qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm

|  |
| --- |
| Προηγμένη Ευρετηρίαση Δεδομένων  Τμήμα Πληροφορικής  Εαρινό Εξάμηνο, 2008 – 2009  Προγραμματιστική Εργασία  Κρητικός Απόστολος (249)  Σύβακας Σταύρος (???) |

Περιεχόμενα

[Θέμα 1ο: Αναγνώριση Ιεραρχίας Μνήμης 3](#_Toc234304133)

[Ζητούμενα 3](#_Toc234304134)

[Μέθοδος 3](#_Toc234304135)

[Μετρήσεις - Αποτελέσματα 5](#_Toc234304136)

[Θέμα 2ο: Εξομοιωτής Ιεραρχίας Μνήμης και Υλοποίηση Αλγορίθμου Ταξινόμησης 11](#_Toc234304137)

[Ζητούμενο 11](#_Toc234304138)

[Προσομοίωση του δίσκου 12](#_Toc234304139)

[Αρχιτεκτονική 12](#_Toc234304140)

[Επιδόσεις 12](#_Toc234304141)

[Εξομοιωτής του controller 13](#_Toc234304142)

[Ελεγκτής σκληρού δίσκου & Χρόνοι προσπέλασης 13](#_Toc234304143)

[Υπολογισμοί χρόνων 14](#_Toc234304144)

[Προσομοίωση της μνήμης 15](#_Toc234304145)

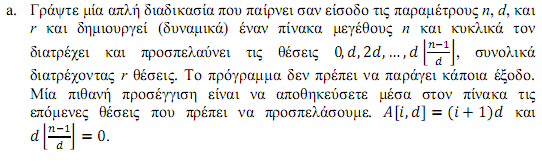
[Η Υλοποίηση 15](#_Toc234304146)

[Μετρήσεις 18](#_Toc234304147)

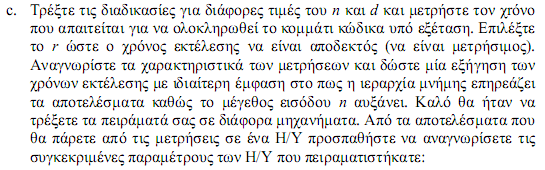
[Βιβλιογραφία 21](#_Toc234304148)

# Θέμα 1ο: Αναγνώριση Ιεραρχίας Μνήμης

## Ζητούμενα







* Μέγεθος κρυφής μνήμης (L1 και L2 αν είναι δυνατό)
* Μέγεθος γραμμής κρυφής μνήμης (L1 και L2 αν είναι δυνατό) Προηγμένη Ευρετηρίαση Δεδομένων  Σελίδα 2
* Σχετικοί χρόνοι προσπέλασης της L2 και κύριας μνήμης ως προς L1

## Μέθοδος

Για την μέτρηση της ταχύτητας της μνήμης στο πρώτο θέμα της εργασίας κάναμε εκτεταμένες μετρήσεις σε δύο υπολογιστές:

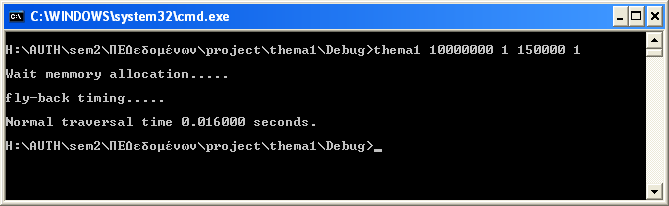
Dell:

* CPU 4-core Q6600
* Cache: 8MB
* RAM: GB

HP:

* CPU P4 1,8
* Cache: 256 KB
* RAM: 1,5 GB

Το λογισμικό που αναπτύξαμε εκτελείται με παραμέτρους από την γραμμή εντολών όπως φαίνεται στο παρακάτω σχήμα και χρησιμοποιεί δυναμική εκχώρηση μνήμης για την δημιουργία του πίνακα που εξομοιώνει την μνήμη του υπολογιστή.

