**Κεφάλαιο**


# Οργάνωση Μνήμης

*Σκοπός του κεφαλαίου αυτού είναι να περιγράψει τον τρόπο που διαχειρίζονται οι υπολογιστές τη μνήμη που περιέχουν, ώστε αυτή να κατανέμεται δίκαια και με ασφάλεια μεταξύ των προγραμμάτων, και ο υπολογιστής να λειτουργεί όσο πιο αποδοτικά είναι δυνατό.*

Όταν ολοκληρώσεις το κεφάλαιο αυτό, θα μπορείς:

- Να περιγράφεις τα τμήματα του συστήματος μνήμης και τα χαρακτηριστικά τους.
- Να εξηγείς τον τρόπο ανάγνωσης και εγγραφής στην κύρια μνήμη.
- Να περιγράφεις τα επίπεδα ιεραρχίας της μνήμης και τα χαρακτηριστικά τους.
- Να καταρτίσεις την κύρια μνήμη για ένα ορισμένο υπολογιστή και να χρησιμοποιείς την τεχνική της διαφύλλωσης.
- Να περιγράφεις τον τρόπο λειτουργίας της λανθάνουσας μνήμης και να υπολογίζεις την επιτάχυνση που προσφέρει στα προγράμματα.

**Μαθήματα**

# Μάθημα 4.1 Τεχνολογία και Χαρακτηριστικά Μνημών

**Σκοπός του μαθήματος**

*Σκοπός του μαθήματος αυτού είναι να παρουσιάσει τα κύρια χαρακτηριστικά του συστήματος μνήμης για έναν υπολογιστή και τις βασικές λειτουργίες της κύριας μνήμης.*

**Τι θα μάθεις;**

*Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:*

- *Να περιγράφεις τα βασικά χαρακτηριστικά των μνημών*
- *Να απαριθμείς τους βασικούς στόχους των σχεδιαστών ενός συστήματος μνήμης*
- *Να περιγράφεις τη διαδικασία ανάγνωσης και εγγραφής της κύριας μνήμης*
- *Να προσδιορίζεις το χώρο διευθύνσεων της μνήμης*

Το *υποσύστημα της μνήμης* σε έναν υπολογιστή περιλαμβάνει όλες εκείνες τις μονάδες που έχουν τη δυνατότητα να αποθηκεύουν δυαδικές πληροφορίες και να επιτρέπουν την αξιόπιστη ανάκτησή τους οποτεδήποτε αυτό ζητηθεί από τον επεξεργαστή.

Η αποθήκευση των πληροφοριών σε μία μονάδα μνήμης γίνεται με τη λειτουργία της *εγγραφής* (write) στη μνήμη και η ανάκτησή τους γίνεται με τη λειτουργία της *ανάγνωσης* (read) από τη μνήμη. Και οι δύο λειτουργίες ενεργοποιούνται μέσω των αντίστοιχων εντολών του επεξεργαστή.

Οι λειτουργίες της εγγραφής και της ανάγνωσης σε κάθε μονάδα μνήμης, για να λειτουργεί η μονάδα ικανοποιητικά, πρέπει να εκτελούνται:

(1) **Αξιόπιστα,** δηλαδή χωρίς σφάλματα. Όταν εκτελείται μια λειτουργία ανάγνωσης στη μνήμη, τα δεδομένα που θα διαβαστούν πρέπει να είναι *ακριβώς* αυτά που είχαν εγγραφεί την τελευταία φορά σε αυτή τη θέση.

(2) **Ταχύτατα**, όσο βέβαια επιτρέπουν οι φυσικοί περιορισμοί της μονάδας μνήμης.

(3) **Με το ελάχιστο δυνατό κόστος**, ώστε να είναι εφικτή η αποθήκευση μεγάλου όγκου δεδομένων χωρίς υπέρμετρη οικονομική επιβάρυνση.

Οι πληροφορίες που είναι αποθηκευμένες σε μια μονάδα μνήμης είναι συνήθως οργανωμένες σε ομάδες δυαδικών ψηφίων. Κάθε ομάδα χαρακτηρίζεται από τα *περιεχόμενά* της και από τη *διεύθυνσή* της. Κάθε προσπέλαση (ανάγνωση ή εγγραφή) στη μνήμη γίνεται για τη μεταφορά μιας ενιαίας ομάδας πληροφοριών, με βάση τη διεύθυνση τής ομάδας. Οι συνηθέστεροι τύποι ομάδων είναι η λέξη και η ενότητα.

Η *λέξη* (word) συνήθως αποτελείται από 1-8 bytes (δηλαδή, 8-64 bits) και χρησιμοποιείται για την προσπέλαση στη μονάδα της κύριας μνήμης του υπολογιστή.

Η *ενότητα* (block ή frame), η οποία συνήθως περιλαμβάνει έναν μεγάλο αριθμό από λέξεις, χρησιμοποιείται συχνά ως βασική μονάδα ανταλλαγής πληροφορίας με τις περιφερειακές συσκευές. Έτσι η ανάγνωση ή η εγγραφή στο σκληρό δίσκο δεν μπορεί να γίνει για μία μόνο λέξη, αλλά για ολόκληρη την ενότητα στην οποία αυτή περιλαμβάνεται.

| Μονάδα | Κόστος (δρχ/bit) | Ταχύτητα | Χωρητικότητα (Mbytes) |
|---|---|---|---|
| Η μνήμη ενός προσωπικού υπολογιστή το 1980 | 0,1 | 120 ns | 0,2 |
| Η μνήμη ενός προσωπικού υπολογιστή το 1999 | $10^{-4}$ | 60 ns | 64 |
| Ένας σκληρός δίσκος για εξυπηρετητή (server) το 1999 | $10^{-5}$ | 10 ms | 10000 |

**Μερικά παραδείγματα μονάδων μνήμης, παλιών και νεότερων, και των χαρακτηριστικών τους.**

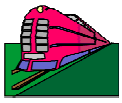
| | | |
|---|---|---|
| **K** | **Kilo** | $2^{10}$ |
| **M** | **Mega** | $2^{20}$ |
| **G** | **Giga** | $2^{30}$ |
| **T** | **Tera** | $2^{40}$ |
| **m** | **milli** | $10^{-3}$ |
| **μ** | **μικρο** | $10^{-6}$ |
| **n** | **nano** | $10^{-9}$ |

**Οι πολλαπλασιαστικοί παράγοντες που χρησιμοποιούνται περισσότερο στο χώρο των υπολογιστών**

Σε κάθε υπολογιστή υπάρχουν διάφορες μορφές μνήμης, που διαφέρουν μεταξύ τους ως προς τα ποσοτικά χαρακτηριστικά τους, όπως:

- ⇨ *Το κόστος τους*, σε δρχ ανά bit αποθηκευμένης πληροφορίας
- ⇨ *Το χρόνο προσπέλασης* (access time). Αυτός είναι ο χρόνος που απαιτείται για την ανάγνωση ή εγγραφή στη μνήμη μίας μονάδας πληροφορίας. Ο χρόνος προσπέλασης σήμερα μετράται σε εκατομμυριοστά του δευτερολέπτου (ns).
- ⇨ *Τη χωρητικότητά της* (storage capacity), δηλαδή τον συνολικό όγκο πληροφοριών που μπορούν να αποθηκευθούν στη μονάδα αυτή. Η χωρητικότητα μιας μονάδας μνήμης μετράται σε εκατομμύρια bytes ή MBytes*.

Οι σχεδιαστές ενός συστήματος μνήμης επιδιώκουν:



Να μεγιστοποιήσουν την ταχύτητα μεταφοράς των πληροφοριών από και προς τη μονάδα αποθήκευσης. Έτσι ελαχιστοποιείται ο χρόνος προσπέλασης στη μνήμη.

Να ελαχιστοποιήσουν το μέσο κόστος αποθήκευσης, ώστε μονάδες μνήμης με μεγάλη χωρητικότητα να έχουν προσιτή τιμή.

Η διαδικασία της μεταφοράς πληροφοριών μεταξύ των διαφόρων μονάδων μνήμης να είναι αυτοματοποιημένη. Έτσι, όταν οι προγραμματιστές γράφουν προγράμματα, τα οποία χρησιμοποιούν τις μονάδες μνήμης, δε χρειάζεται να καταβάλουν προσπάθεια στον τομέα αυτό.

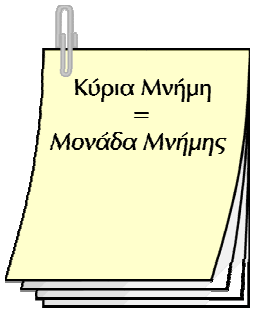
* Στο χώρο των υπολογιστών, το πρόθεμα Mega συμβολίζει έναν πολλαπλασιαστικό παράγοντα με τιμή $2^{20}$ = 1.048.576, και όχι τον συνηθισμένο παράγοντα $10^6$ = 1.000.000. Επειδή όμως οι δύο τιμές είναι παραπλήσιες, πολλές φορές λέμε «εκατομμύρια» και εννοούμε 1.048.576.

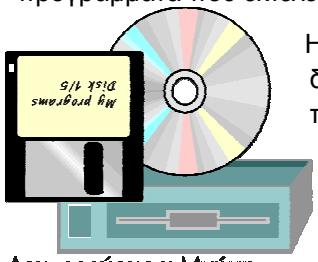
Να παρέχουν αποτελεσματικούς μηχανισμούς που θα προστατεύουν τα αποθηκευμένα δεδομένα. Ο κίνδυνος για αυτά προέρχεται από σφάλματα ή ανεπίτρεπτες ενέργειες των προγραμμάτων, που είναι δυνατόν να οδηγήσουν στην καταστροφή των δεδομένων ή σε αθέμιτη αντιγραφή τους.

## Κύρια και δευτερεύουσα μνήμη

Η μνήμη ενός υπολογιστή διακρίνεται συνήθως στην *κύρια μνήμη* (main memory) και τη *δευτερεύουσα μνήμη* (secondary memory).

Η κύρια μνήμη είναι αυτή που είδαμε στο μάθημα 3.1 και μπορεί να προσπελαστεί απ' ευθείας από την ΚΜΕ, χωρίς να παρεμβληθεί η μονάδα εισόδου/εξόδου. Ένα βασικό της χαρακτηριστικό είναι ότι τα περιεχόμενά της χάνονται, όταν σταματήσει η λειτουργία του υπολογιστή. Στην κύρια μνήμη αποθηκεύονται από την ΚΜΕ τα προγράμματα που εκτελεί, καθώς και τα δεδομένα των προγραμμάτων αυτών.



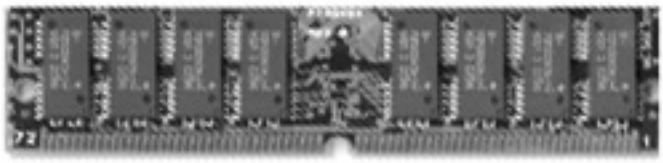




Δευτερεύουσα Μνήμη = Περιφερειακές Συσκευές

Η δευτερεύουσα ή βοηθητική μνήμη απαρτίζεται από διάφορες περιφερειακές συσκευές. Σ' αυτήν αποθηκεύονται τα δεδομένα που δεν είναι άμεσα απαραίτητα στην ΚΜΕ, τα οποία διατηρούνται αναλλοίωτα και μετά τη διακοπή λειτουργίας του υπολογιστή. Η δευτερεύουσα μνήμη αποτελείται συνήθως από μαγνητικούς δίσκους, οπτικούς δίσκους (CD-ROM) και μαγνητικές ταινίες.

