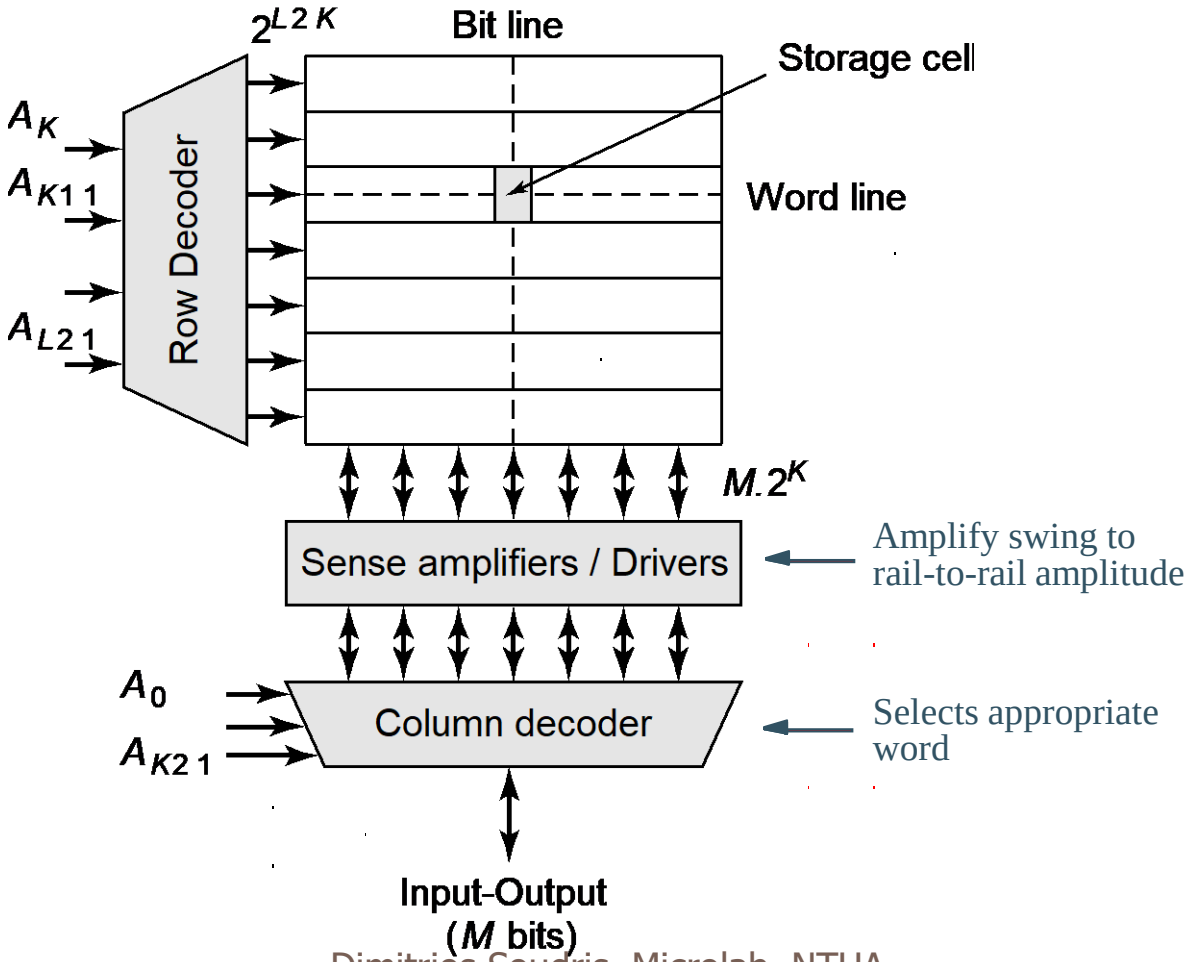
- DRAM **δυναμική RAM** (Dynamic RAM)
  - χρησιμοποιεί πυκνωτές
  - Αν ο πυκνωτής είναι φορτισμένος, η κατάσταση είναι 1, αν είναι αφόρτιστος, η κατάσταση είναι 0.
  - Επειδή οι πυκνωτές χάνουν ένα μέρος του φορτίου τους με την πάροδο του χρόνου, οι θέσεις τα κελιά μνήμης χρειάζονται περιοδική ανανέωση.
  - είναι αργές αλλά φτηνές.