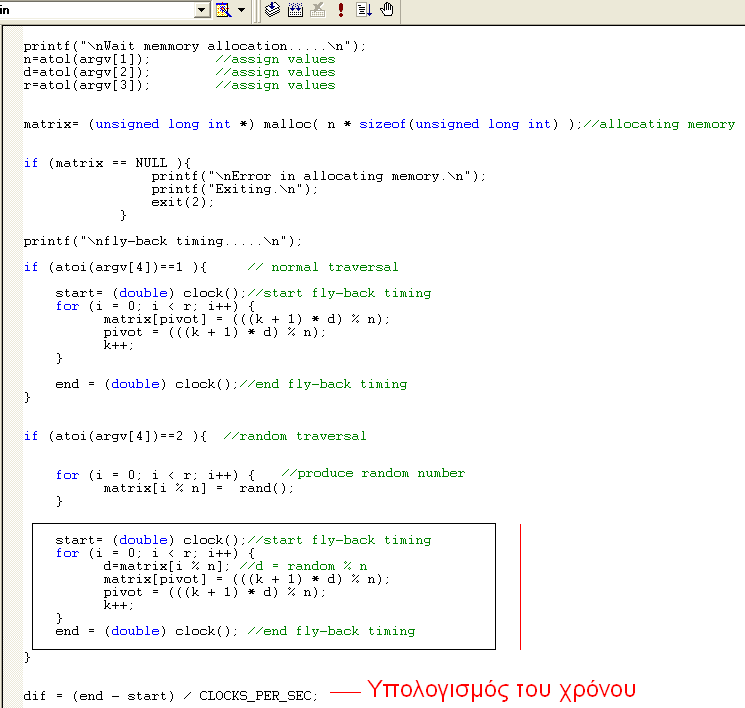


***Σχήμα 1: εκτέλεση του προγράμματος***

**Παράμετροι:**

* Η πρώτη παράμετρος είναι το μέγεθος του πίνακα (n)
* Η δεύτερη παράμετρος είναι το μέγεθος του step (d)
* Η τρίτη παράμετρος είναι το πλήθος προσπελάσεων (r)
* Η τέταρτη παράμετρος παίρνει τιμές 1 ή 2 με:
  + 1 :: Normal Traversal
  + 2 :: Random Traversal

Για την δεύτερη μέθοδο χρησιμοποιούμε τη γεννήτρια τυχαίων αριθμών της C rand() χωρίς φυσικά να συνυπολογίζουμε στο χρόνο εκτέλεσης τον χρόνο παραγωγής των τυχαίων αριθμών (Σχήμα 2)



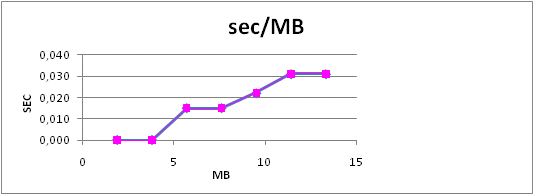
***Σχήμα 2***

## Μετρήσεις - Αποτελέσματα

Η μετρήσεις έγιναν σε δύο διαφορετικούς υπολογιστές όπως αναφέραμε παραπάνω για να ανακαλύψουμε τα επίπεδα μνήμης τους, L1 και L2 RAM και σκληρό δίσκο.

Για το πρώτο μοντέλο Ηλεκτρονικού Υπολογιστή εκτελούμε τα παρακάτω test στα οποία θα προσπαθήσουμε αρχικά να μετρήσουμε την cache του συστήματος (στο test κρατάμε σταθερό το μέγεθος του πίνακα και αλλάζουμε μόνο της προσπελάσεις, με step=1 και normal traversal)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TEST L1 & L2 |  |  |  |  |  |
| Χρόνος | Μεγ Πίνακα(n) | Step(d) | Προσπελάσεις(r) | Μνήμη σε MB Προσπέλαση | test |
| 0,000 | 400000000 | 1 | 500000 | 2 | 1 |
| 0,000 | 400000000 | 1 | 1000000 | 4 | 1 |
| 0,015 | 400000000 | 1 | 1500000 | 6 | 1 |
| 0,015 | 400000000 | 1 | 2000000 | 8 | 1 |
| 0,022 | 400000000 | 1 | 2500000 | 10 | 1 |
| 0,031 | 400000000 | 1 | 3000000 | 11 | 1 |
| 0,031 | 400000000 | 1 | 3500000 | 13 | 1 |



Από το παραπάνω γράφημα διαφαίνεται καθαρά πως έχουμε δύο επίπεδα μνήμης L1 και L2

**«Η L1 είναι 4ΜΒ Και η L2 είναι επίσης 4MB   
επομένως το σύνολο της cache memory είναι 8MB»**

Η επόμενη μέτρηση θα προσπαθήσει να δώσει απάντηση στο μέγεθος της ram του συστήματος που εξετάζουμε (κρατάμε σταθερό το μέγεθος του πίνακα και αλλάζουμε μόνο της προσπελάσεις, με step=1 και normal traversal) .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TEST RAM & HD |  |  |  |  |  |
| Χρόνος | Μεγ Πίνακα(n) | Step(d) | Προσπελάσεις(r) | Μνήμη σε GB Προσπέλαση | test |
| 9,856 | 500000000 | 1 | 300000000 | 1,118 | 1 |
| 10,458 | 500000000 | 1 | 500000000 | 1,863 | 1 |
| 11,480 | 500000000 | 1 | 1000000000 | 3,725 | 1 |
| 17,000 | 500000000 | 1 | 1500000000 | 5,588 | 1 |
| 31,450 | 500000000 | 1 | 2000000000 | 7,451 | 1 |

# 

**«Από το παραπάνω γράφημα διαφαίνεται πως η μνήμη Ram του συστήματος είναι 4GB»**

Στην συνέχεια θα προσπαθήσουμε να μετρήσουμε το Buffer του συστήματος από τον δίσκο στην κύρια μνήμη Ram (στο test κρατάμε σταθερό το μέγεθος του πίνακα και της προσπελάσεις, αλλάζουμε μόνο το step με normal traversal) .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TEST BLOCK |  |  |  |  |  |
| Χρόνος | Μεγ Πίνακα(n) | Step(d) | Προσπελάσεις(r) | Μνήμη σε MB Προσπέλαση | test |
| 0,016 | 400000000 | 2 | 2000000 | 8 | 1 |
| 0,016 | 400000000 | 3 | 2000000 | 8 | 1 |
| 0,016 | 400000000 | 4 | 2000000 | 8 | 1 |
| 0,031 | 400000000 | 5 | 2000000 | 8 | 1 |
| 0,031 | 400000000 | 6 | 2000000 | 8 | 1 |
| 0,031 | 400000000 | 7 | 2000000 | 8 | 1 |
| 0,063 | 400000000 | 8 | 2000000 | 8 | 1 |
| 0,063 | 400000000 | 9 | 2000000 | 8 | 1 |
| 0,063 | 400000000 | 10 | 2000000 | 8 | 1 |