## Μνήμη άμεσης προσπέλασης

Η κύρια μνήμη του υπολογιστή είναι *μνήμη άμεσης προσπέλασης* (Random Access Memory). Σε μία τέτοια μνήμη, ο χρόνος που χρειάζεται για να διαβάσουμε ή να γράψουμε μία πληροφορία στη μνήμη είναι πάντα ο ίδιος, ανεξάρτητα από τη συγκεκριμένη διεύθυνση της πληροφορίας αυτής.



Μια μονάδα μνήμης άμεσης προσπέλασης πραγματοποιείται συνήθως με ένα ολοκληρωμένο κύκλωμα, το οποίο επικοινωνεί με τον υπολογιστή με ηλεκτρικά σήματα που περνούν από τους ακροδέκτες του. Υπάρχουν τέτοια σήματα για τον προσδιορισμό μιας διεύθυνσης, για την ανταλλαγή των πληροφοριών που αποθηκεύει η μνήμη και για τον προσδιορισμό της λειτουργίας που θα επιτελέσει.

Οι δύο βασικές τεχνολογίες που χρησιμοποιούνται σήμερα για την κύρια μνήμη είναι (α) οι στατικές μνήμες (SRAM), με χρόνο προσπέλασης 10-20 ns και (β) οι δυναμικές μνήμες (DRAM), με χρόνο προσπέλασης 90-120 ns*. Το κόστος των δυναμικών μνημών είναι περίπου το 25% αυτού των στατικών μνημών.

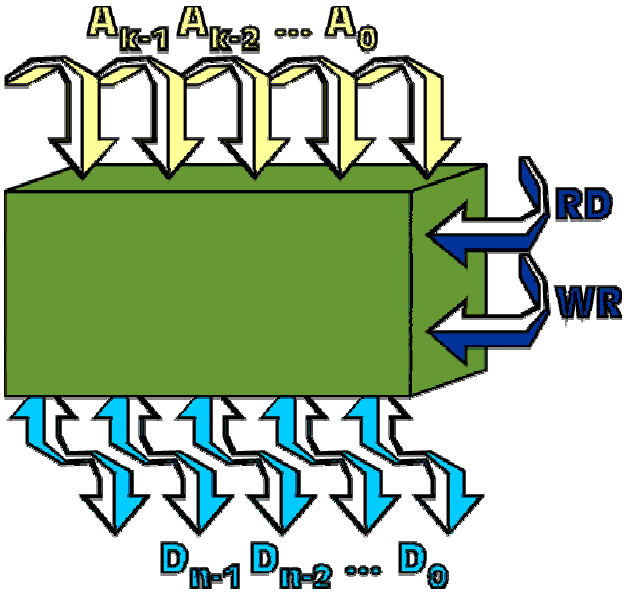
Μπορούμε να φανταζόμαστε μια μονάδα μνήμης άμεσης προσπέλασης σαν ένα πίνακα από λέξεις, κάθε μία από τις οποίες αντιστοιχεί σε μία διεύθυνση. Αν η μονάδα περιέχει $2^K$ λέξεις, οι διευθύνσεις τους θα είναι οι αριθμοί από 0 έως $2^K$-1, και θα απαιτούν κ bits για να παρασταθούν.

**Αν το πλήθος των λέξεων σε μία μονάδα μνήμης είναι δύναμη του 2, π.χ. $2^K$, τότε οι διευθύνσεις των λέξεων χρειάζονται *ακριβώς* K bits για να παρασταθούν.**

* Οι τιμές αυτές, οι οποίες βελτιώνονται συνεχώς, είναι ενδεικτικές για το 1999.

Έτσι λοιπόν το ολοκληρωμένο κύκλωμα της μνήμης επικοινωνεί με τις υπόλοιπες μονάδες του υπολογιστή με:

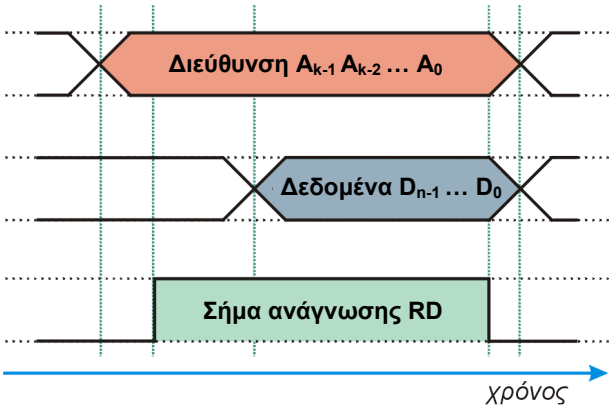
- ✓ Τις *γραμμές διεύθυνσης*, που είναι k το πλήθος, και έχουν τα ονόματα $A_{k-1}$-$A_0$.
- ✓ Τις *γραμμές δεδομένων*, που είναι n το πλήθος (n είναι το μέγεθος σε bits της λέξης για τη μνήμη) με ονόματα $D_{n-1}$-$D_0$.
- ✓ Μια γραμμή για την ενεργοποίηση της εγγραφής στη μνήμη, με το όνομα WR (από το Write, γράφω)



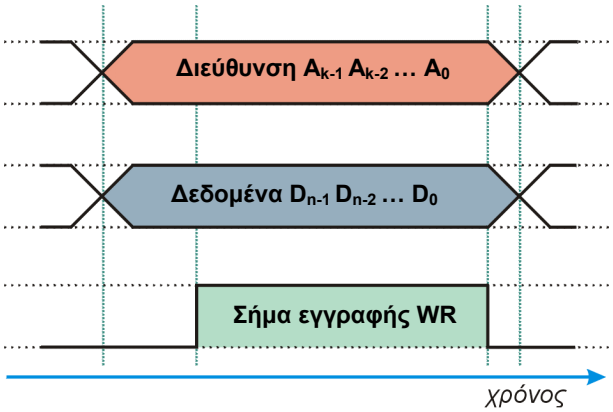

- ✓ Μια γραμμή για την ενεργοποίηση της ανάγνωσης από τη μνήμη, με το όνομα RD (από το Read, διαβάζω)

Για να γίνει μια λειτουργία ανάγνωσης από τη μονάδα της μνήμης, οι γραμμές διεύθυνσης ($A_{k-1}$…$A_0$) μεταφέρουν τη δυαδική διεύθυνση της λέξης που θα διαβαστεί. Στη συνέχεια ο επεξεργαστής ενεργοποιεί το σήμα RD, και μετά από μια μικρή καθυστέρηση οι γραμμές δεδομένων ($D_{n-1}$…$D_0$) ενεργοποιούνται με τα δεδομένα της λέξης.

Με παρόμοιο τρόπο, για να γίνει μια εγγραφή στη μονάδα της μνήμης, οι γραμμές *διεύθυνσης* μεταφέρουν τη διεύθυνση της λέξης που θα γραφεί και οι γραμμές δεδομένων τα περιεχόμενά της. Κατόπιν ο επεξεργαστής ενεργοποιεί το σήμα WR και μετά από ένα μικρό χρονικό διάστημα η λέξη έχει εγγραφεί στη μνήμη.




**Χρονική εξέλιξη των σημάτων κατά την ανάγνωση από τη μνήμη**




**Χρονική εξέλιξη των σημάτων κατά την εγγραφή στη μνήμη**

## Ο χώρος διευθύνσεων του επεξεργαστή

Ο επεξεργαστής του υπολογιστή μπορεί να αναφερθεί άμεσα στις λέξεις της μνήμης οι οποίες περιγράφονται από μια διεύθυνση μεταξύ 0 και $2^k$-1, αφού οι γραμμές διεύθυνσης είναι k. Το σύνολο των διευθύνσεων αυτών αποτελεί το *χώρο διευθύνσεων* (address space) του επεξεργαστή.

Οι σύγχρονοι επεξεργαστές χαρακτηρίζονται από σχετικά μεγάλο χώρο διευθύνσεων.

> Οι επεξεργαστές Intel της οικογένειας Pentium διαθέτουν 32 γραμμές διεύθυνσης, οπότε έχουν χώρο διευθύνσεων $2^{32}$ bytes = $4 \times 2^{30}$ bytes = 4 GB.

Η αλματώδης αύξηση των απαιτήσεων των εφαρμογών σε μνήμη έχει απαξιώσει πρόωρα πολλούς τύπους υπολογιστών, επειδή αυτοί μπορούσαν να αναφερθούν σε μικρό χώρο διευθύνσεων, και έτσι αποδείχθηκαν ανεπαρκείς για νεότερα προγράμματα και λειτουργικά συστήματα.

## Άλλοι τύποι μνήμης

Η δευτερεύουσα μνήμη χρησιμοποιείται για τη μόνιμη και οικονομικότερη αποθήκευση μεγάλου όγκου πληροφοριών.

Κύριο χαρακτηριστικό των περιφερειακών μονάδων που αποτελούν τη δευτερεύουσα μνήμη είναι ο συνδυασμός ηλεκτρομηχανικών διατάξεων για τη μαγνητική ή οπτική αποθήκευση και ανάκτηση μεγάλου όγκου πληροφοριών. Το μηχανικό μέρος αυτών των συσκευών τις καθιστά βραδύτερες σε σχέση με την κύρια μνήμη, της οποίας η λειτουργία είναι καθαρά ηλεκτρονική.

Επειδή η διαδικασία του εντοπισμού και της ανάκτησης των δεδομένων στις μονάδες αυτές είναι σχετικά χρονοβόρα, οι λειτουργίες ανάγνωσης και εγγραφής γίνονται για μεγαλύτερες ενότητες δεδομένων από ό,τι στην κύρια μνήμη.

> Σε ένα σκληρό δίσκο η ενότητα είναι συχνά 512 bytes, δηλαδή πολύ μεγαλύτερη από τη λέξη του επεξεργαστή, που σε πολλούς επεξεργαστές είναι μόνο 1 byte.
>
> Για να διαβάσουμε από το δίσκο αυτό 2000 συνεχόμενα bytes θα χρειαστούν 4 αναγνώσεις. Αν όμως η μονάδα διαβάζει ενότητες των 8 bytes, θα απαιτούνταν 250 λειτουργίες ανάγνωσης και 50 φορές περισσότερος χρόνος!

Ας σημειώσουμε ότι η δευτερεύουσα μνήμη δεν είναι άμεσα προσπελάσιμη από τον επεξεργαστή· τα περιεχόμενά της πρέπει πρώτα να μεταφερθούν στην κύρια μνήμη με τη βοήθεια της μονάδας εισόδου-εξόδου, και μετά να χρησιμοποιηθούν από τον επεξεργαστή.