Τύποι Μνήμης ROM

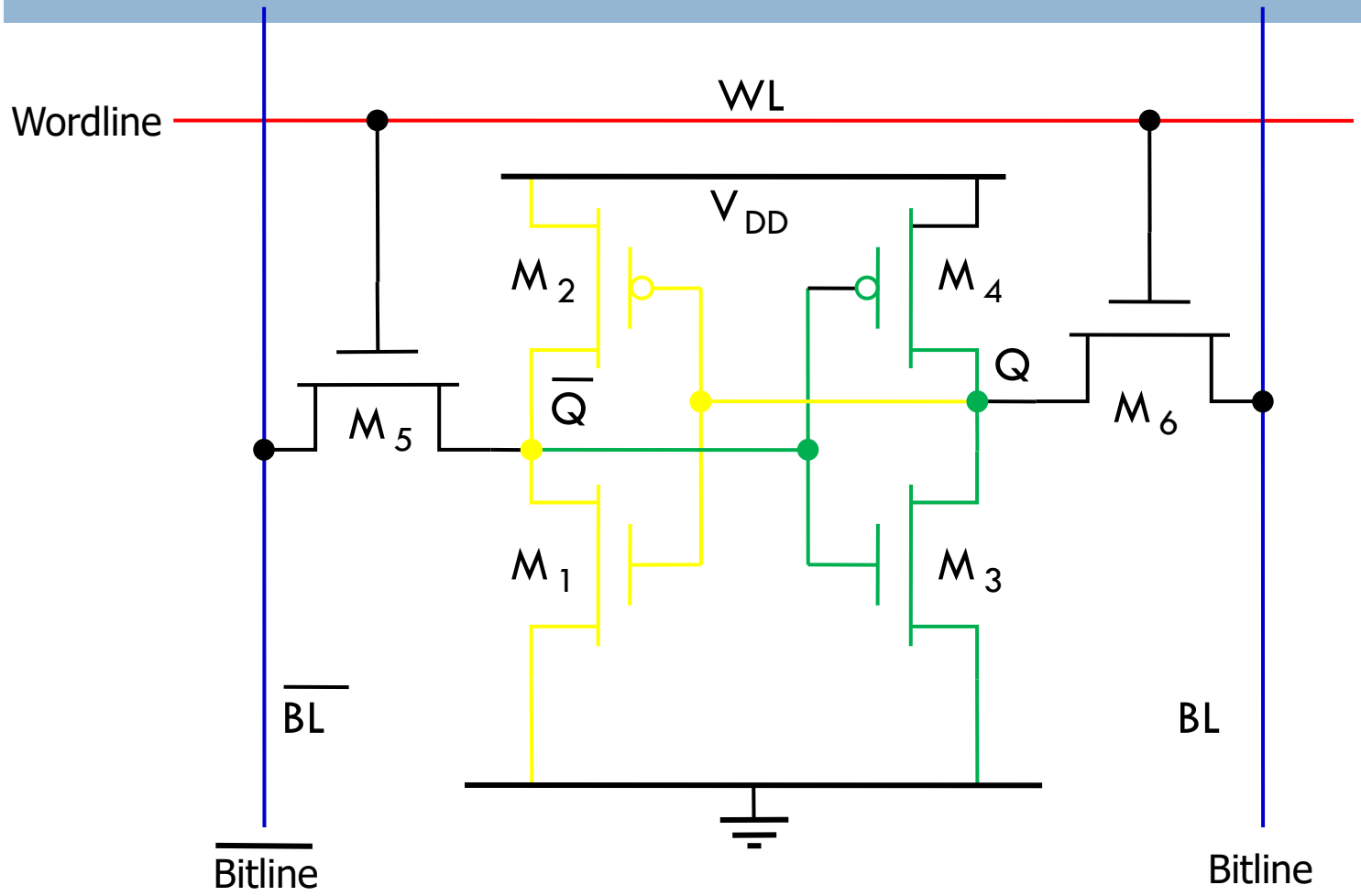
- Η **ηλεκτρονικά διαγράψιμη προγραμματιζόμενη μνήμη μόνο για ανάγνωση** (Electronically Erasable Programmable Read-Only Memory, ή **EEPROM**) αποτελεί μια παραλλαγή της EPROM.
  - Μπορεί να προγραμματιστεί και να διαγραφεί μέσω ηλεκτρονικών παλμών χωρίς να απαιτείται η αφαίρεσή της από τον υπολογιστή