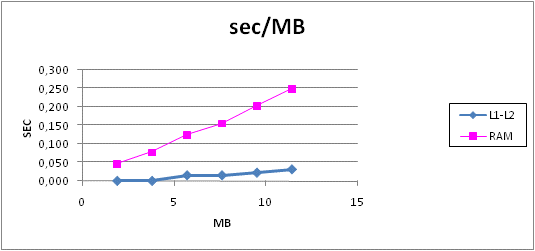
# 

Διαπιστώνουμε πως για τρία διαδοχικά step 2,3,4 δεν έχουμε αλλαγή στον χρόνο προσπέλασης ενώ προσπελαύνουμε 8Mb σε κάθε test (24MB) συμπεράνουμε λοιπόν πως το buffer δίσκου – κύριας μνήμης είναι 24MB.

Τέλος θα προσπαθήσουμε να μετρήσουμε τον μέσο χρόνο προσπέλασης σε όλη την μνήμη RAM για το λόγο αυτό θα κάνουμε αρκετά test με μέθοδο προσπέλασης τυχαία test = 2 (random traversal)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TEST RAM |  |  |  |  |  |
| Χρόνος | Μεγ Πίνακα(n) | Step(d) | Προσπελάσεις(r) | Μνήμη σε MB Προσπέλαση | test |
| 0,046 | 400000000 | 1 | 500000 | 2 | 2 |
| 0,078 | 400000000 | 1 | 1000000 | 4 | 2 |
| 0,125 | 400000000 | 1 | 1500000 | 6 | 2 |
| 0,156 | 400000000 | 1 | 2000000 | 8 | 2 |
| 0,204 | 400000000 | 1 | 2500000 | 10 | 2 |
| 0,250 | 400000000 | 1 | 3000000 | 11 | 2 |

Το παρακάτω διάγραμμα συγκρίνει του χρόνους εκτέλεσης σε L1&L2 με την RAM



**«Διαπιστώνουμε πως ο χρόνος εκτέλεσης στην L1 & L2 είναι σχεδόν γραμμικός ενώ στην Ram αυξάνει όσο αυξάνει το πλήθος των στοιχείων που επεξεργαζόμαστε»**

Ακολουθούν οι μετρήσεις που κάναμε στον 2ο μοντέλο ηλεκτρονικού υπολογιστή:

Εκτελούμε τα παρακάτω test στα οποία θα προσπαθήσουμε αρχικά να μετρήσουμε την cache του συστήματος (στο test κρατάμε σταθερό το μέγεθος του πίνακα και αλλάζουμε μόνο της προσπελάσεις, με step=1 και Normal traversal).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TEST L1 & L2(RAM) & HD |  |  |  |  |  |
| Χρόνος | Μεγ Πίνακα(n) | Step(d) | Προσπελάσεις(r) | Μνήμη σε KB Προσπέλαση | test |
| 0,000 | 10000000 | 1 | 40000 | 156 | 1 |
| 0,000 | 10000000 | 1 | 50000 | 195 | 1 |
| 0,000 | 10000000 | 1 | 60000 | 234 | 1 |
| 0,016 | 10000000 | 1 | 70000 | 273 | 1 |
| 0,016 | 10000000 | 1 | 80000 | 313 | 1 |
| 0,016 | 10000000 | 1 | 200000 | 781 | 1 |
| 0,016 | 10000000 | 1 | 400000 | 1563 | 1 |
| 0,031 | 10000000 | 1 | 500000 | 1953 | 1 |
| 0,032 | 10000000 | 1 | 600000 | 2344 | 1 |

# 

**«και σε αυτό το γράφημα ξεκάθαρα διαπιστώνουμε Cache memory (σε αυτή την περίπτωση ενός επιπέδου μόνο L1) περίπου 256ΚΒ»**

**«Ενώ η μνήμη Ram του συστήματος είναι 1,5GB»**

Στην συνέχεια θα προσπαθήσουμε να μετρήσουμε το Buffer του συστήματος από τον δίσκο στην κύρια μνήμη Ram (στο test κρατάμε σταθερό το μέγεθος του πίνακα και της προσπελάσεις, αλλάζουμε μόνο το step με Normal traversal).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TEST BLOCK |  |  |  |  |  |
| Χρόνος | Μεγ Πίνακα(n) | Step(d) | Προσπελάσεις(r) | Μνήμη σε ΚB Προσπέλαση | test |
| 0,016 | 10000000 | 1 | 400000 | 1563 | 1 |
| 0,016 | 10000000 | 5 | 400000 | 1563 | 1 |
| 0,031 | 10000000 | 10 | 400000 | 1563 | 1 |
| 0,031 | 10000000 | 15 | 400000 | 1563 | 1 |
| 0,047 | 10000000 | 20 | 400000 | 1563 | 1 |
| 0,047 | 10000000 | 25 | 400000 | 1563 | 1 |
| 0,062 | 10000000 | 30 | 400000 | 1563 | 1 |
| 0,062 | 10000000 | 35 | 400000 | 1563 | 1 |