**Ανακεφαλαίωση**

*Η μνήμη του υπολογιστή συνήθως οργανώνεται σε ομάδες δυαδικών ψηφίων, τις λέξεις. Κάθε λέξη έχει μία διεύθυνση, η οποία είναι ένας αριθμός. Η κύρια μνήμη του υπολογιστή συνήθως είναι μνήμη άμεσης προσπέλασης, όπου κάθε λέξη απαιτεί τον ίδιο χρόνο για να διαβαστεί ή να γραφεί. Αυτό γίνεται ενεργοποιώντας, με τη διεύθυνση της λέξης, τις κατάλληλες γραμμές εξόδου στο ολοκληρωμένο κύκλωμα της μνήμης και διαβάζοντας ή γράφοντας τα δεδομένα στις αντίστοιχες γραμμές-ακροδέκτες του κυκλώματος.*

**Γλωσσάριο όρων**

| | |
|---|---|
| *Ανάγνωση* | *Read* |
| *Γραμμές Δεδομένων* | *Data Lines* |
| *Γραμμές Διεύθυνσης* | *Address Lines* |
| *Δευτερεύουσα Μνήμη* | *Secondary Memory* |
| *Εγγραφή* | *Write* |
| *Ενότητα* | *Block ή Frame* |
| *Κύρια Μνήμη* | *Main Memory* |
| *Λέξη* | *Word* |
| *Μνήμη Άμεσης Προσπέλασης* | *Random Access Memory* |
| *Χρόνος Προσπέλασης* | *Access Time* |
| *Χωρητικότητα* | *Storage Capacity* |
| *Χώρος Διευθύνσεων* | *Address Space* |

## Ερωτήσεις

? Τι είναι η λέξη μνήμης;

? Τι είναι η ενότητα μνήμης;

? Τι είναι η κύρια και τι η δευτερεύουσα μνήμη;

? Πώς επικοινωνεί το ολοκληρωμένο κύκλωμα της μνήμης με τον υπόλοιπο υπολογιστή;

? Τι είναι ο χώρος διευθύνσεων ενός επεξεργαστή;

# Μάθημα 4.2 Ιεραρχία Μνήμης

**Σκοπός του μαθήματος**

*Σκοπός του μαθήματος αυτού είναι να περιγράψει την ιεραρχική οργάνωση της μνήμης στους υπολογιστές.*

**Τι θα μάθεις;**

*Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:*

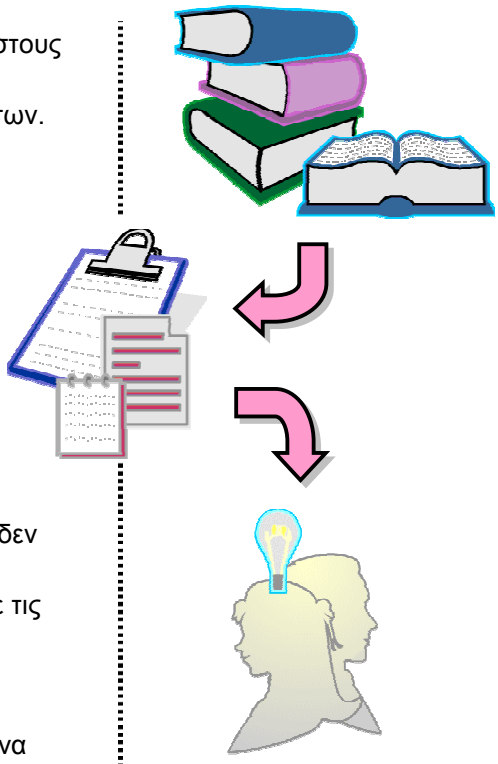
- *Να εξηγείς την τοπικότητα της αναφοράς των προγραμμάτων*
- *Να απαριθμείς τα διάφορα επίπεδα στην ιεραρχία της μνήμης*
- *Να περιγράφεις τις ιδιότητες του κάθε επιπέδου*

## Η τοπικότητα αναφοράς για τα προγράμματα και τα δεδομένα

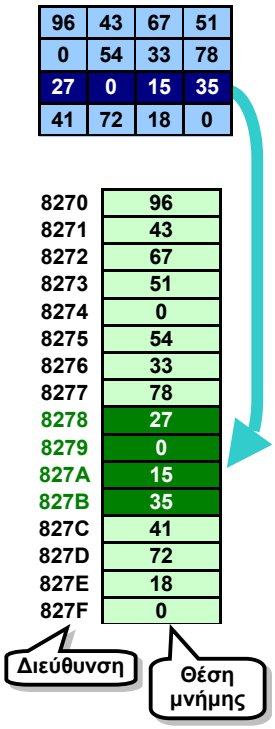
Η σπουδαιότερη ιδιότητα που χαρακτηρίζει την εκτέλεση των προγραμμάτων των υπολογιστών είναι η *τοπικότητα της αναφοράς* (locality of reference). Όταν ένα πρόγραμμα εκτελείται από την ΚΜΕ, η επόμενη εντολή που θα ανακληθεί κάθε φορά από τη μνήμη και θα εκτελεστεί στην ΚΜΕ θα ανήκει πιθανότατα σε κάποια γειτονική θέση σε σχέση με αυτή που εκτελέστηκε αμέσως προηγουμένως.

Στην καθημερινή μας ζωή η τοπικότητα της αναφοράς εμφανίζεται στους αριθμούς τηλεφώνου που χρησιμοποιούμε, γιατί οι περισσότερες τηλεφωνικές κλήσεις μας γίνονται σε περιορισμένο αριθμό προσώπων.

Για να εκμεταλλευθούμε την τοπικότητα αυτή, καταχωρίζουμε τους αριθμούς που μας ενδιαφέρουν σε προσωπικά ευρετήρια ή σημειωματάρια, αντί να χρησιμοποιούμε τους επίσημους τηλεφωνικούς καταλόγους. Επίσης συνηθίζουμε να απομνημονεύουμε μερικούς τηλεφωνικούς αριθμούς που χρησιμοποιούμε πάρα πολύ τακτικά, ώστε η ανάκλησή τους να γίνεται ακόμη πιο γρήγορα. Έτσι, διευκολυνόμαστε στις περισσότερες επικοινωνίες μας, αποφεύγοντας την αναζήτηση στοιχείων στους ογκώδεις καταλόγους, στους οποίους καταφεύγουμε μόνο σε σπάνιες περιπτώσεις.



Λόγω του περιορισμένου τους μεγέθους, τα προσωπικά ευρετήρια δεν μπορούν να καλύψουν πάντα τους αριθμούς τηλεφώνων που χρησιμοποιούμε πολύ συχνά. Αναγκαζόμαστε τότε να διαγράψουμε τις παλαιότερες ή λιγότερο χρησιμοποιούμενες εγγραφές, για να τις αντικαταστήσουμε με νέες, αναμένοντας ότι αυτές θα ζητηθούν συχνότερα στο άμεσο μέλλον. Κάτι παρόμοιο γίνεται και με τους απομνημονευμένους αριθμούς, αφού ξεχνάμε όσους έχουμε καιρό να καλέσουμε, αλλά θυμόμαστε αυτούς που καλέσαμε πρόσφατα.




Στην περίπτωση των προγραμμάτων, η τοπικότητα της αναφοράς είναι πάρα πολύ συχνή. Εκτός από τις εντολές, ισχύει και για τις αναφορές προς τα δεδομένα που είναι αποθηκευμένα στη μνήμη.

> Ένα πρόγραμμα μαθηματικών, που εκτελεί πράξεις σε πίνακες, αποθηκεύει τα στοιχεία των πινάκων κατά γραμμές στη μνήμη. Ο πίνακας του σχήματος, που έχει διαστάσεις $4 \times 4$, αποθηκεύεται ξεκινώντας από τη διεύθυνση $8270_{(16)}$ της μνήμης.
>
> Όπως βλέπουμε και στο σχήμα, τα περιεχόμενα του πίνακα αποθηκεύονται σε συνεχόμενες θέσεις της μνήμης. Όταν π.χ. το πρόγραμμα εκτελεί μια πράξη που χρησιμοποιεί τον πίνακα κατά γραμμές, θα προσπελάσει με τη σειρά όλες τις θέσεις μνήμης που του αναλογούν, ξεκινώντας από τη διεύθυνση 8270, και συνεχίζοντας μέχρι την 827F.

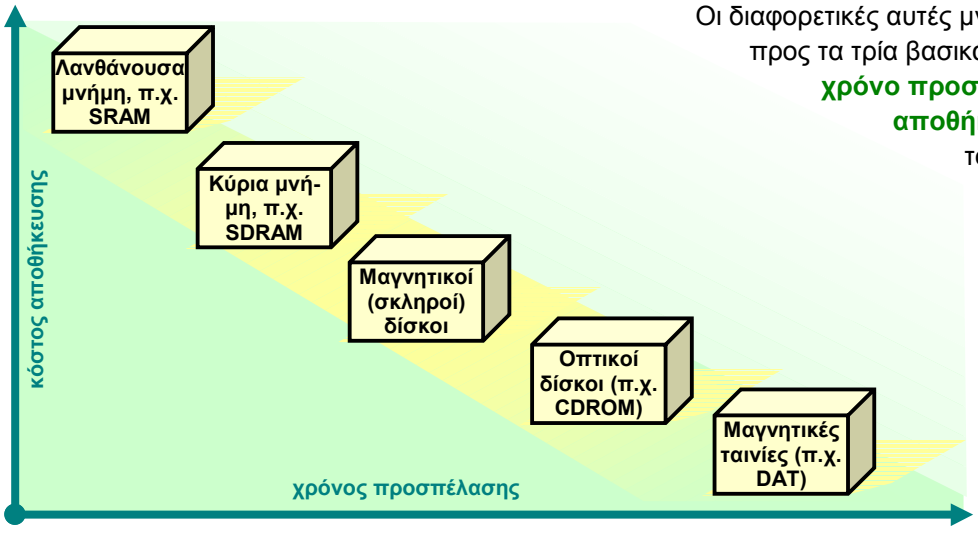
Όπως λοιπόν διαχειρίζεται ένας άνθρωπος τους τηλεφωνικούς αριθμούς, μπορεί και ο υπολογιστής να διαχειριστεί τα δεδομένα του: τα δεδομένα που χρησιμοποιούνται συχνά να βρίσκονται αποθηκευμένα σε μία μνήμη με μικρό μέγεθος και γρήγορη προσπέλαση, ενώ τα υπόλοιπα δεδομένα θα είναι αποθηκευμένα σε μνήμη με μεγαλύτερη χωρητικότητα για την οποία όμως απαιτείται πιο μεγάλος χρόνος προσπέλασης.

Στο εσωτερικό του επεξεργαστή αυτή η μέθοδος εφαρμόζεται με τη *λανθάνουσα μνήμη* (cache memory), την οποία θα δούμε αναλυτικά στο Μάθημα 4.4. Η μνήμη αυτή κρατά τα δεδομένα που χρησιμοποιούνται συχνά. Έτσι αυτά είναι άμεσα διαθέσιμα για χρήση και δεν είναι απαραίτητη μία εξωτερική επικοινωνία με την κύρια μνήμη.

Η ίδια μέθοδος εφαρμόζεται και στο σύστημα της *εικονικής μνήμης* (virtual memory), την οποία διαχειρίζεται το λειτουργικό σύστημα (θα παρουσιαστεί στο Κεφάλαιο 9). Εκεί η κύρια μνήμη του συστήματος παίζει το ρόλο της μικρής και γρήγορης μνήμης, ενώ η «αργή» και μεγαλύτερη σε μέγεθος μνήμη βρίσκεται σε κάποια περιφερειακή συσκευή, συνήθως στο σκληρό δίσκο.

## Ιεραρχία μνήμης στους σύγχρονους υπολογιστές

Βλέπουμε ότι τα σύγχρονα υπολογιστικά συστήματα, για να καλύψουν διαφόρων επιπέδων ανάγκες σε ταχύτητα και χωρητικότητα μνήμης, επιστρατεύουν πολλαπλούς τύπους μνήμης.