Αρχιτεκτονική Μνήμης

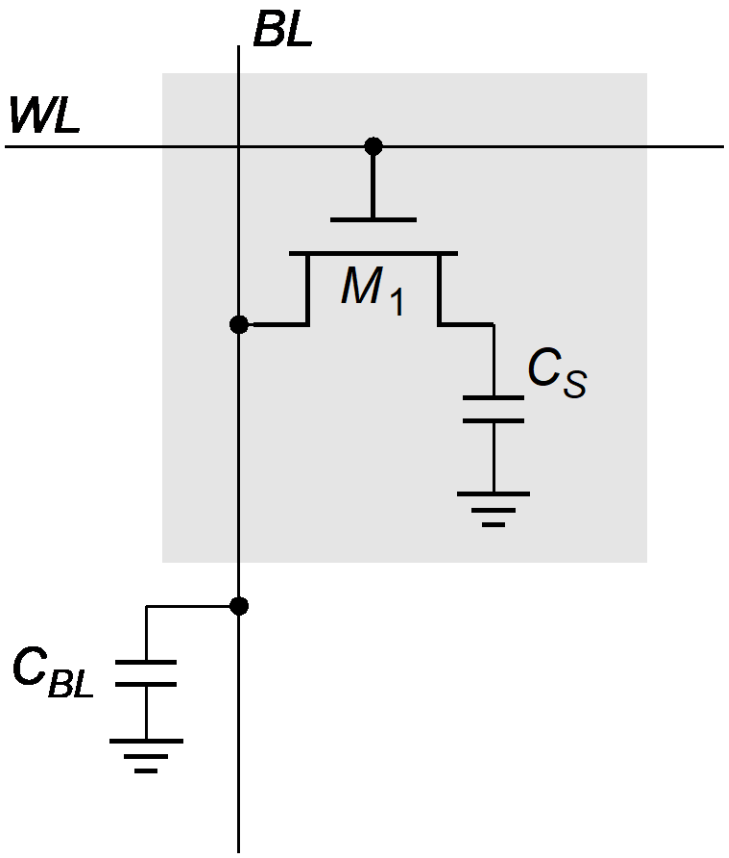
Problem: ASPECT RATIO or HEIGHT >> WIDTH