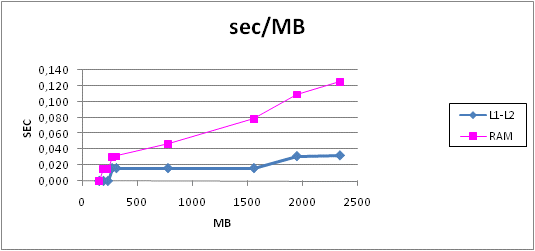
# 

**«Διαπιστώνουμε πως για πέντε διαδοχικά step 1-5 δεν έχουμε αλλαγή στον χρόνο προσπέλασης ενώ προσπελαύνουμε 1563Kb σε κάθε test (8MB) συμπεράνουμε λοιπόν πως το buffer δίσκου – κύριας μνήμης είναι 8MB»**

Τέλος και για το 2ο σύστημα μας θα προσπαθήσουμε να μετρήσουμε τον μέσο χρόνο προσπέλασης σε όλη την μνήμη RAM για το λόγο αυτό θα κάνουμε αρκετά test με μέθοδο προσπέλαση τυχαία test=2 (random traversal).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TEST RAM |  |  |  |  |  |
| Χρόνος | Μεγ Πίνακα(n) | Step(d) | Προσπελάσεις(r) | Μνήμη σε MB Προσπέλαση | test |
| 0,000 | 10000000 | 1 | 40000 | 156 | 2 |
| 0,015 | 10000000 | 1 | 50000 | 195 | 2 |
| 0,015 | 10000000 | 1 | 60000 | 234 | 2 |
| 0,030 | 10000000 | 1 | 70000 | 273 | 2 |
| 0,031 | 10000000 | 1 | 80000 | 313 | 2 |
| 0,047 | 10000000 | 1 | 200000 | 781 | 2 |
| 0,078 | 10000000 | 1 | 400000 | 1563 | 2 |
| 0,109 | 10000000 | 1 | 500000 | 1953 | 2 |
| 0,125 | 10000000 | 1 | 600000 | 2344 | 2 |

**«Το παρακάτω διάγραμμα συγκρίνει του χρόνους εκτέλεσης L1&L2 με την RAM για το 2ο σύστημα μας»**



Ο κώδικας του λογισμικού που αναπτύξαμε για τις παραπάνω μετρήσεις αλλα και το αρχείου του excel όπου δημιουργήθηκαν τα διαγράμματα βρίσκεται συνημμένο ως thema1.zip.

# Θέμα 2ο: Εξομοιωτής Ιεραρχίας Μνήμης και Υλοποίηση Αλγορίθμου Ταξινόμησης

## Ζητούμενο

1. Σε αυτή την εργασία σας ζητείται να υλοποιήσετε έναν εξομοιωτή μίας διεπίπεδης ιεραρχίας μνήμης. Η ιεραρχία αυτή θα αποτελείται από την κύρια μνήμη μεγέθους M (π.χ. Μ=512ΚΒ) και έναν δίσκο θεωρητικά απείρου μεγέθους. Η μεταφορά των δεδομένων μεταξύ κύριας μνήμης και δίσκου θα γίνεται σε μπλοκ μεγέθους Β.

Υπάρχουν πολλές παράμετροι που μπορείτε να καθορίσετε σχετικά με αυτήν την διεπίπεδη μνήμη, όπως οι χρονικές καθυστερήσεις, ύπαρξη buffer μεταξύ κύριας μνήμης και δίσκου, πλήθος δίσκων στον σκληρό δίσκο κοκ. Όλα αυτά είναι σχεδιαστικές επιλογές που αν θέλετε μπορείτε να κάνετε για τον εξομοιωτή σας. Για ευκολία ορισμού των χρονικών περιορισμών θεωρείστε ότι η προσπέλαση στην κύρια μνήμη από τον επεξεργαστή απαιτεί 1 χρονική μονάδα. Αυτό που ζητείται οπωσδήποτε είναι να εξομοιώσετε την λειτουργία του ελεγκτή του δίσκου. Αυτό σημαίνει ότι οι λογικές διευθύνσεις των μπλοκ (1,2,3,…) θα πρέπει εσείς να τις μεταφράζετε σε κινήσεις της κεφαλής υπολογίζοντας την χρονική καθυστέρηση με βάση τις τιμές που έχετε δώσει στις παραμέτρους του δίσκου που την επηρεάζουν. Κατά αυτόν τον τρόπο, η ανάγνωση ενός μπλοκ μπορεί να απαιτεί χρόνο για την αναζήτηση της κατάλληλης τροχιάς, χρόνο για την περιστροφική καθυστέρηση και τέλος χρόνο για την μεταφορά στην κύρια μνήμη. Αυτό σημαίνει ότι αν το μπλοκ είναι κοντά στην παρούσα θέση της κεφαλής ο χρόνος αυτός θα είναι μικρός, διαφορετικά θα μεγαλώνει.