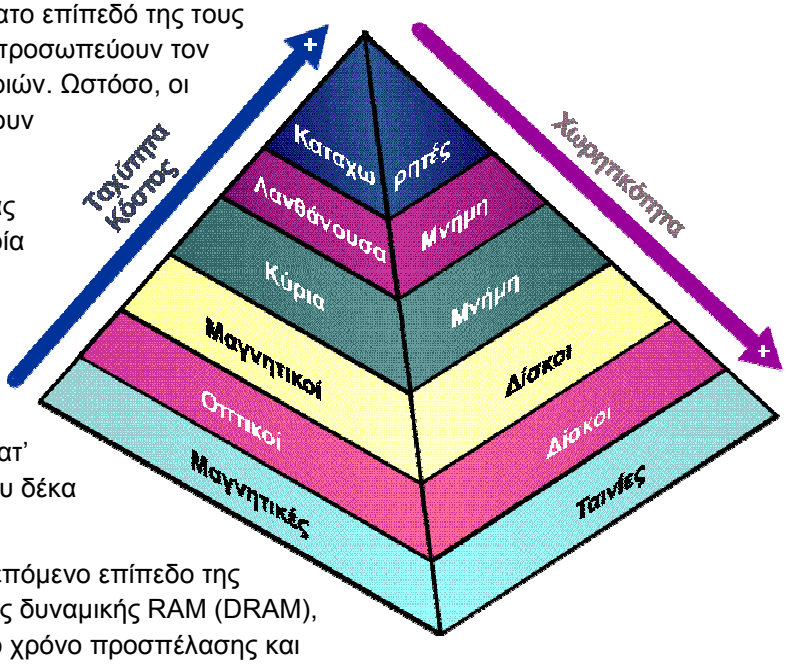

Οι διαφορετικές αυτές μνήμες διαφοροποιούνται ως προς τα τρία βασικά τους χαρακτηριστικά: το **χρόνο προσπέλασης**, το **κόστος αποθήκευσης** και τη **χωρητικότητά** τους. Μάλιστα, ο χρόνος προσπέλασης και η χωρητικότητα είναι «ανάλογα» μεταξύ τους, δηλαδή όταν το ένα αυξάνει, τότε αυξάνει και το άλλο, ενώ αυτά τα δύο μεγέθη είναι «αντιστρόφως ανάλογα» προς το κόστος αποθήκευσης.

**Η κύρια μνήμη ενός προσωπικού υπολογιστή είναι περίπου 1000 φορές πιο γρήγορη από ένα σκληρό δίσκο και γύρω στις 100 φορές πιο μικρή σε χωρητικότητα. Επίσης είναι περίπου 100 φορές πιο ακριβή ανά bit.**

Για να μπορεί να εκμεταλλευθεί ο υπολογιστής τα διαφορετικά χαρακτηριστικά των μνημών που διαθέτει, χωρίς το κόστος του να είναι παράλογο, οι μονάδες μνήμης μπορούν να θεωρηθούν ότι είναι οργανωμένες σε μια ιδεατή *ιεραρχία* (memory hierarchy). Στην «κορυφή» της ιεραρχίας, όπου το μέγεθος της μνήμης είναι πολύ μικρό, βρίσκονται οι πιο γρήγορες και ακριβές μνήμες. Στη «βάση» της ιεραρχίας το μέγεθος της μνήμης είναι μεγάλο και τοποθετούνται αργές και φθηνές μνήμες.

Με τον τρόπο αυτό ελαχιστοποιείται ο μέσος χρόνος προσπέλασης στη μνήμη και το κόστος αποθήκευσης ανά μονάδα πληροφορίας, ενώ μεγιστοποιείται ο συνολικός χώρος αποθήκευσης του συστήματος.

Μια τυπική ιεραρχία μνήμης έχει στο ανώτατο επίπεδό της τους **καταχωρητές** της ΚΜΕ, επειδή αυτοί αντιπροσωπεύουν τον αμεσότερο τρόπο αποθήκευσης πληροφοριών. Ωστόσο, οι καταχωρητές δεν μπορούν να αποθηκεύσουν αξιόλογο όγκο δεδομένων.




Στο αμέσως επόμενο επίπεδο της ιεραρχίας τοποθετείται η **λανθάνουσα μνήμη**, η οποία συνήθως υλοποιείται με ψηφίδες στατικής RAM (SRAM). Στη λανθάνουσα μνήμη αντιγράφονται τμήματα της κύριας μνήμης, με την προσδοκία ότι οι επόμενες αναφορές προς την κύρια μνήμη θα μπορέσουν να εξυπηρετηθούν κατ' ευθείαν.Η λανθάνουσα μνήμη είναι περίπου δέκα φορές ταχύτερη από την κύρια μνήμη.

Η **κύρια μνήμη** καταλαμβάνει το αμέσως επόμενο επίπεδο της ιεραρχίας. Υλοποιείται συνήθως με ψηφίδες δυναμικής RAM (DRAM), οι οποίες χαρακτηρίζονται από μεγαλύτερο χρόνο προσπέλασης και μικρότερο κόστος ανά μονάδα αποθηκευμένης πληροφορίας σε σχέση με τα κυκλώματα τύπου SRAM. Η κύρια μνήμη χρησιμοποιείται, όταν μία λέξη δεν βρεθεί στη λανθάνουσα μνήμη. Τότε, εκτός από την ανάγνωση ή εγγραφή αυτής της λέξης, ενεργοποιείται και ο μηχανισμός για την αντιγραφή των γειτονικών της θέσεων από την κύρια προς τη λανθάνουσα μνήμη. Αυτό γίνεται γιατί η τοπικότητα της αναφοράς προβλέπει ότι είναι πιθανόν οι επόμενες προσπελάσεις στη μνήμη να γίνουν σε λέξεις γειτονικές με αυτή που τώρα προσπελάστηκε.

Στα τελευταία επίπεδα της ιεραρχίας τοποθετούνται οι μονάδες των **μαγνητικών και οπτικών δίσκων**, και οι μονάδες των **μαγνητικών ταινιών**. Αυτές οι μονάδες εξασφαλίζουν τη μόνιμη και ασφαλή αποθήκευση μεγάλου όγκου δεδομένων. Η ταχύτητα προσπέλασης σε αυτές είναι σχετικά χαμηλή, λόγω των ηλεκτρομηχανικών διαδικασιών προσπέλασης στα δεδομένα. Οι μαγνητικές ταινίες είναι ιδεώδεις για την αποθήκευση *εφεδρικών αντιγράφων* (backup copies) των δεδομένων. Οι μαγνητικοί δίσκοι προσφέρονται και για την υλοποίηση του μηχανισμού της εικονικής μνήμης.


*Η τοπικότητα της αναφοράς βοηθά στην οργάνωση του συστήματος μνήμης με τρόπο που επιταχύνει την εκτέλεση των προγραμμάτων. Τα υπολογιστικά συστήματα διαθέτουν πολλούς τύπους μνήμης, που διαφέρουν ως προς τρία χαρακτηριστικά: τη χωρητικότητα, το κόστος και την ταχύτητα προσπέλασης. Οι διάφοροι τύποι μνήμης σχηματίζουν μια ιεραρχία, στην κορυφή της οποίας βρίσκονται οι καταχωρητές της ΚΜΕ και στη βάση της οι αργές και φθηνές μνήμες όπως οι μαγνητικές ταινίες.*


| | |
|---|---|
| *Εικονική Μνήμη* | *Virtual Memory* |
| *Εφεδρικό Αντίγραφο* | *Backup Copy* |
| *Ιεραρχία Μνήμης* | *Memory Hierarchy* |
| *Λανθάνουσα Μνήμη* | *Cache Memory* |
| *Τοπικότητα Αναφοράς* | *Locality Of Reference* |

## Ερωτήσεις

- ? Τι είναι η τοπικότητα της αναφοράς στα προγράμματα;
- ? Πώς είναι οργανωμένη η ιεραρχία της μνήμης;
- ? Ποιο είδος μνήμης βρίσκεται στην κορυφή της ιεραρχίας της μνήμης;
- ? Ποια μεγέθη αυξάνονται όσο ανεβαίνουμε σε ανώτερα επίπεδα της ιεραρχίας της μνήμης;
- ? Ποια μεγέθη μειώνονται όσο ανεβαίνουμε σε ανώτερα επίπεδα της ιεραρχίας της μνήμης;
- ? Για ποιες λειτουργίες είναι κατάλληλες οι μνήμες στο κατώτατο επίπεδο της ιεραρχίας της μνήμης;

# Μάθημα 4.3 Οργάνωση και Σχεδίαση Μνήμης

**Σκοπός του μαθήματος**

*Σκοπός του μαθήματος αυτού είναι να παρουσιάσει τους τρόπους σχεδίασης ενός συστήματος κύριας μνήμης.*

**Τι θα μάθεις;**

*Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:*

- *Να καταρτίζεις το Χάρτη Διευθύνσεων για ένα σύστημα μνήμης*
- *Να περιγράφεις την επικοινωνία του επεξεργαστή με την κάθε ψηφίδα*
- *Να εξηγείς τον τρόπο λειτουργίας ενός συστήματος με διαφύλλωση μνήμης*

## Σχεδίαση συστήματος κύριας μνήμης

Η κύρια μνήμη σε ένα υπολογιστικό σύστημα αποτελείται από τη *μνήμη μόνο ανάγνωσης* (ROM) και από τη *μνήμη ανάγνωσης - εγγραφής* (RAM)[1].

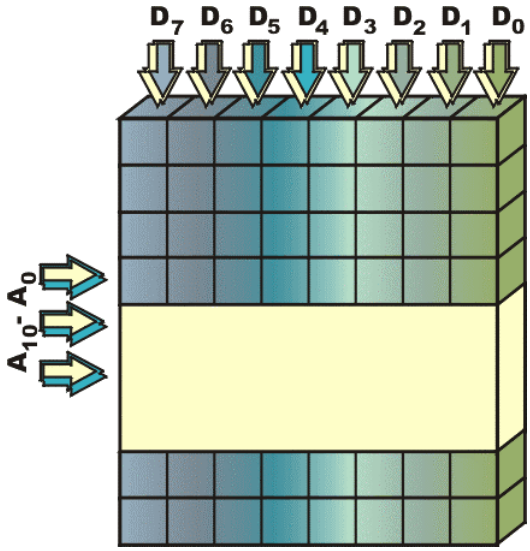
Η κύρια μνήμη είναι άμεσα προσπελάσιμη από τον επεξεργαστή, χρησιμοποιώντας τα σήματα ανάγνωσης (RD), εγγραφής (WR), Διευθύνσεων ($A_{k-1}$ - $A_0$) και Δεδομένων ($D_{n-1}$ - $D_0$), όπως είδαμε στο Μάθημα 4.1, σύμφωνα με τον κύκλο ανάγνωσης και τον κύκλο εγγραφής της μνήμης.

Για τη δημιουργία της κύριας μνήμης συνήθως χρησιμοποιούνται ψηφίδες ολοκληρωμένων κυκλωμάτων μνημών τύπου ROM και RAM. Οι ψηφίδες αυτές χαρακτηρίζονται από τις *διαστάσεις* (dimensions) τους, δηλαδή από το πλήθος των λέξεων που περιέχουν και το μήκος σε bits της κάθε λέξης.

> Μια ψηφίδα διαστάσεων 1024×8 περιέχει 1024 λέξεις των 8 bits η κάθε μία.

Με την κατάλληλη διασύνδεση τέτοιων ψηφίδων είναι δυνατή η σχεδίαση μονάδων κύριας μνήμης οποιωνδήποτε διαστάσεων (πλήθος και μέγεθος λέξεων), για την εξυπηρέτηση των συγκεκριμένων απαιτήσεων ενός υπολογιστικού συστήματος.

> Ένα σύστημα στηρίζεται σε επεξεργαστή ο οποίος διαθέτει 8 γραμμές δεδομένων ($D_7$ – $D_0$) και 11 γραμμές διευθύνσεων ($A_{10}$ – $A_0$). Ο χώρος μνήμης του υπολογιστή αυτού λοιπόν θα αποτελείται από $2^{11}$ = 2048 λέξεις των 8 bits.