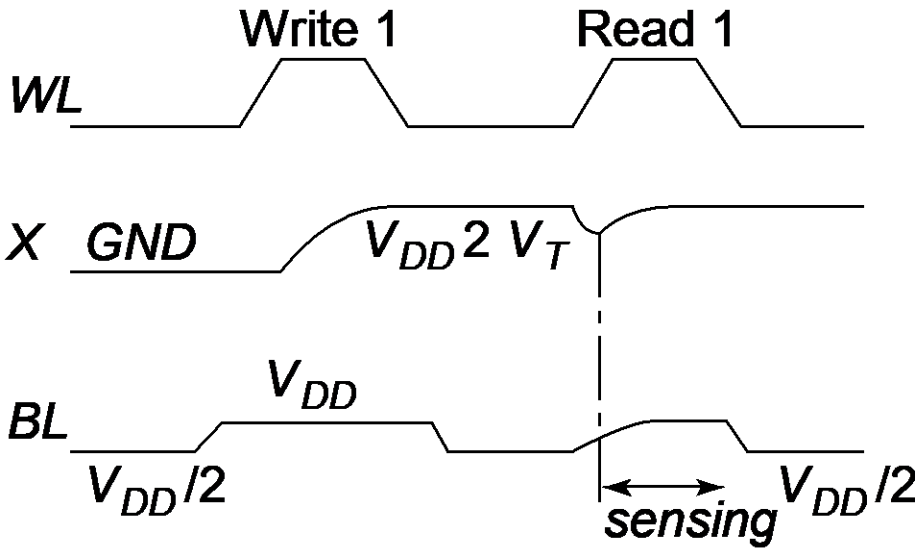


Κύτταρο SRAM με 6-τρανζίστορ



DRAM με ένα τρανζίστορ

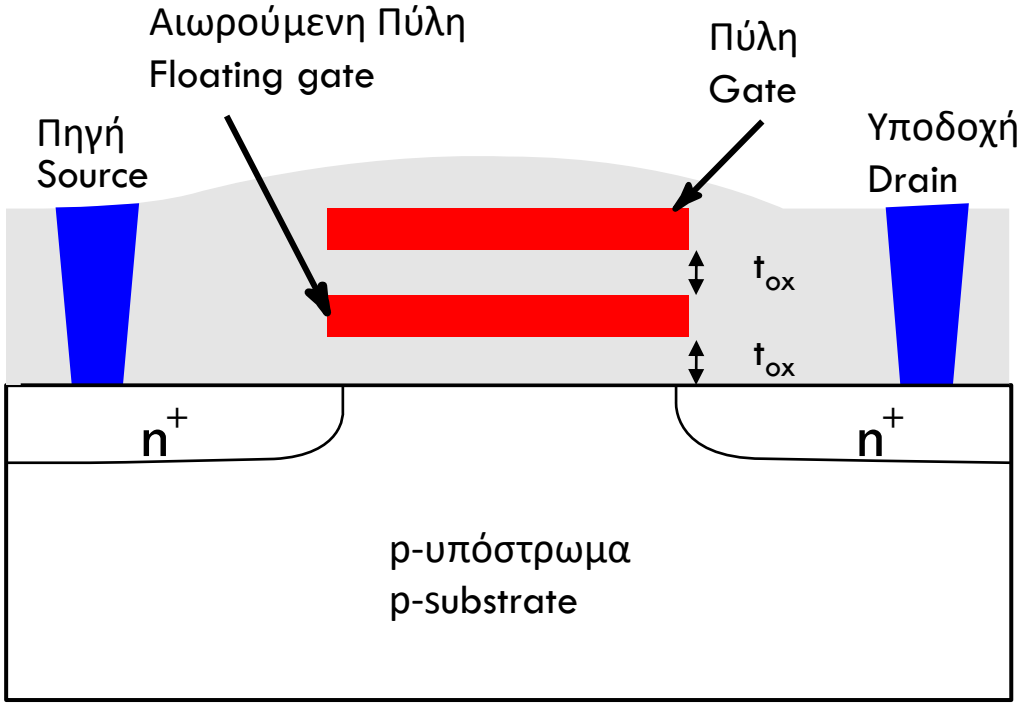




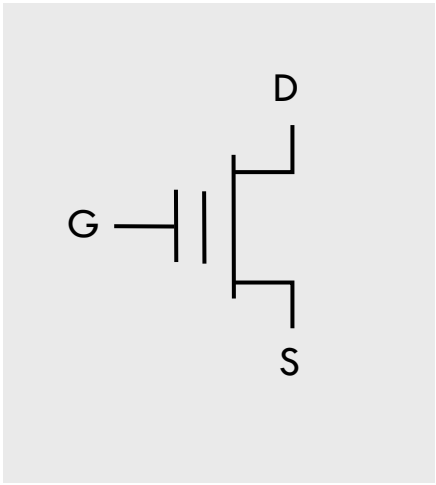
Μη-Πτητικές Μνήμες
Non-Volatile Memories

- Το περιεχόμενο της μνήμης παραμένει και χωρίς τάση τροφοδοσίας
- «Προγραμματιζόμενη» μνήμη
- Παράδειγμα: Flash memories, EEPROM

Μη-Πτητικές Μνήμες (Non-Volatile Memories)
Τρανζίστορ Αιωρούμενης Πύλης
(Floating-gate transistor [FAMOS])