Καθορίστε όλες τις παραμέτρους του συστήματος κύριας μνήμης – δίσκου που πρόκειται να εξομοιώσετε. Καλό θα είναι αυτές οι παράμετροι να είναι όσο πιο ρεαλιστικές γίνεται. Μία ιδέα για εύκολη υλοποίηση είναι να αναπαραστήσετε το δίσκο σαν ένα αρχείο στη C (ή όποια άλλη γλώσσα προγραμματισμού θέλετε) και την μνήμη σαν ένα πίνακα κατάλληλου τύπου μεγέθους M. Κάθε μπλοκ στο αρχείο θα έχει προκαθορισμένο μέγεθος ενώ θα μεταφέρει και την ακριβή θέση του μπλοκ πάνω στην επιφάνεια του δίσκου ώστε να μπορείτε να εκτελείται υπολογισμούς σχετικά με την καθυστέρηση ανάγνωσής του ή εγγραφής του. Για παράδειγμα, αν το μπλοκ είναι στην επιφάνεια 1, τροχιά 10, τομέα 3 και η κεφαλή είναι στην επιφάνεια 1, τροχιά 20, τομέα 1, εφαρμόζοντας κάποιες αριθμητικές πράξεις με βάση τις παραμέτρους που έχετε δώσει για τον σκληρό δίσκο.

1. Στον παραπάνω εξομοιωτή που κατασκευάσετε, προγραμματίστε τους αλγόριθμους ταξινόμησης συγχώνευσης και διαχωρισμού. Μετρήστε το χρόνο (με βάση τις παραμέτρους που θέσατε στον εξομοιωτή παραπάνω) για διάφορα μεγέθη εισόδου. Έπειτα μεταβάλλετε το μέγεθος της κυρίας μνήμης και δείτε πως μεταβάλλεται ο χρόνος για συγκεκριμένο μέγεθος εισόδου για τους δύο αλγόριθμους. Τέλος συγκρίνετε τους δύο αυτούς αλγόριθμους μεταξύ τους. Ποιος είναι ο καλύτερος;

## Προσομοίωση του δίσκου

**Ο δίσκος προσομοιώνεται ως αρχείο στον δίσκο με τα παρακάτω χαρακτηριστικά (τα οποία ως παράμετροι είναι όλα μεταβλητά)**

### Αρχιτεκτονική

Κεφαλές Heads :16

Τομείς Sectors :63

bytesPerSector :512

Κύλινδροι Track :16383

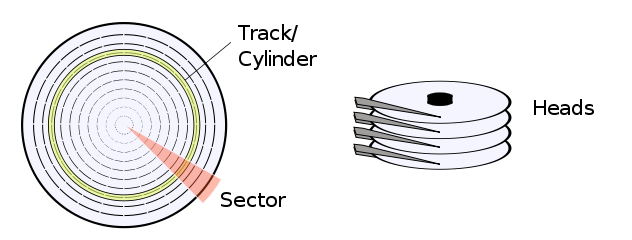
### Επιδόσεις

Ταχύτητα περιστροφής : 5400 στροφές το λεπτό

FullMovement :20ms

Command Overhead Time :0,5ms

Settle Time :0,1ms



## Εξομοιωτής του controller

O controller υλοποιήθηκε ως μία συνάρτηση η οποία καλείται μέσα από τους αλγορίθμους ταξινόμησης όποτε υπάρχει ανάγνωση / εγγραφή στο δίσκο στην οποία υπολογίζουμε το χρόνο που χρειάζεται για να εκτελεστεί η συγκεκριμένη λειτουργία (access time).

### Ελεγκτής σκληρού δίσκου & Χρόνοι προσπέλασης

Η επικοινωνία του σκληρού δίσκου με την CPU πραγματοποιείται με τη βοήθεια ηλεκτρονικής διάταξης που καλείται ελεγκτής σκληρού δίσκου (hard disk controller). Ο ελεγκτής είναι υπεύθυνος για τη λήψη δεδομένων από τη CPU και την αποθήκευσή τους στο δίσκο. Επίσης, είναι υπεύθυνος για την ανάγνωση δεδομένων από το δίσκο και τη μεταφορά αυτών στη CPU. Υπάρχουν διάφορα πρωτόκολλα επικοινωνίας μεταξύ ελεγκτή και δίσκου και CPU, ανάλογα με τη διεπαφή (interface) που χρησιμοποιείται (π.χ. USB, IDE, SCSI). Η μεταφορά των δεδομένων από το δίσκο προς τη CPU πραγματοποιείται σε τρεις φάσεις που έχουν ως εξής:

#### Φάση αναζήτησης:

Ο ελεγκτής δίσκου προσδιορίζει τη θέση όπου βρίσκονται τα δεδομένα. Στη συνέχεια, οι κεφαλές μετακινούνται επάνω από τη σωστή τροχιά. Ο χρόνος που απαιτείται για την ολοκλήρωση της φάσης αυτής καλείται **χρόνος αναζήτησης (seek time, Tseek).**