[1] Ο όρος RAM ο οποίος σημαίνει Μνήμη Τυχαίας Προσπέλασης, έχει επικρατήσει ιστορικά, χωρίς να περιγράφει επακριβώς τη μνήμη στην οποία αναφερόμαστε

Θέλουμε να σχεδιάσουμε μια μονάδα κύριας μνήμης που να διαθέτει μνήμη τύπου RAM για τις διευθύνσεις από 0 έως 1023 και μνήμη τύπου ROM για τις διευθύνσεις από 1024 έως 2047. Διαθέτουμε ψηφίδες RAM διαστάσεων 256x8 bits και ROM διαστάσεων 512x4 bits.

Για τον σκοπό αυτό καταρτίζουμε τον *Χάρτη Διευθύνσεων* (Address Map), όπου καταγράφονται όλες οι αναγκαίες ψηφίδες μνήμης, μαζί με τους αντίστοιχες περιοχές του χώρου διευθύνσεων του επεξεργαστή.

Επειδή οι μνήμες ROM περιέχουν λέξεις των 4 bits, για κάθε λέξη της μνήμης που αντιστοιχεί στη ROM (8 bits) θα χρησιμοποιήσουμε 2 λέξεις των 4 bits με την ίδια διεύθυνση από δύο διαφορετικές ψηφίδες ROM.

Οι διευθύνσεις που καταλαμβάνονται από μνήμη RAM, οι 0-1023, απαιτούν 1024/256 = 4 ψηφίδες, τις $RAM_0$, $RAM_1$, $RAM_2$ και $RAM_3$. Οι διευθύνσεις που καταλαμβάνονται από μνήμη ROM, οι 1024-2047 απαιτούν 1024/512 = 2 ζεύγη ψηφίδων ROM, τα ($ROM_0$, $ROM_1$) και ($ROM_2$, $ROM_3$).

Ο χάρτης διευθύνσεων θα είναι λοιπόν:

| Ψηφίδα | | Διευθύνσεις $A_{10}$ | $A_9$ | $A_8$ | $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $RAM_0$ | από | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = | $0_{(10)}$ |
| | έως | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | $255_{(10)}$ |
| $RAM_1$ | από | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = | $256_{(10)}$ |
| | έως | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | $511_{(10)}$ |
| $RAM_2$ | από | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = | $512_{(10)}$ |
| | έως | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | $767_{(10)}$ |
| $RAM_3$ | από | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = | $768_{(10)}$ |
| | έως | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | $1023_{(10)}$ |
| $ROM_0$ και $ROM_1$ | από | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = | $1024_{(10)}$ |
| | έως | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | $1535_{(10)}$ |
| $ROM_2$ και $ROM_3$ | από | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = | $1536_{(10)}$ |
| | έως | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | $2047_{(10)}$ |

Από το χάρτη διευθύνσεων βλέπουμε για κάθε διεύθυνση μνήμης την ψηφίδα στην οποία αντιστοιχεί. Για να βρούμε την εσωτερική διεύθυνση μέσα στην ψηφίδα που πρέπει να προσπελαστεί, αφαιρούμε την αρχική διεύθυνση για την ψηφίδα.

Η διεύθυνση 777 βρίσκεται μέσα στο διάστημα 768-1023, άρα βρίσκεται στην ψηφίδα $RAM_3$. Η εσωτερική της διεύθυνση μέσα στην ψηφίδα είναι η 777-768 = 9.

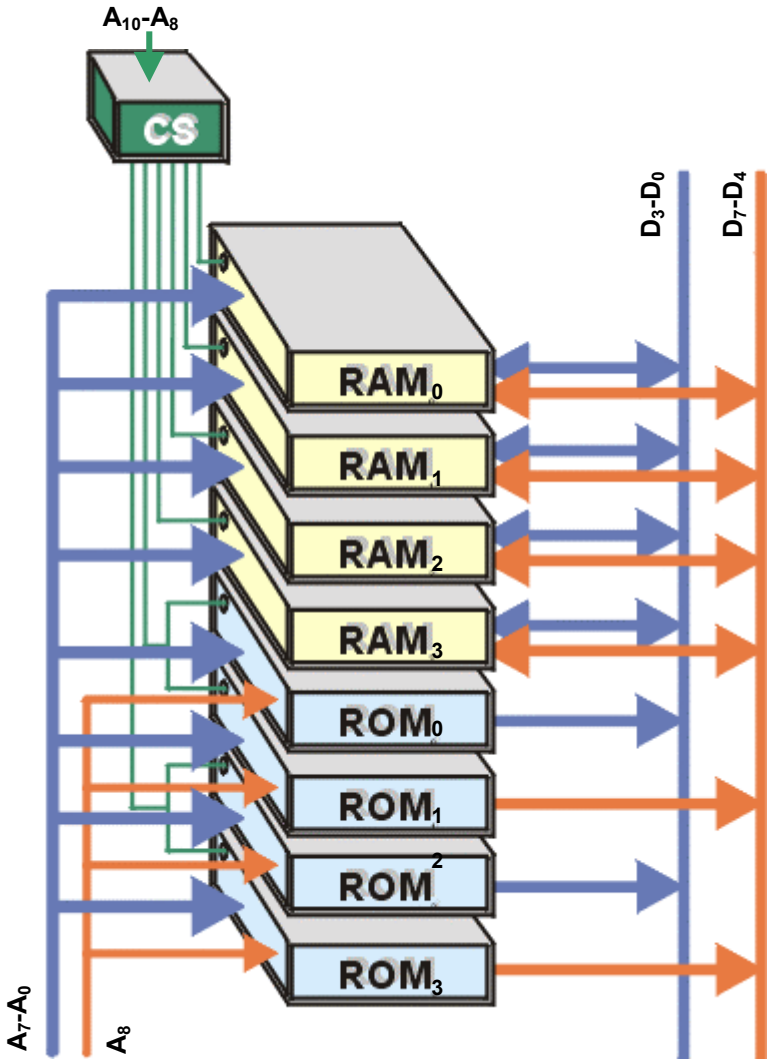
Η διεύθυνση 1548 καλύπτεται από τις ψηφίδες $ROM_2$ και $ROM_3$, από τις εσωτερικές θέσεις 1548-1536=12 κάθε ψηφίδας. Κάθε ψηφίδα παρέχει 4 από τα 8 bit της λέξης.

Προκειμένου να γίνεται επιλεκτική ενεργοποίηση των ψηφίδων της μνήμης, ανάλογα με την τιμή της διεύθυνσης, κάθε ψηφίδα έχει μια επιπλέον γραμμή εισόδου, την *επιλογή ψηφίδας* (chip select, CS).

Όταν η είσοδος CS είναι ανενεργή (δηλαδή έχει την τιμή 0), τότε η ψηφίδα μένει αδρανής και δεν ανταποκρίνεται σε κανένα από τα άλλα σήματα (RD ή WR). Μόνον όταν η είσοδος CS είναι ενεργή (έχει την τιμή 1) η ψηφίδα της μνήμης ανταποκρίνεται, σύμφωνα με τον κύκλο ανάγνωσης μνήμης (όταν RD=1) ή με τον κύκλο εγγραφής μνήμης (όταν WR=1), όπως είδαμε στο Μάθημα 4.1.

Για τη σχεδίαση του συστήματος κύριας μνήμης του παραδείγματός μας, πρέπει αρχικά να συνδέσουμε τις γραμμές δεδομένων όλων των ψηφίδων ($D_7$ – $D_0$) με τις αντίστοιχες γραμμές τού επεξεργαστή. Οι ψηφίδες ROM συνδέονται ανά ζεύγη, έτσι ώστε να καλύπτεται όλο το μήκος λέξης του υπολογιστή.

Οι γραμμές διευθύνσεως των ψηφίδων μνήμης συνδέονται επίσης με τις αντίστοιχες γραμμές διευθύνσεως του υπολογιστή. Οι ψηφίδες RAM που έχουν 256 λέξεις, διαθέτουν 8 γραμμές διευθύνσεως, οι οποίες συνδέονται με τις γραμμές $A_0$ έως $A_7$ του υπολογιστή. Αντίστοιχα, οι ψηφίδες ROM που έχουν 512 λέξεις, διαθέτουν 9 γραμμές διευθύνσεως, οι οποίες συνδέονται με τις γραμμές $A_0$ έως $A_8$ του υπολογιστή. Επίσης, όλες οι ψηφίδες RAM συνδέονται με τα σήματα RD και WR του υπολογιστή, για τον προσδιορισμό της λειτουργίας που πρέπει να εκτελεστεί κάθε φορά.




Επειδή κάθε διεύθυνση που δημιουργείται από τον επεξεργαστή (και μεταφέρεται μέσω των γραμμών διευθύνσεως) αναφέρεται σε συγκεκριμένη ψηφίδα (ή ζεύγος ψηφίδων), σύμφωνα με τον Χάρτη Διευθύνσεων, είναι απαραίτητος ένας επιπλέον μηχανισμός (CS), ο οποίος να επιλέγει και να ενεργοποιεί κάθε φορά την κατάλληλη ψηφίδα, ώστε αυτή μόνο να μεταφέρει δεδομένα πάνω στις γραμμές δεδομένων του επεξεργαστή.

Ο μηχανισμός CS θα επιλέγει την ψηφίδα $RAM_3$ μόνο όταν $A_{10}$=0, $A_9$=1 και $A_8$=1.

| $A_{10}$ | $A_9$ | $A_8$ | Ψηφίδα |
|---|---|---|---|
| 0 | 0 | 0 | $RAM_0$ |
| 0 | 0 | 1 | $RAM_1$ |
| 0 | 1 | 0 | $RAM_2$ |
| 0 | 1 | 1 | $RAM_3$ |
| 1 | 0 | 0 ή 1 | $ROM_0$ και $ROM_1$ |
| 1 | 1 | 0 ή 1 | $ROM_2$ και $ROM_3$ |

Μπορούμε να δούμε από τον Χάρτη Διευθύνσεων ότι οι τιμές των ψηφίων $A_{10}$, $A_9$ και $A_8$ μπορούν να χρησιμοποιηθούν για την ενεργοποίηση των ψηφίδων μνήμης, σύμφωνα με τον πίνακα.

## Διαφύλλωση μνήμης

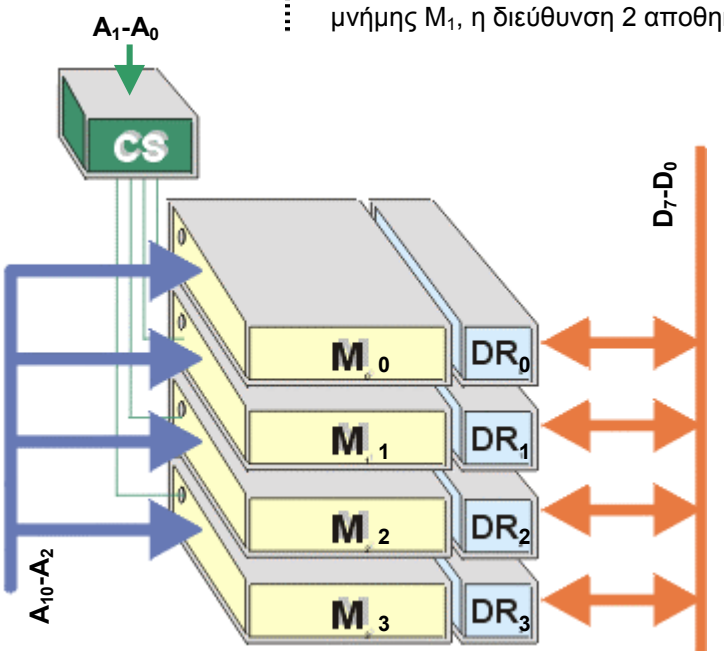
Στο παράδειγμα της προηγούμενης παραγράφου, είδαμε ότι οι γειτονικές διευθύνσεις μνήμης αποθηκεύονται στην ίδια ψηφίδα. Μπορούμε όμως να αντιστοιχίσουμε γειτονικές διευθύνσεις μνήμης σε διαφορετικές ψηφίδες με τέτοιο τρόπο, ώστε, υπό ορισμένες προϋποθέσεις, να μειωθεί ο μέσος χρόνος που απαιτείται για μια αναφορά στη μνήμη.