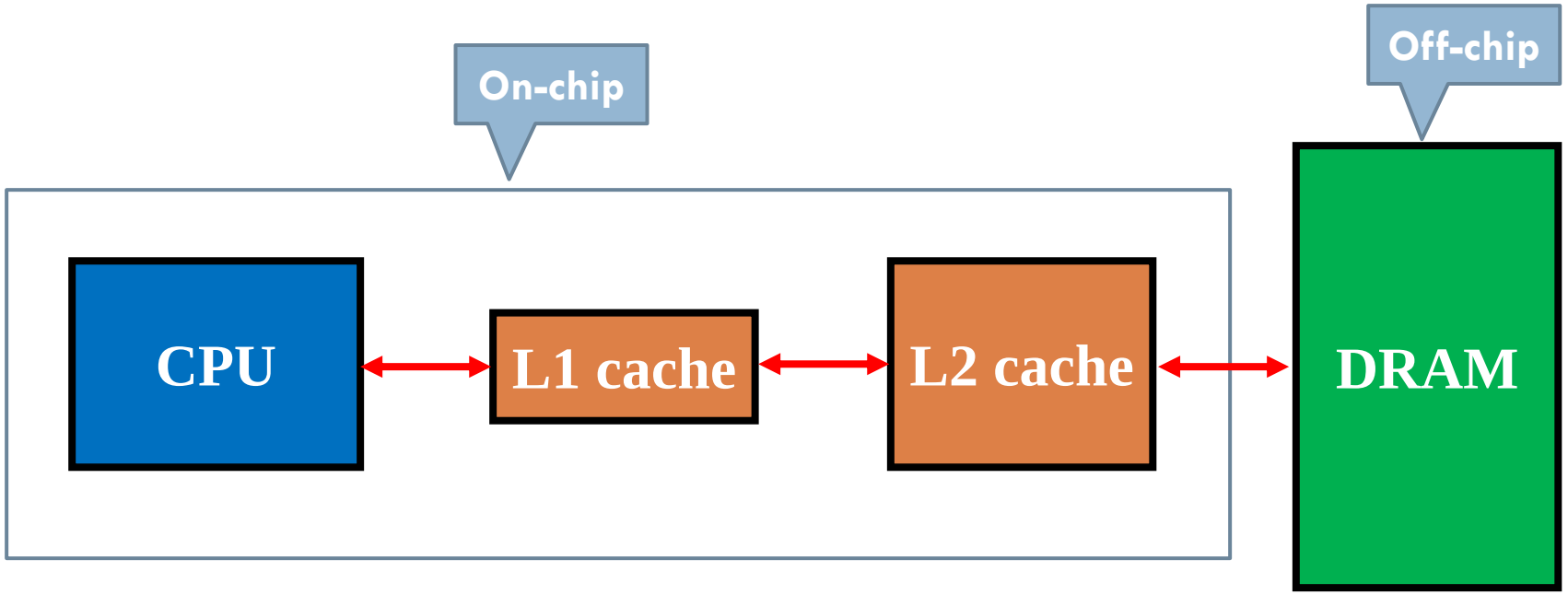


Διατομή Στοιχείου
Device cross-section



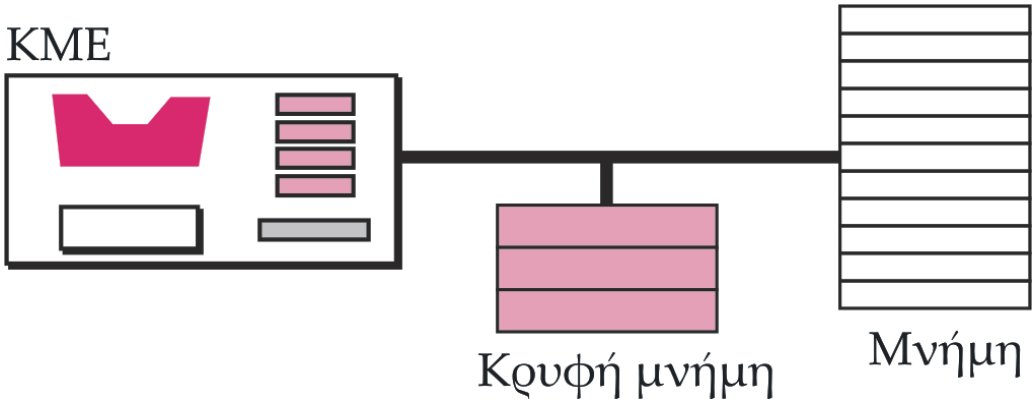
Σύμβολο Σχηματικό

Πολλαπλά Επίπεδα Μνημών



Κρυφή Μνήμη

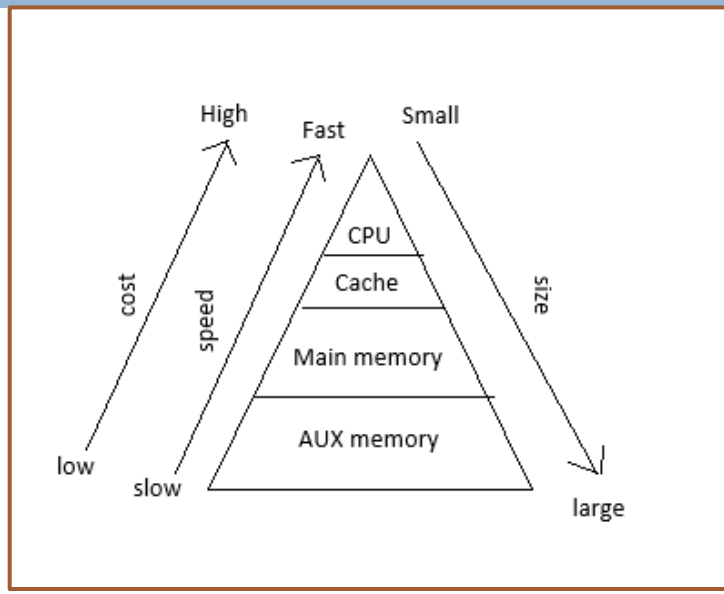
- Είναι γρηγορότερη από την κύρια μνήμη αλλά πιο αργή από την ΚΜΕ και τους καταχωρητές της.
- Η κρυφή μνήμη, η οποία συνήθως έχει μικρό μέγεθος, μεσολαβεί μεταξύ της ΚΜΕ και της κύριας μνήμης