#### Φάση περιστροφικής καθυστέρησης:

Ο ελεγκτής ενεργοποιεί την αντίστοιχη κεφαλή, μόλις τα δεδομένα περάσουν κάτω από την κεφαλή. Υπενθυμίζεται ότι οι κεφαλές μετακινούνται μόνο κατά μήκος της διαμέτρου των δίσκων και οι δίσκοι περιστρέφονται με σταθερή ταχύτητα γύρω από τον άξονα περιστροφής. Ο χρόνος που απαιτείται για την ολοκλήρωση της φάσης αυτής καλείται **χρόνος περιστροφικής καθυστέρησης (rotational delay, Trot ή latency).**

#### Φάση μεταφοράς δεδομένων:

Μόλις το πρώτο byte περάσει κάτω από την κεφαλή, ξεκινά η ανάγνωση των δεδομένων από το μαγνητικό μέσο και η μεταφορά αυτών στον ελεγκτή. Ο χρόνος για την ολοκλήρωση της μεταφοράς των δεδομένων καλείται χρόνος **μεταφοράς (transfer time, Ttrans).**

#### Χρόνος ολοκλήρωσης μιας προσπέλασης στο δίσκο (Access Time, Taccess):

Προκύπτει εν γένει από το άθροισμα των προαναφερθέντων χρόνων ( Tseek + Trot + Ttrans ). Ωστόσο, στην παρούσα εργασία, ακολουθούμε ένα ελαφρώς διαφορετικό μοντέλο στο οποίο ο χρόνος Access Time προκύπτει ως ακολούθως:

***Access Time = Command Overhead Time + Seek Time + Settle Time + Latency***

Όπου:

* Seek Time = Seek Time
* Settle Time: περιλαμβάνεται στον Transfer Time
* Latency = Rotational delay

#### Command Overhead Time:

Είναι ένας επιπλέον χρόνος που το μοντέλο μας παίρνει υπόψην του και συνίσταται στο χρονικό διάστημα από τη στιγμή που ο σκληρός δίσκος λαμβάνει την εντολή να επιστρέψει κάτι στη μνήμη, μέχρι, ουσιαστικά, τα μηχανικά του μέρη να αρχίσουν να δουλεύουν προς αυτήν την κατεύθυνση, μέχρι δηλαδή να αρχίσει πραγματικά να συμβαίνει κάτι. Συνήθως είναι εξαιρετικά μικρός σε διάρκεια (τάξης ms) και παραλείπεται.

### Υπολογισμοί χρόνων

#### Υπολογισμός Seek Time:

Μία μετακίνηση από άκρη σε άκρη διαρκεί 20ms

Δηλαδή 16383 σε 20ms

1 σε 0,001ms

Έτσι υπολογίζουμε την διαφορά πάντα ( ο πρώτος κύλινδρος είναι στο εσωτερικό )

Από το μεγαλύτερο π.χ. είμαστε στον 16000 και θέλουμε να πάμε στον 14000

14000-16000=2000 \* 0,001ms = 2ms

#### Υπολογισμός χρόνου περιστροφής Latency:

Μία στροφή χρειάζεται 5,555ms ( 60 /5400 \*0,5)\*1000

Ο χρόνος περιστροφής από ένα τομέα στον επόμενο είναι 5,555/63 = 0,0881ms

Παράδειγμα: Αν η κεφαλή είναι στο τομέα 5 και θέλει να μετακινηθεί στο τομέα 3 τότε

If current > next

63-current +next=61 (63-5+3=61)

Παράδειγμα: Αν η κεφαλή είναι στο τομέα 5 και θέλει να μετακινηθεί στο τομέα 13 τότε

Else ( current < next )

next - current = 8 (13-5=8)

Έτσι εφόσον υπολογίσουμε πόσους τομείς θα μετακινηθεί η κεφαλή πολλαπλασιάζουμε \* χρόνο στο 1 παράδειγμα έχουμε 61 τομείς μετακίνηση \* 0,0001762 χρόνος μετακίνησης ανά τομέα = 0,1074 sec συνολικός χρόνος σε αυτόν πρέπει να

## Προσομοίωση της μνήμης

Η μνήμη προσομοιώνεται ως πίνακας στην μνήμη και το μέγεθος της είναι παράμετρος. Όποτε αλλάζει ο χρόνος προσπέλασης της μνήμης θεωρείται αμελητέος οπότε δεν προσμετράτε στον αλγόριθμο υπολογισμού του χρόνου ταξινόμησης .

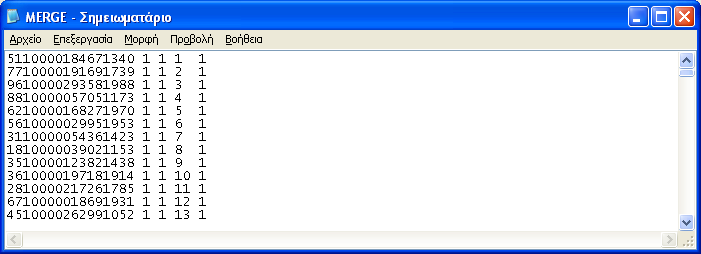
## Η Υλοποίηση

Το πρόγραμμα μας όπως αναφέραμε και παραπάνω προσομοιώνει τη μνήμη το δίσκο και φυσικά τον controller του δίσκου.