Η τεχνική κατά την οποία η απεικόνιση γειτονικών θέσεων του χώρου μνήμης γίνεται σε διαφορετικές ψηφίδες, αναφέρεται ως *διαφύλλωση μνήμης* (memory interleaving).

Θέλουμε να συνδυάσουμε 4 ψηφίδες μνήμης ($M_0$, $M_1$, $M_2$ και $M_3$), διαστάσεων 512×8 bits η κάθε μια, για τη δημιουργία ενός συστήματος μνήμης 2K (=2048) λέξεων.

Κάθε μια από τις ψηφίδες M τοποθετεί σε ένα εσωτερικό της καταχωρητή, τον *καταχωρητή δεδομένων εξόδου* (Data Register, DR) το περιεχόμενο της λέξης που υποδεικνύεται από την τιμή των γραμμών διευθύνσεών της. Το περιεχόμενο του καταχωρητή DR μεταφέρεται στις γραμμές δεδομένων του συστήματος μόνον όταν ενεργοποιηθεί η αντίστοιχη είσοδος επιλογής CS.

Με τη συνδεσμολογία του σχήματος, η διεύθυνση 0 αποθηκεύεται στη λέξη 0 της μνήμης $M_0$, η διεύθυνση 1 αποθηκεύεται στη λέξη 0 της μνήμης $M_1$, η διεύθυνση 2 αποθηκεύεται στη λέξη 0 της μνήμης $M_2$ και η διεύθυνση 3 αποθηκεύεται στη λέξη 0 της μνήμης $M_3$. Ανάλογα, προκύπτει ότι διεύθυνση 4 αποθηκεύεται στη θέση 1 της μνήμης $M_0$ κ.ο.κ.




Όταν γίνεται ανάγνωση μιας λέξης από τη μνήμη, ταυτόχρονα διαβάζονται και οι άλλες τρεις γειτονικές της λέξεις, από τις άλλες τρεις ψηφίδες.

Για παράδειγμα, όταν ζητηθεί η ανάγνωση από τη διεύθυνση 5 (**00000000101**), αυτή η λέξη θα δοθεί από τη μνήμη $M_1$, μέσω του $DR_1$.Ταυτόχρονα όμως, θα διαβαστούν και οι γειτονικές της διευθύνσεις 4, 6 και 7. Τα περιεχόμενα των διευθύνσεων αυτών θα παραμείνουν στους καταχωρητές $DR_0$, $DR_2$ και $DR_3$,

αντίστοιχα. Έτσι, εάν τύχει η επόμενη αναφορά μνήμης να είναι για ανάγνωση από κάποια από τις διευθύνσεις 4, 5, 6 ή 7, τότε αυτή η λέξη διαβάζεται απευθείας από τον αντίστοιχο καταχωρητή DR, χωρίς την καθυστέρηση που απαιτείται για την μεταφορά της λέξης από το εσωτερικό της μνήμης στον καταχωρητή DR. Δεδομένου ότι οι περισσότερες αναφορές στη μνήμη ακολουθούν την ιδιότητα της τοπικότητας της αναφοράς, και επειδή η πλειοψηφία των αναφορών στη μνήμη γίνεται για ανάγνωση δεδομένων, παρατηρείται σημαντική βελτίωση του μέσου χρόνου προσπέλασης της μνήμης με τη χρήση της διαφύλλωσης.

Στην περίπτωση του παραδείγματός μας έχουμε *διαφύλλωση 4 δρόμων* (4-way interleaving), επειδή γίνεται ταυτόχρονη ανάκτηση 4 γειτονικών θέσεων. Το πλήθος των δρόμων είναι κάποια δύναμη του δύο· όσο μεγαλύτερο το πλήθος των δρόμων, τόσο μεγαλύτερη η πιθανότητα για ταχεία ανάγνωση μιας λέξεως από τη μνήμη.

Γενικά λοιπόν, ένα σύστημα μνήμης με *διαφύλλωση κ-δρόμων* (k-way interleaving), όπου $\kappa=2^m$, αποτελείται από κ ψηφίδες μνήμης. Η επιλογή της ψηφίδας γίνεται από τα m δεξιότερα ψηφία της διεύθυνσης, και η εσωτερική διεύθυνση της ψηφίδας δίνεται από τα υπόλοιπα bits της διεύθυνσης μνήμης. Με τον τρόπο αυτό, οι κ συνεχόμενες θέσεις μνήμης αποθηκεύονται σε κ διαφορετικές ψηφίδες. Όταν γίνεται ανάγνωση για οποιαδήποτε λέξη της μνήμης, τότε μεταφέρονται στους εσωτερικούς καταχωρητές (DR) των ψηφίδων της μνήμης οι υπόλοιπες κ-1 γειτονικές λέξεις. Έτσι, αν στη συνέχεια διαβαστεί μια από αυτές, κάτι που γίνεται συχνά, θα είναι αμέσως διαθέσιμη.

**Ανακεφαλαίωση**

*Για να σχεδιάσουμε ένα σύστημα κύριας μνήμης για ένα συγκεκριμένο επεξεργαστή από δεδομένους τύπους ψηφίδων, πρώτα καταρτίζουμε το Χάρτη Διευθύνσεων. Από το χάρτη αυτό βρίσκουμε τον τρόπο σύνδεσης των ψηφίδων με τις γραμμές διευθύνσεων και δεδομένων της ΚΜΕ. Αν θέλουμε να επιταχύνουμε τις προσπελάσεις σε γειτονικές θέσεις μνήμης, χρησιμοποιούμε τη μέθοδο της διαφύλλωσης της μνήμης, που κατανέμει γειτονικές θέσεις μνήμης σε διαφορετικές ψηφίδες, ώστε να είναι όλες διαθέσιμες με μια ανάγνωση.*

**Γλωσσάριο όρων**

| | |
|---|---|
| *Διαστάσεις* | *Dimensions* |
| *Διαφύλλωση κ-Δρόμων* | *k-Way Interleaving* |
| *Διαφύλλωση Μνήμης* | *Memory Interleaving* |
| *Επιλογή Ψηφίδας* | *Chip Select - CS* |
| *Καταχωρητής Δεδομένων Εξόδου* | *Data Register - DR* |
| *Μνήμη Μόνο Ανάγνωσης* | *Read Only Memory - ROM* |
| *Μνήμη Ανάγνωσης-Εγγραφής* | *RAM* |
| *Χάρτης Διευθύνσεων* | *Address Map* |

## Ερωτήσεις

? Τι παριστάνουν οι διαστάσεις μιας ψηφίδας μνήμης;

? Πώς καταρτίζουμε το χάρτη διευθύνσεων για ένα σύστημα μνήμης;

? Ποια είναι η λειτουργία της επιλογής ψηφίδας;

? Τι είναι η διαφύλλωση μνήμης;

? Τι επιτυγχάνουμε με τη διαφύλλωση μνήμης;

# Μάθημα 4.4 Λανθάνουσα Μνήμη

**Σκοπός του μαθήματος**

*Σκοπός του μαθήματος αυτού είναι να περιγράψει την οργάνωση της λανθάνουσας μνήμης και τον τρόπο που επιταχύνει την εκτέλεση των προγραμμάτων.*

**Τι θα μάθεις;**

*Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:*

- *Να περιγράφεις την οργάνωση της λανθάνουσας μνήμης*
- *Να καθορίζεις πότε αντιγράφονται δεδομένα από την κύρια στη λανθάνουσα μνήμη*
- *Να υπολογίζεις πόσο επιταχύνεται η ταχύτητα εκτέλεσης ενός προγράμματος από την ύπαρξη της λανθάνουσας μνήμης*

## Αρχές λειτουργίας της λανθάνουσας μνήμης

Η *λανθάνουσα μνήμη* (cache memory) ανήκει στα ανώτερα στρώματα της ιεραρχίας της μνήμης, επειδή χαρακτηρίζεται από τη **μεγάλη ταχύτητα προσπέλασης** αλλά και το **υψηλό κόστος κατασκευής**.

Συνήθως υλοποιείται με την τεχνολογία των στατικών μνημών (SRAM), που διαθέτουν τα δύο αυτά χαρακτηριστικά. Επίσης οι στατικές μνήμες έχουν μικρή πυκνότητα αποθήκευσης, ιδιαίτερα σε σχέση με τις δυναμικές μνήμες, που σε συνδυασμό με το υψηλό κόστος τους δεν τις καθιστούν κατάλληλες για τη δημιουργία όλου του χώρου της κύριας μνήμης.

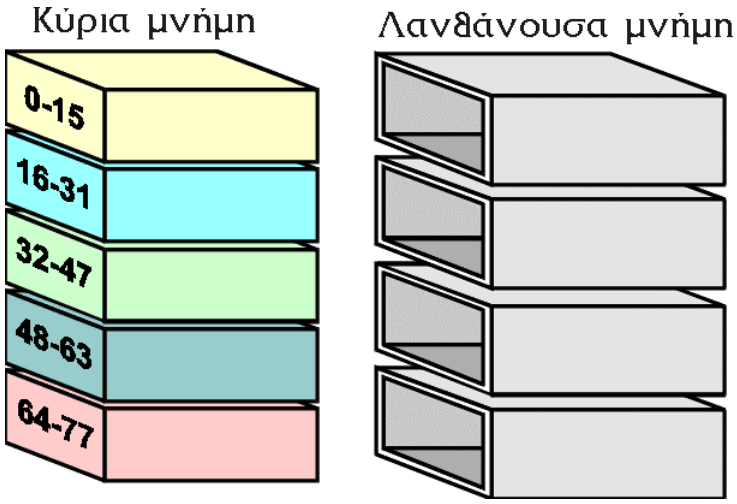
Αυτό συνεχίζει να ισχύει και σήμερα, παρά τη διαρκή μείωση των τιμών για όλους τύπους μνήμης, επειδή παράλληλα διογκώνονται συνεχώς και οι απαιτήσεις σε χώρο μνήμης όλων των σύγχρονων εφαρμογών και λειτουργικών συστημάτων. Έτσι, προτιμάται η χρήση και των δύο τεχνολογιών μνήμης, δυναμικής και στατικής, για την κύρια και λανθάνουσα μνήμη αντίστοιχα.

Όπως είδαμε και στο Μάθημα 4.2, η λειτουργία της λανθάνουσας μνήμης στηρίζεται στην αντιγραφή μέρους από τα περιεχόμενα της κύριας μνήμης. Αξιοποιώντας την τοπικότητα της αναφοράς των προγραμμάτων, οι περισσότερες αναφορές σε θέσεις μνήμης εξυπηρετούνται από τη λανθάνουσα μνήμη με πολύ μικρή καθυστέρηση.