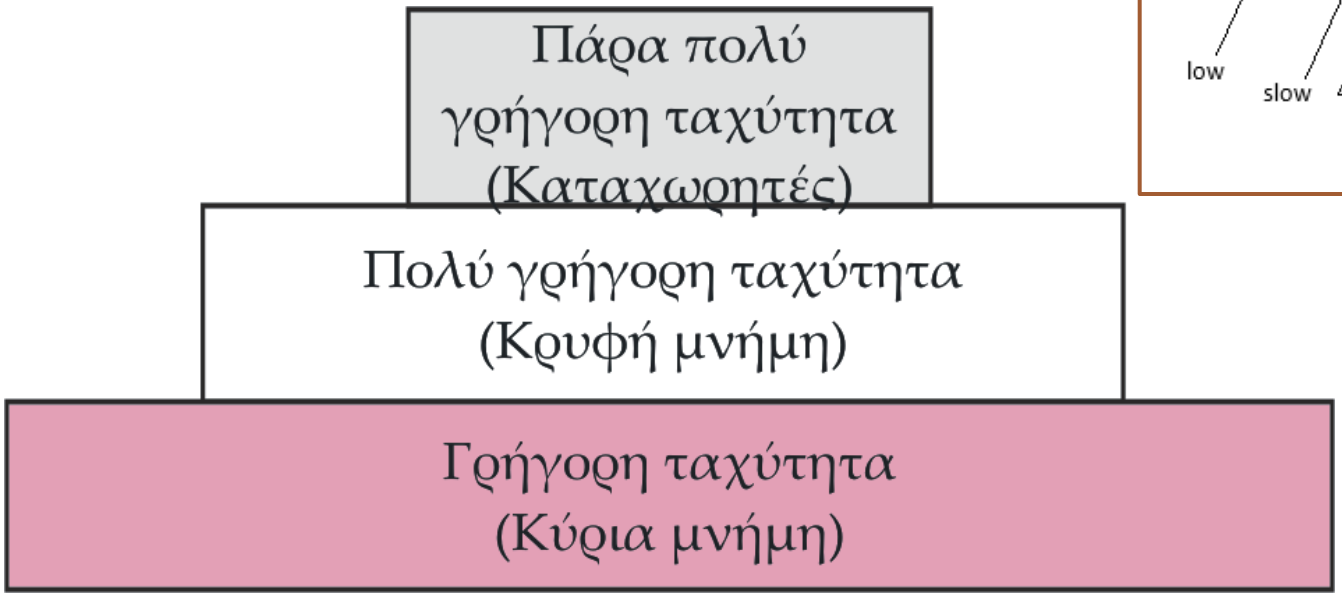


Κρυφή Μνήμη

- Η κρυφή μνήμη περιέχει συνεχώς ένα αντίγραφο κάποιου τμήματος της κύριας μνήμης. Όταν η ΚΜΕ πρέπει να προσπελάσει μια λέξη στην κύρια μνήμη, ακολουθείται η εξής διαδικασία:
  1. Η ΚΜΕ ελέγχει την κρυφή μνήμη.
  2. Αν βρει εκεί τη λέξη, την αντιγράφει, αν όχι, η ΚΜΕ προσπελάζει την κεντρική μνήμη και αντιγράφει το τμήμα της το οποίο ξεκινάει με την επιθυμητή λέξη. Το τμήμα αντικαθιστά τα προηγούμενα περιεχόμενα της κρυφής μνήμης.
  3. Η ΚΜΕ προσπελάζει την κρυφή μνήμη και αντιγράφει τη λέξη.