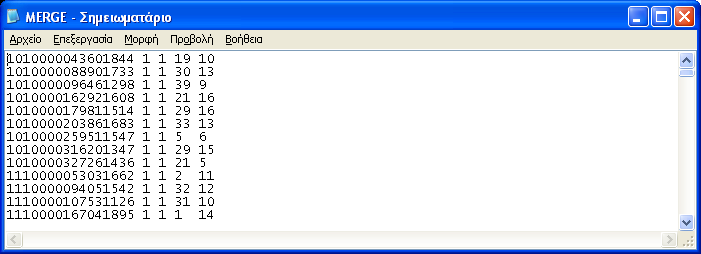
Αρχικά ζητείται από τον χρήστη να προσδιορίσει το είδος της ταξινόμησης που θα εκτελεστεί. Στο σημείο αυτό αξίζει να τονίσουμε **πως η ταξινόμηση υλοποιείται (εκτελείται) πραγματικά στο παραγόμενο αρχείο.** Στην συνέχεια ζητείται από τον χρήστη να προσδιορίσει το πλήθος των εγγραφών του αρχείου που θα δημιουργηθεί.

Το αρχείο που δημιουργείται για της ανάγκες της προσομοίωσης έχει εγγραφές μήκους 16 bits που προσομοιώνουν τα δεδομένα (block) οι οποίες είναι τυχαίες αριθμητικές τιμές στην συνέχεια του αρχείου υπάρχουν τα δεδομένα που δηλώνουν την θέση του block στον δίσκο (disk, plate, sector, track).

Το αρχείο που δημιουργείται ανάλογα με την μέθοδο ταξινόμησης έχει όνομα QUICK ή MERGE αντίστοιχα.

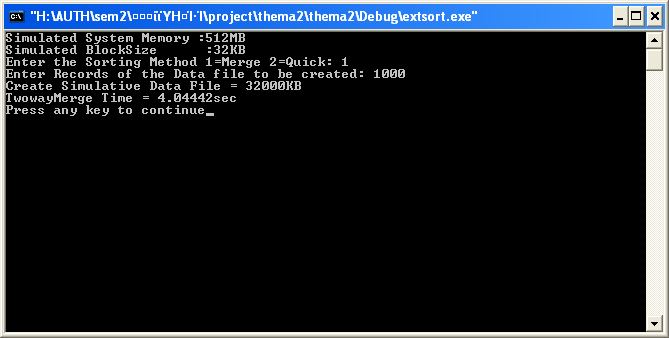


***Σχήμα 1. Δείγμα του αρχείου που παράγεται (αταξινόμητο)***

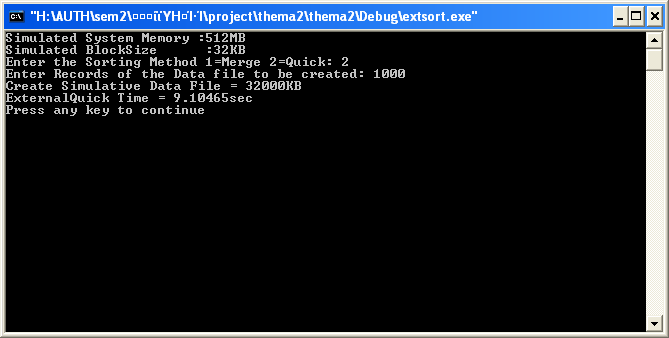


***Σχήμα 2. Δείγμα του αρχείου μετά την ταξινόμηση.***

Στο παρακάτω σχήμα βλέπουμε την εκτέλεση του προγράμματος.