Για τον σκοπό αυτό η λανθάνουσα μνήμη είναι οργανωμένη σε ενότητες λέξεων, που αναφέρονται και ως *γραμμές* (cache lines). Κάθε γραμμή της λανθάνουσας μνήμης έχει σταθερό μέγεθος (π.χ. 16 λέξεις) και περιλαμβάνει τα αντίγραφα λέξεων, που είναι γειτονικές στην κύρια μνήμη. Έτσι, μπορούμε να φανταστούμε ότι η κύρια μνήμη χωρίζεται σε αντίστοιχες νοητές ενότητες, μεγέθους όσο μια γραμμή της λανθάνουσας μνήμης. Ορισμένες από τις ενότητες αυτές μπορούν να αντιγραφούν στη λανθάνουσα μνήμη.

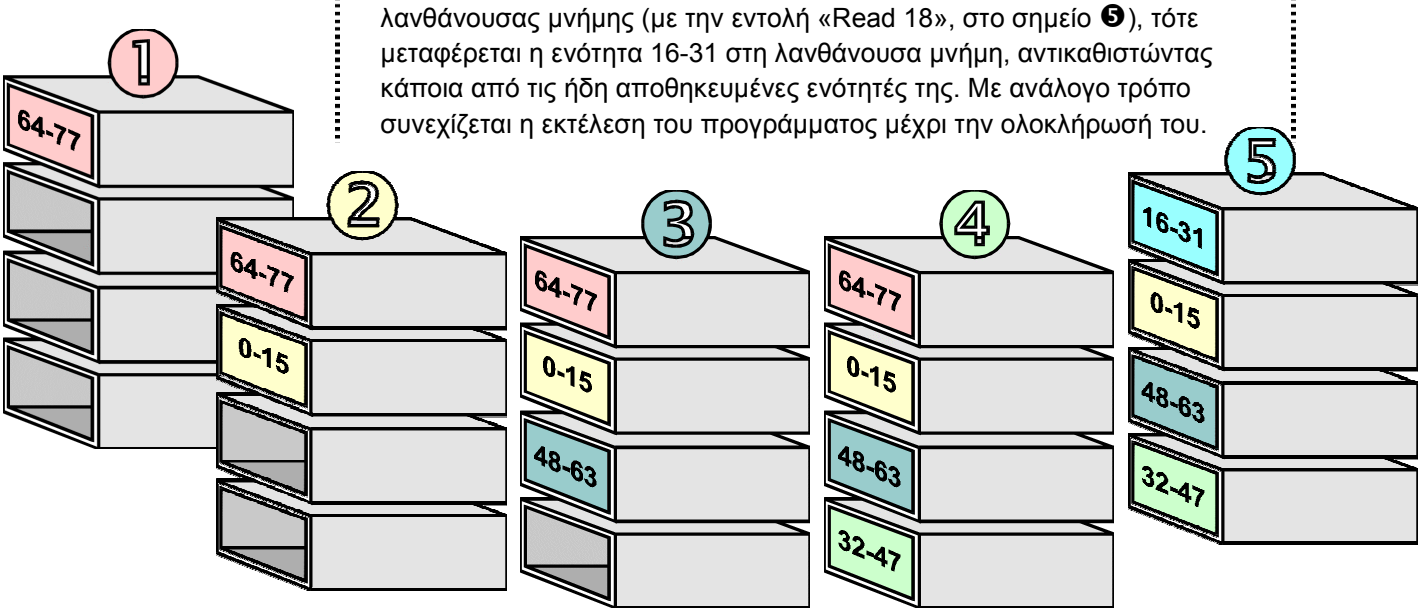
```
read 65   ①
read 66
read 8    ②
read 67
read 9
read 65
read 66
write 51  ③
read 7
read 8
read 33   ④
read 34
write 52
read 35
read 18   ⑤
read 36
```

Ας δούμε ένα σενάριο προσπελάσεων μνήμης από ένα υποθετικό πρόγραμμα. Η λανθάνουσα μνήμη αποτελείται από 64 λέξεις, χωρισμένες σε 4 γραμμές των 16 λέξεων η κάθε μια, και αρχικά είναι κενή.




Κάθε φορά που σημειώνεται *αποτυχία της λανθάνουσας μνήμης* (cache miss), δηλαδή αδυναμία να βρεθεί η ζητούμενη λέξη σε κάποια γραμμή της, μεταφέρεται ολόκληρη η αντίστοιχη ενότητα από την κύρια μνήμη στην λανθάνουσα μνήμη.

Αυτό συμβαίνει στα σημεία ❶,❷,❸ και ❹ του προγράμματος. Μετά από αυτά, όλες οι ενότητες της λανθάνουσας μνήμης είναι κατειλημμένες με περιεχόμενα της κύριας μνήμης. Όταν παρατηρηθεί η επόμενη αποτυχία λανθάνουσας μνήμης (με την εντολή «Read 18», στο σημείο ❺), τότε μεταφέρεται η ενότητα 16-31 στη λανθάνουσα μνήμη, αντικαθιστώντας κάποια από τις ήδη αποθηκευμένες ενότητές της. Με ανάλογο τρόπο συνεχίζεται η εκτέλεση του προγράμματος μέχρι την ολοκλήρωσή του.




Όλες οι αναγκαίες λειτουργίες για τη διαχείριση της λανθάνουσας μνήμης γίνονται εντελώς αυτόματα από το υλικό του επεξεργαστή. Οι λειτουργίες αυτές είναι:

- ο έλεγχος των περιεχομένων της λανθάνουσας μνήμης·
- οι μεταφορές των δεδομένων από την κύρια μνήμη·
- οι αντικαταστάσεις των δεδομένων όταν η λανθάνουσα μνήμη είναι πλήρης.

Έτσι δεν προκαλείται αξιόλογη καθυστέρηση στο πρόγραμμα που εκτελείται, γιατί κάτι τέτοιο θα εκμηδένιζε τα πλεονεκτήματα της λανθάνουσας μνήμης.

## Χρήση της λανθάνουσας μνήμης

Όταν ξεκινά μια ανάγνωση από ή μια εγγραφή στη μνήμη, γίνονται τα ακόλουθα βήματα:

- Ελέγχεται άν η ζητούμενη λέξη είναι αποθηκευμένη στη λανθάνουσα μνήμη.
- Εάν αυτό ισχύει, τότε η αναφορά στη μνήμη ολοκληρώνεται ταχύτατα από τη λανθάνουσα μνήμη, χωρίς την ανάμιξη της κύριας μνήμης.
- Εάν η ζητούμενη διεύθυνση μνήμης δεν περιλαμβάνεται στα τρέχοντα περιεχόμενα της λανθάνουσας μνήμης, τότε η προσπέλαση (ανάγνωση ή εγγραφή) γίνεται στην κύρια μνήμη. Ταυτόχρονα αντιγράφονται από την κύρια μνήμη όλες οι λέξεις της ενότητας στην οποία ανήκει η ζητούμενη λέξη. Αυτό γίνεται επειδή, σύμφωνα με την αρχή της **τοπικότητας της αναφοράς**, οι επόμενες αναφορές στη μνήμη είναι πολύ πιθανό να γίνουν σε διευθύνσεις γειτονικές με την προηγούμενη.

Με τη διαδικασία αυτή, ένα μεγάλο ποσοστό από τις αναφορές στην μνήμη εξυπηρετείται ταχύτατα μέσω της λανθάνουσας μνήμης. Αυτό το ποσοστό, που συνήθως αναφέρεται και ως *λόγος επιτυχίας* λ (hit ratio), είναι δυνατόν να ξεπεράσει το 99%, οδηγώντας έτσι σε σημαντική μείωση του χρόνου εκτέλεσης των προγραμμάτων.

> Εάν ο χρόνος προσπέλασης στην κύρια μνήμη είναι 100 ns, ο χρόνος προσπέλασης στη λανθάνουσα μνήμη είναι 10 ns, και ο λόγος επιτυχίας είναι λ=95%, τότε ο μέσος χρόνος προσπέλασης $t_\mu$ στη μνήμη είναι:
>
> $$t_\mu = \frac{95}{100} \times 10ns + \frac{5}{100} \times (100ns + 10ns) = 9{,}5ns + 5{,}5ns = 15ns,$$
>
> επειδή το 95% των προσπελάσεων γίνεται στη λανθάνουσα μνήμη με ταχύτητα 10ns, ενώ για το 5% των προσπελάσεων γίνεται αναφορά και στη λανθάνουσα μνήμη και στην κύρια μνήμη με ταχύτητα 10ns + 100ns.

Στη γενική περίπτωση, εάν θεωρήσουμε ότι $t_m$ είναι ο χρόνος προσπέλασης στην κύρια μνήμη, $t_c$ ο χρόνος προσπέλασης στη λανθάνουσα μνήμη και λ είναι ο λόγος επιτυχίας, τότε ο μέσος χρόνος προσπέλασης στη μνήμη είναι:

$$t_\mu = \frac{\lambda}{100} \times t_c + (1 - \frac{\lambda}{100}) \times (t_c + t_m)$$

Παρατηρούμε ότι όσο ο λόγος επιτυχίας λ πλησιάζει το 100, η τιμή του μέσου χρόνου προσπέλασης στη μνήμη ($t_\mu$) τείνει να εξισωθεί με τον χρόνο προσπέλασης της λανθάνουσας μνήμης ($t_c$).

## Μέθοδοι εγγραφής στη λανθάνουσα μνήμη

Σε ένα σύστημα με λανθάνουσα μνήμη η λειτουργία της εγγραφής στη μνήμη μπορεί να αντιμετωπιστεί με διαφορετικούς τρόπους[1].

> Η εντολή «Write 52», στο προηγούμενο παράδειγμα, οδηγεί στην καταγραφή μιας νέας λέξης στη διεύθυνση μνήμης 52, η οποία τυχαίνει

[1] Σημειώνουμε ότι το πλήθος των εγγραφών στη μνήμη είναι πολύ μικρότερο από αυτό των αναγνώσεων.

να είναι αποθηκευμένη και στη λανθάνουσα μνήμη (ενότητα 48-63). Τόσο τα περιεχόμενα της κύριας όσο και της λανθάνουσας μνήμης πρέπει να ενημερωθούν.

Στην περίπτωση αυτή το σύστημα μπορεί να εφαρμόσει μια από τις ακόλουθες δύο τεχνικές:

α) ***Επανεγγραφή*** (write back): Στην περίπτωση αυτή, η νέα τιμή της θέσης μνήμης 52 καταγράφεται μόνο στη λανθάνουσα μνήμη. Όταν όμως αργότερα χρειαστεί να απομακρυνθεί η ενότητα 48-63 από τη λανθάνουσα μνήμη, θα πρέπει τα περιεχόμενά της να αντιγραφούν πάλι στις αντίστοιχες θέσεις της κύριας μνήμης. Η μέθοδος αυτή έχει ως κύριο πλεονέκτημα την πολύ γρήγορη εκτέλεση των εγγραφών στη μνήμη. Απαιτεί, ωστόσο, πρόσθετο έργο για την αντιγραφή από τη λανθάνουσα μνήμη στην κύρια μνήμη[2] των ενοτήτων που αντικαθίστανται.

β) ***Διεγγραφή*** (write through): Στην περίπτωση αυτή, η νέα τιμή της θέσης μνήμης 52 καταγράφεται στη λανθάνουσα μνήμη και ταυτόχρονα, ενώ συνεχίζεται η εκτέλεση του προγράμματος με επόμενες λειτουργίες, γίνεται καταγραφή της νέας τιμής και στη διεύθυνση 52 της κύριας μνήμης. Η μέθοδος αυτή εξασφαλίζει τη μόνιμη συμφωνία ανάμεσα στα περιεχόμενα της λανθάνουσας μνήμης και της κύριας μνήμης. Έτσι, όταν μια ενότητα αντικαθίσταται στη λανθάνουσα μνήμη, δε χρειάζεται να επανεγγραφεί στην κύρια μνήμη. Μπορεί όμως να προκαλέσει καθυστέρηση στην ολοκλήρωση των εγγραφών στη μνήμη.