Ιεραρχία της μνήμης





Λειτουργία Κρυφής Μνήμης

- **Πολλές θέσεις της κύριας μνήμης απεικονίζονται σε κρυφή μνήμη**
- **Είδη κρυφών μνημών:**
  - **μικροεντολές**
  - **δεδομένα;**
  - **δεδομένα + μικροεντολές (unified)**
- **Η ταχύτητα πρόσβασης δεν είναι καθορισμένη**

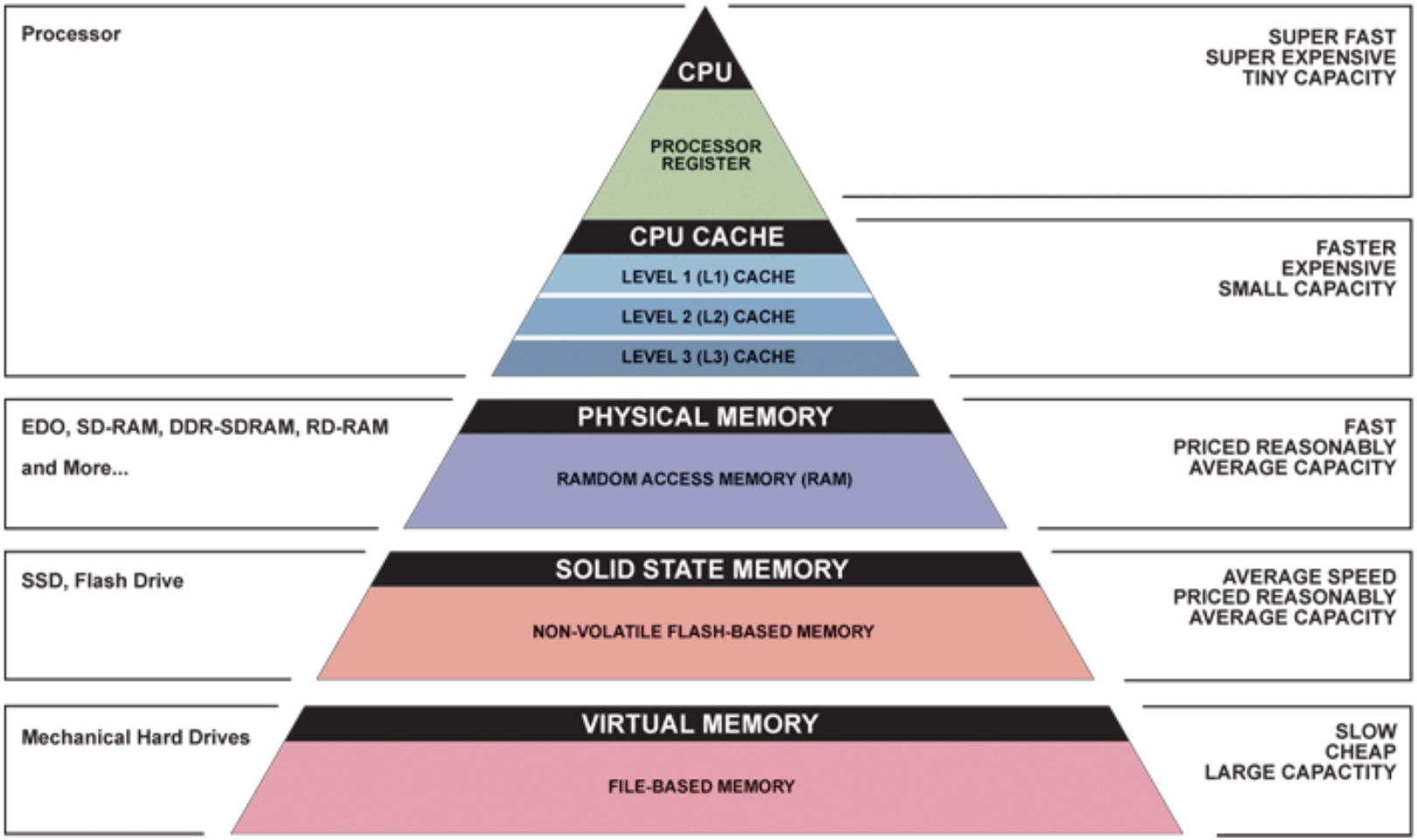
Χαρακτηριστικά Κρυφών Μνημών

- Ευστοχία κρυφής μνήμης (Cache hit)
  - το απαιτούμενο δεδομένο είναι κρυφή μνήμη
- Αστοχία κρυφής μνήμης (Cache miss)
  - το απαιτούμενο δεδομένο ΔΕΝ είναι κρυφή μνήμη
- Σύνολο εργασίας (Working set)
  - Σύνολο θέσεων χρησιμοποιούμενο από το πρόγραμμα σε ένα χρονικό διάστημα
- Είδη αστοχιών
  - Υποχρεωτική (Compulsory): location has never been accessed.
  - Χωρητικότητας (Capacity): working set is too large.
  - Conflict: multiple locations in working set map to same cache entry.