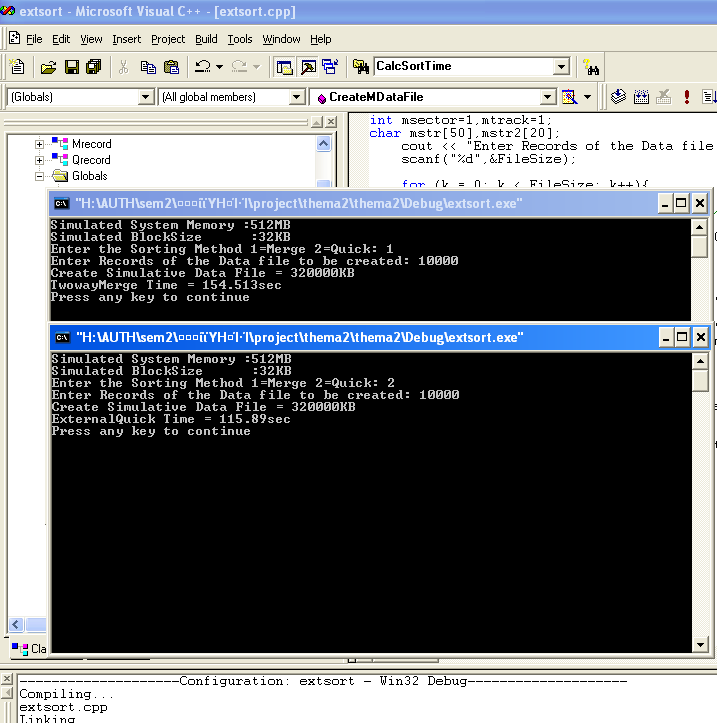
  
***Σχήμα 3. Εκτέλεση του προγράμματος***

* Έχει επιλεγεί η μέθοδος Two Way Merge Sort και το αρχείο που δημιουργήθηκε έχει 1000 εγγραφές (κάθε εγγραφή έχει μήκος 32kb αφού προσομοιώνει ένα block στον δίσκο) και 32000KB (32MB) μέγεθος.
* Η εκτέλεση της ταξινόμησης είχε 4,04442 δευτερόλεπτα διάρκεια (Σχήμα 3).
* Για αρχείο ίδιου μεγέθους η ταξινόμηση External Quick Sort είχε διάρκεια 9,10465 δευτερόλεπτα (Σχήμα 4).

  
 ***Σχήμα 4. Εκτέλεση του προγράμματος***

**«Διαπιστώνουμε πως για μικρά σε μέγεθος αρχεία ή μέθοδος Two Way Merge Sort είναι γρηγορότερη από την External Quick Sort»**

Στη συνέχεια θα δοκιμάσουμε να ταξινομήσουμε μεγαλύτερα αρχεία:



**«Για μεγαλύτερα αρχεία (320MB) βλέπουμε πως η μέθοδος External Quick Sort δίνει καλύτερα αποτελέσματα σε σχέση με την Two Way Merge Sort»**

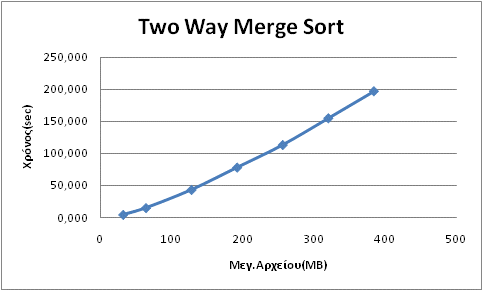
### Μετρήσεις

Στην συνέχεια θα προσπαθήσουμε να συγκρίνουμε της δύο μεθόδους ταξινόμησης.

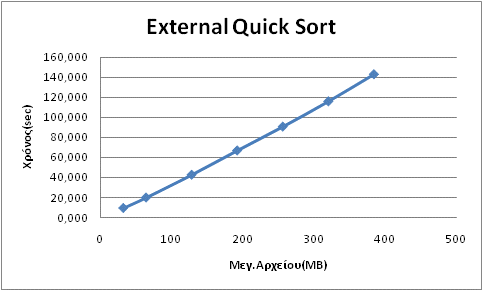
Θα πρέπει να τονίσουμε πως τα αποτελέσματα των μετρήσεων διαφέρουν από μέθοδο σε μέθοδο οι μετρήσεις αφορούν τις δικές μας υλοποιήσεις των αλγορίθμων ταξινόμησης .

* Two Way Merge Sort
* External Quick Sort

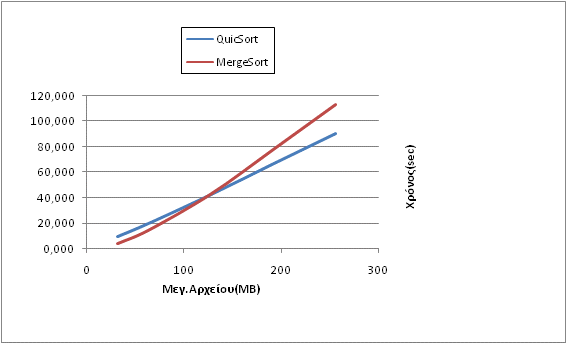
|  |  |  |
| --- | --- | --- |
| Two Way Merge Sort |  |  |
| Χρόνος(sec) | Εγγραφες | Μεγ Αρχείου(ΜΒ) |
| 4,044 | 1000 | 32 |
| 14,651 | 2000 | 64 |
| 42,939 | 4000 | 128 |
| 77,773 | 6000 | 192 |
| 112,912 | 8000 | 256 |
| 154,513 | 10000 | 320 |
| 196,510 | 12000 | 384 |



|  |  |  |
| --- | --- | --- |
| External Quick Sort |  |  |
| Χρόνος | Εγγραφες | Μεγ Αρχείου(ΜΒ) |
| 9,104 | 1000 | 32 |
| 19,605 | 2000 | 64 |
| 42,394 | 4000 | 128 |
| 66,724 | 6000 | 192 |
| 90,594 | 8000 | 256 |
| 115,890 | 10000 | 320 |
| 142,963 | 12000 | 384 |



Συγκριτικό των δύο μεθόδων ταξινόμησης:



Ο κώδικας του λογισμικού που αναπτύξαμε για τις παραπάνω μετρήσεις αλλα και το αρχείου του excel όπου δημιουργήθηκαν τα διαγράμματα επισυνάπτετε (thema2.zip).

# Βιβλιογραφία

1. Ι. Μανωλόπουλος, Α. Παπαδόπουλος, Συστήματα Βάσεων Δεδομένων. Θεωρία και πρακτική εφαρμογή, 2008.