Προγραμματιστική Εργασία σε Ιεραρχίες Μνήμης (1.5 μονάδες)

Καταληκτική Ημερομηνία Παράδοσης: 3/4/2015

Η παρακάτω προγραμματιστική εργασία είναι 1-2 ατόμων. Τα παραδοτέα της εργασίας θα είναι σύντομο κείμενο με χαρακτηριστικά, υπολογισμούς και πειραματικά αποτελέσματα, καθώς και ο κώδικας σε ηλεκτρονική μορφή. Για τον κώδικα μπορείτε να χρησιμοποιήσετε μόνο C(++).

1. Αναγνώριση Στοιχείων Δίσκου

Αυτό το ερώτημα αφορά την αναγνώριση των χαρακτηριστικών στοιχείων του δίσκου του Η/Υ σας και την καταγραφή τους.

a. Να εντοπίσετε τον σκληρό δίσκο στον οποίο είναι αποθηκευμένο το λειτουργικό σας σύστημα και να καταγράψετε τον τύπο του, το μοντέλο του και την μέγιστη χωρητικότητά του όπως αυτή αναγράφεται στο μοντέλο.
b. Με κατάλληλες εντολές του λειτουργικού συστήματος ή με κάποιο έτοιμο tool να καταγράψετε τα βασικά του χαρακτηριστικά (bytes/τομέα, τομείς/τροχιά κατά μέσο όρο ή λογικά μπλοκ, τροχιές/επιφάνεια, επιφάνειες/δίσκο, δίσκοι/σκληρό) και να υπολογίσετε την χωρητικότητά του από αυτά. Να συγκρίνετε την μέγιστη χωρητικότητα με την διαμορφωμένη χωρητικότητα.
c. Από τα χαρακτηριστικά του να υπολογίσετε τον χρόνο μέσης αναζήτησης, τον χρόνο μέσης περιστροφής, τον χρόνο μέσης μεταφοράς και τον συνολικό χρόνο προσπέλασης του δίσκου σας. Εάν χρησιμοποιήσατε κάποιο έτοιμο tool το οποίο εμφανίζει κάποιους από αυτούς τους χρόνους να κάνετε τη σύγκριση με τους δικούς σας υπολογισμούς.

2. Αναγνώριση Ιεραρχίας Μνήμης

Αυτό το ερώτημα αφορά την συγγραφή μικρών προγραμμάτων τα οποία θα χρησιμοποιήσετε για να αναγνωρίσετε βασικές παραμέτρους της ιεραρχίας μνήμης των υπολογιστών. Ο στόχος δεν είναι να κάνετε μία υλοποίηση μεγάλης έκτασης αλλά να:

- Αποκτήσετε εμπειρία στην μέτρηση της απόδοσης πραγματικών προγραμμάτων.
- Κάνετε παρατηρήσεις σχετικά με την ιεραρχία μνήμης βασιζόμενοι στον χρόνο εκτέλεσης πολύ απλών προγραμμάτων – συγκεκριμένα να δείτε αν συγκεκριμένα χαρακτηριστικά της ιεραρχίας μνήμης εμφανίζονται όταν το μέγεθος εισόδου του προγράμματος αυξάνεται.

a. Γράψτε μία απλή διαδικασία που παίρνει σαν είσοδο τις παραμέτρους *n* και *d*, δημιουργεί (δυναμικά) έναν πίνακα μεγέθους *n* και κυκλικά τον διατρέχει προσπελάζοντας τις θέσεις του διαδοχικά ανά απόσταση d, (αν φθάσει σε θέση $r > \left\lfloor \frac{n-1}{d} \right\rfloor$ τότε θα πρέπει να προσπελαστεί ο πίνακας πάλι από την αρχή κυκλικά). Το πλήθος των επαναλήψεων διάσχισης του πίνακα να είναι

τέτοιο ώστε ο συνολικός χρόνος εκτέλεσης του κώδικα να είναι αποδεκτός (μετρήσιμος). Το πρόγραμμα δεν πρέπει να παράγει κάποια έξοδο.