## Αντικατάσταση ενοτήτων στη λανθάνουσα μνήμη

Δεν είναι δύσκολο να δούμε ότι η επιλογή των ενοτήτων που θα μεταφερθούν στη λανθάνουσα μνήμη επηρεάζει σε μεγάλο βαθμό το λόγο επιτυχίας της λανθάνουσας μνήμης και συνεπώς την ταχύτητα εκτέλεσης των προγραμμάτων.

Όταν εκτελείται ένα πρόγραμμα, ενότητες από την κύρια μνήμη μεταφέρονται στη λανθάνουσα μνήμη έως ότου αυτή γεμίσει. Αφού συμβεί αυτό, όλες οι επόμενες αποτυχίες στη λανθάνουσα μνήμη αναγκαστικά οδηγούν στην αντικατάσταση μιας από τις υπάρχουσες ενότητες με αυτή στην οποία παρουσιάστηκε η αποτυχία.

Αν αντικατασταθεί (δηλαδή διαγραφεί) μια ενότητα και κατόπιν γίνει αναφορά σε κάποια λέξη της ενότητας αυτής, προκαλείται μια νέα αποτυχία λανθάνουσας μνήμης. Αυτό έχει προφανείς αρνητικές συνέπειες στην απόδοση του συστήματος μνήμης. Για το λόγο αυτό, επιδιώκεται η αντικατάσταση εκείνης της ενότητας για την οποία η πιθανότητα να γίνει αναφορά στο άμεσο μέλλον είναι ελάχιστη. Με αυτό το στόχο, και δεδομένου ότι μας είναι άγνωστο ποιες ακριβώς θα είναι οι επόμενες αναφορές μνήμης ενος προγράμματος, υπάρχουν οι ακόλουθες τακτικές αντικατάστασης:

- **Αντικατάσταση με βάση το χρόνο παραμονής** (First-In First-Out, FIFO), κατά την οποία αντικαθίσταται κάθε φορά εκείνη η ενότητα που είχε εισαχθεί παλαιότερα στη λανθάνουσα μνήμη.
- **Αντικατάσταση με βάση το χρόνο τελευταίας προσπέλασης** (Least Recently Used, LRU), κατά την οποία αντικαθίσταται κάθε φορά εκείνη η ενότητα στην οποία έχει να γίνει αναφορά για περισσότερο χρόνο.

[2] Αυτό μπορεί να αποφευχθεί μόνον αν, στην ενότητα που θα αντικατασταθεί, δεν έχει γίνει καμία εγγραφή για όσο χρόνο η ενότητα αυτή βρισκόταν στη λανθάνουσα μνήμη.

Και οι δύο αυτές μέθοδοι προσπαθούν να εκμεταλλευθούν την τοπικότητα αναφοράς των προγραμμάτων, προκειμένου να προβλέψουν την ενότητα με τη μικρότερη πιθανότητα αναφοράς. Από αυτήν την άποψη, η μέθοδος LRU είναι η πιο αποτελεσματική. Η υλοποίησή της, όμως, είναι πιο πολύπλοκη από ό,τι η FIFO, επειδή πρέπει να κρατά και να επεξεργάζεται στοιχεία για το χρόνο στον οποίο γίνεται προσπέλαση σε κάθε ενότητα της λανθάνουσας μνήμης.

## Πολυεπίπεδη οργάνωση λανθάνουσας μνήμης

Με τη διαρκή πρόοδο στον τομέα των ολοκληρωμένων κυκλωμάτων έχει γίνει δυνατή η υλοποίηση ενος μέρους της λανθάνουσας μνήμης μέσα στο κύκλωμα του ίδιου του επεξεργαστή. Αυτή η μνήμη είναι πάρα πολύ γρήγορη, επειδή η προσπέλαση γίνεται στο εσωτερικό του ίδιου ολοκληρωμένου κυκλώματος, του επεξεργαστή.

Αυτή η «εσωτερική» μνήμη, όμως, είναι σχετικά περιορισμένη σε μέγεθος· έτσι, χρησιμοποιείται και άλλη, εξωτερική μνήμη, ως λανθάνουσα μνήμη, δημιουργώντας έτσι δύο επίπεδα λανθάνουσας μνήμης:

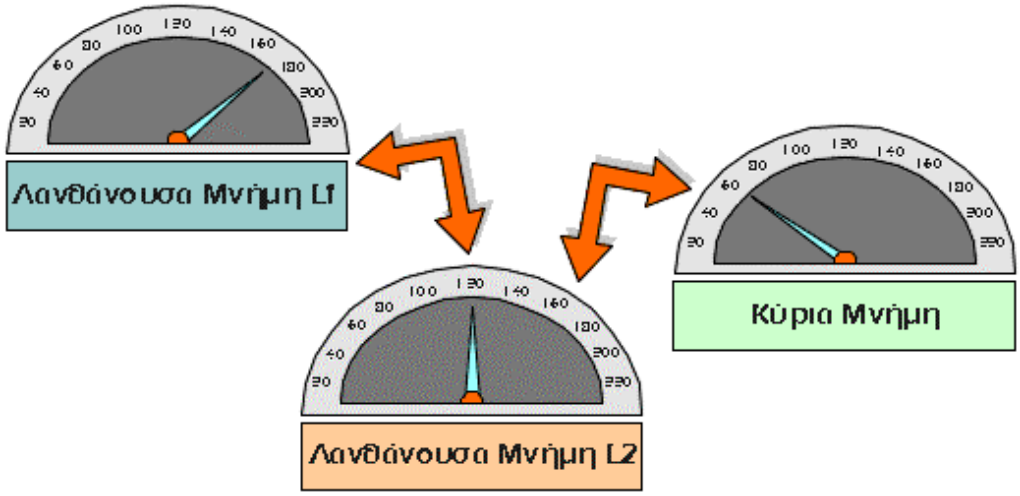
**L1** Το *πρώτο επίπεδο λανθάνουσας μνήμης* (Level 1 cache ή L1) είναι πάρα πολύ ταχύ επειδή ανήκει στον επεξεργαστή και λειτουργεί με την ίδια ταχύτητα με αυτόν.

**L2** Το *δεύτερο επίπεδο λανθάνουσας μνήμης* (Level 2 cache ή L2) έχει συνήθως περισσότερο χώρο αποθήκευσης και σχετικά μεγαλύτερο χρόνο προσπέλασης.

Το σύστημα διαχείρισης της λανθάνουσας μνήμης προσπαθεί να επιτύχει τη μέγιστη αξιοποίηση της μνήμης L1, ώστε να ελαχιστοποιείται ο χρόνος εκτέλεσης των προγραμμάτων.

Στην περίπτωση αυτή, η προσπέλαση της μνήμης γίνεται σε τρία βήματα:

- Αν η λέξη που προσπελάζεται βρίσκεται στη μνήμη L1, τότε χρησιμοποιείται απευθείας από εκεί.
- Αν δε βρεθεί στη μνήμη L1, τότε ελέγχεται μήπως βρίσκεται στη μνήμη L2. Αν ναι, χρησιμοποιείται και αντιγράφεται στη μνήμη L1.
- Αν δε βρεθεί ούτε στη μνήμη L2, τότε η λέξη προσπελάζεται στο χώρο της κύριας μνήμης και αντιγράφεται συγχρόνως στη μνήμη L2.



**Ανακεφαλαίωση**

*Η λανθάνουσα μνήμη είναι μια γρήγορη μνήμη που κρατά τα πιο πρόσφατα χρησιμοποιημένα περιεχόμενα της κύριας μνήμης. Όταν γίνεται μία αναφορά στη μνήμη, καταβάλλεται προσπάθεια να εξυπηρετηθεί αυτή από τη λανθάνουσα μνήμη, και αν αυτό δε γίνει, τότε οι αντίστοιχες λέξεις αντιγράφονται από τη λανθάνουσα στην κύρια μνήμη. Το ποσοστό των αναφορών στη μνήμη που εξυπηρετούνται από τη λανθάνουσα μνήμη ονομάζεται ποσοστό επιτυχίας.*

**Γλωσσάριο όρων**

| | |
|---|---|
| *Αντικατάσταση με Βάση το Χρόνο Παραμονής* | *First-In First-Out - FIFO* |
| *Αντικατάσταση με Βάση το Χρόνο Τελευταίας Προσπέλασης* | *Least Recently Used - LRU* |
| *Αποτυχία της Λανθάνουσας Μνήμης* | *Cache Miss* |
| *Γραμμή* | *Cacheline* |
| *Δεύτερο Επίπεδο Λανθάνουσας Μνήμης* | *Level 2 Cache - L2* |
| *Διεγγραφή* | *Write Through* |
| *Επανεγγραφή* | *Write Back* |
| *Λανθάνουσα Μνήμη* | *Cache Memory* |
| *Λόγος Επιτυχίας* | *Hit Ratio* |
| *Πρώτο Επίπεδο Λανθάνουσας Μνήμης* | *Level 1 Cache - L1* |

## Ερωτήσεις

? Ποια είναι τα κύρια χαρακτηριστικά της λανθάνουσας μνήμης;

? Με τι τύπους κυκλωμάτων συνήθως υλοποιείται η λανθάνουσα μνήμη;

? Πώς ονομάζονται τα τμήματα στα οποία χωρίζεται η λανθάνουσα μνήμη;

? Τι είναι ο λόγος επιτυχίας λ;

? Ποιοι είναι οι τρόποι εγγραφής στη λανθάνουσα μνήμη;

? Ποιες είναι οι μέθοδοι αντικατάστασης γραμμών στη λανθάνουσα μνήμη;

? Τι είναι τα επίπεδα λανθάνουσας μνήμης L1 και L2;

**Τι μάθαμε σε αυτό το κεφάλαιο**

- Η μνήμη του υπολογιστή συνήθως οργανώνεται σε ομάδες δυαδικών ψηφίων, τις λέξεις.
- Οι διάφοροι τύποι μνήμης του υπολογιστή σχηματίζουν μία ιεραρχία, στην κορυφή της οποίας βρίσκονται οι καταχωρητές της ΚΜΕ και στη βάση της οι αργές και φθηνές μνήμες.
- Η κύρια μνήμη ενός υπολογιστή μπορεί να καταρτιστεί από διάφορες ψηφίδες, διασυνδέοντάς τις με τον επεξεργαστή όπως υπαγορεύει ο χάρτης διευθύνσεων.
- Με τη διαφύλλωση μνήμης αποθηκεύουμε γειτονικές διευθύνσεις μνήμης σε διαφορετικές ψηφίδες μνήμης.
- Η λανθάνουσα μνήμη είναι μία γρήγορη μνήμη που κρατά τα πιο πρόσφατα χρησιμοποιημένα περιεχόμενα της κύριας μνήμης για να επιταχύνει την εκτέλεση των προγραμμάτων.

## Βιβλιογραφία – Πηγές


- Παπακωσταντίνου Γ., Π. Τσανάκα, Γ. Φραγκάκη, Αρχιτεκτονική Υπολογιστών, Συμμετρία, 1987.
- Patterson D., Hennesy J., Computer Organization and Design: The Hardware/Software Interface, Morgan Kaufmann, 2η έκδοση, 1998.
- Hayes J., Computer Architecture and Organization, McGraw-Hill, 1988.