Στρατηγική προσωρινής αποθήκευσης

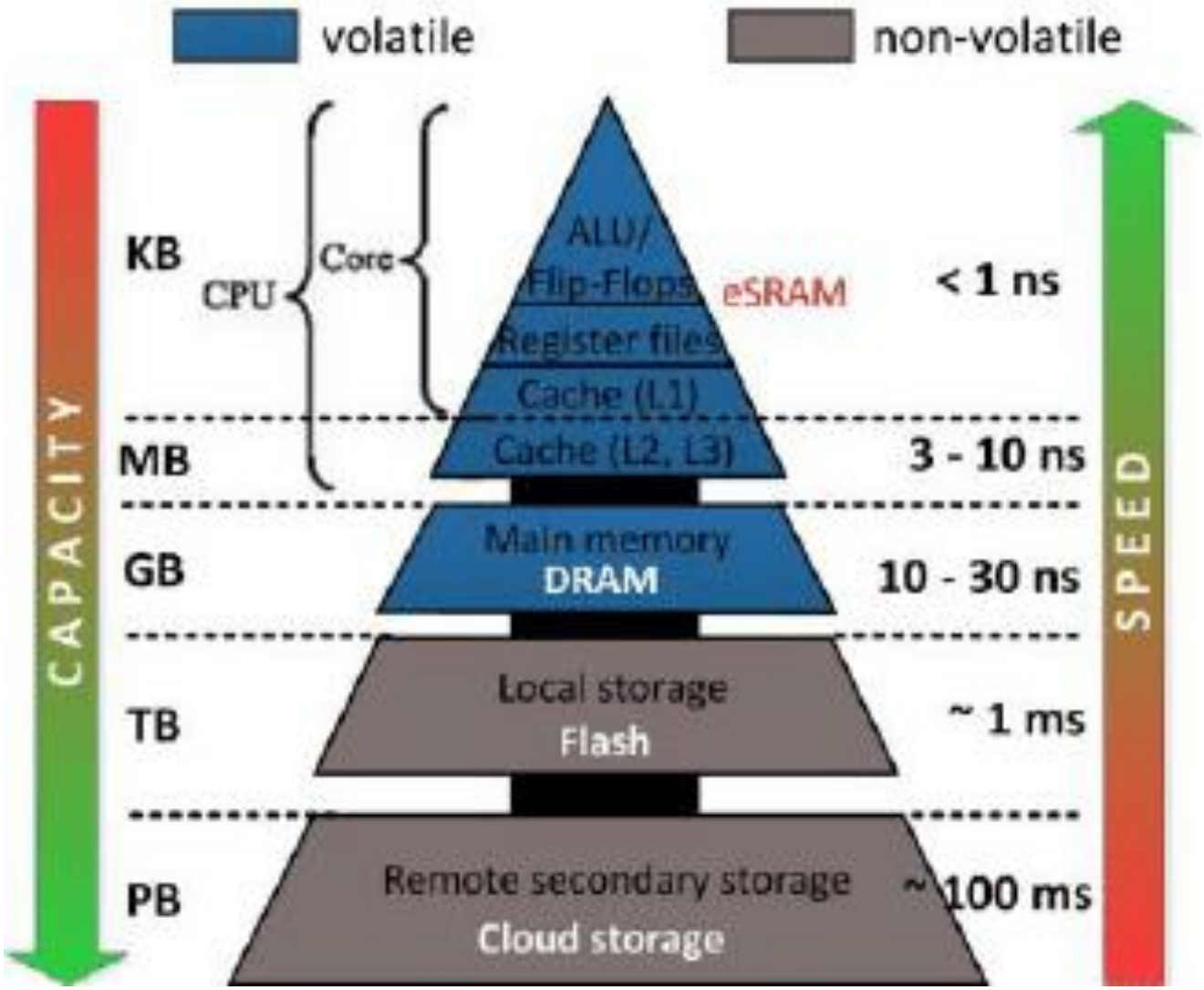
- Διατηρήστε τα δημοφιλή δεδομένα σε ακριβή, μικρή και γρήγορη μνήμη.
- Κρατήστε τα λιγότερο δημοφιλή στοιχεία στη φτηνή, μεγάλη, και αργή μνήμη.

Ιεραρχία μνημών



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

Πτητικές VS. Μη-πτητικές



Memory Hierarchy

We are therefore forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible.

A. W. Burks, H. H. Goldstine, J. von Neumann: *Preliminary Discussion of the Logical Design of Electronic Computing Instrument, Part I, Vol. I, Report prepared for the U.S. Army Ord. Dept.*
28 June 1946