b. Φτιάξτε μία αντίστοιχη διαδικασία που εκτός της απλής προσπέλασης των θέσεων να κάνει και αλλαγή του περιεχομένου τους (π.χ. να αυξάνει τις τιμές τους κατά 1).
c. Τρέξτε τις δύο διαδικασίες για διάφορες τιμές του *n* και του *d* και μετρήστε τον χρόνο που απαιτείται μέχρι να ολοκληρωθεί το κομμάτι κώδικα υπό εξέταση. Αναγνωρίστε τα χαρακτηριστικά των μετρήσεων και δώστε μία εξήγηση των χρόνων εκτέλεσης με ιδιαίτερη έμφαση στο πως η ιεραρχία μνήμης επηρεάζει τα αποτελέσματα καθώς το μέγεθος εισόδου *n* αυξάνει. Από τα αποτελέσματα που θα πάρετε από τις μετρήσεις προσπαθήστε να αναγνωρίσετε τις συγκεκριμένες παραμέτρους του Η/Υ που πειραματιστήκατε:

    i. Μέγεθος κρυφής μνήμης (L1 και L2 αν είναι δυνατό)
    ii. Μέγεθος γραμμής κρυφής μνήμης (L1 και L2 αν είναι δυνατό)
    iii. Σχετικός χρόνος προσπέλασης κρυφής μνήμης και κύριας μνήμης (L1 και L2 αν είναι δυνατό)
    iv. Σχετικός χρόνος προσπέλασης του σκληρού σας δίσκου με την κύρια μνήμη

Προσοχή θα πρέπει να δοθεί στην κατανόηση της ιεραρχίας μνήμης ιδιαίτερα μεταξύ κρυφής και κύριας μνήμης ώστε να μπορείτε να καταλάβετε πως θα κάνετε τα πειράματα για να αναγνωρίστε τις ζητούμενες ποσότητες.

<u>Κάποια βοήθεια:</u>

1. Στη C μπορείτε να χρησιμοποιήσετε την time.h για να μετρήσετε το χρόνο.
2. Αν υπάρχουν πολλές εντολές προγράμματος για κάθε προσπέλαση μνήμης στο πρόγραμμα που φτιάξατε, το κόστος της μεταγωγής από την κρυφή μνήμη στην κύρια μνήμη δεν θα είναι εμφανές. Για αυτό προσπαθήστε τα προγράμματά σας να είναι όσο το δυνατό πιο μικρά και βελτιστοποιημένα. Μπορείτε να χρησιμοποιήσετε in-line συναρτήσεις, προτιμήστε καθολικές μεταβλητές, ελαχιστοποιήστε το πλήθος των if/else, κλπ. Επίσης να θέσετε των compiler να κάνει μέγιστη βελτιστοποίηση.
3. Φτιάξτε γραφήματα με τα αποτελέσματά σας (άξονας y χρόνος εκτέλεσης και άξονας x μέγεθος εισόδου *n* ή ο λογάριθμός του log*n*).
4. Τρέξτε τα πειράματά σας πολλές φορές για κάθε μέγεθος εισόδου και χρησιμοποιήστε μέσους χρόνους εκτέλεσης για τα αποτελέσματά σας. Αυτό θα πρέπει να εξομαλύνει διαταραχές λόγω εκτέλεσης άλλων διεργασιών παράλληλα στον Η/Υ σας.
5. Το πρόγραμμά σας δεν πρέπει να παράγει κάποια έξοδο στην οθόνη ή κάπου αλλού (αφού η έξοδος στον Η/Υ είναι εξαιρετικά πιο χρονοβόρα σε σχέση με μία προσπέλαση στη RAM). Όμως, θα πρέπει να προσέξετε τα προγράμματα που φτιάχνετε να είναι σωστά και να κάνουν αυτό που πρέπει.
6. Μπορείτε να χρησιμοποιήσετε και τις τεχνικές που αναπτύχθηκαν στο μάθημα.