Αγγελίδη Μαριάνθη

ΑΜ.: Π2013061

Εργασία #1 – Παράλληλος Προγραμματισμός

Εκτέλεση σε Windows

Η ανάπτυξη του προγράμματος έγινε στο Dev-C++ στα windows καθώς δεν διαθέτω υπολογιστή με linux. Σαν αποτέλεσμα, το compile και η εκτέλεση του κώδικα γίνονταν μέσα από το Dev-C++ και όχι με εντολή από command line, οπότε οι παράμετροι των διαστάσεων της εικόνας καθορίστηκαν με σταθερές:

#define N 600 //image height (rows)
#define M 800 //image width (columns)

Επίσης, λόγω των windows, στην εκδοχή της χρήσης SSE, η συνάρτηση *posix_memalign()* δεν υποστηρίζονταν. Ψάχνοντας στο internet βρήκα σε αξιόπιστο forum[1] μια ισοδύναμη της, την *_aligned_malloc()* που με κατάλληλη αλλαγή των εισόδων της χρησιμοποιείται για να καθορίσει μια δική μας *posix_memalign()* οπότε και δεν χρειάστηκε να αλλάξει ο κώδικας μας. Λόγω της *_aligned_malloc()*, στο τέλος του κώδικα, στην ελευθέρωση της μνήμης, πρέπει να χρησιμοποιηθεί η *_aligned_free()* αντί της *free(a)*.

Επιπλέον υλικό από τρίτους

Για την αρχικοποίηση της εικόνας με τυχαίους αριθμούς, με αναζήτηση στο internet βρήκα μια συνάρτηση τυχαίων αριθμών[2] την οποία τροποποίησα για να επιστρέφει το αριθμητικό δεδομένο και να μην λειτουργεί με pointer και σε τύπο float που είναι η εικόνα μας.

Ανάπτυξη – Περιγραφή εκδοχής χωρίς SSE.

Οι πίνακες P, K και newP αντιστοιχούν στην εικόνα, τον πίνακα του μετασχηματισμού και στην μετασχηματισμένη (τελική) εικόνα. Δηλώνονται σαν pointers και δεσμεύεται ο απαραίτητος για αυτούς χώρος στη μνήμη, με χρήση της malloc (όπως και στον κώδικα της

[1] http://stackoverflow.com/questions/33696092/whats-the-correct-replacement-for-posix-memalign-in-windows

[2] http://www.insomnia.gr/topic/374643-%CF%84%CF%85%CF%87%CE%B1%CE%AF%CE%BF%CE%B9-%CE%B1%CF%81%CE%B9%CE%B8%CE%BC%CE%BF%CE%AF-%CF%83%CE%B5-c/

άσκησης που βασιστήκαμε). Το μέγεθος των εικόνων είναι NxM. Ο πίνακας του μετασχηματισμού αρχικά δηλώθηκε 3x3 (όσο ο πίνακας που μας δίνονταν). Στη συνέχεια (για να γίνει ο παραλληλισμός που επιλέξαμε) αλλάχθηκε σε 3x4. Στην αρχικοποίηση του η επιπλέον στήλη περιέχει μηδενικά ώστε να μην επηρεαστεί το αποτέλεσμα του μετασχηματισμού. Στο τέλος πήρα μετρήσεις της απόδοσης (MFLOPS/sec) και με πίνακα μετασχηματισμού 3x3 και με 3x4 για να γίνει πιο σωστή σύγκριση με τον παράλληλο προγραμματισμό.

Για την υλοποίηση του μετασχηματισμού επιλέχθηκε η εκδοχή ενός 4-απλου for-loop. Τα 2 πρώτα να ορίζουν το index της γραμμής και της στήλης της εικόνας που βρίσκεται το κεντρικό pixel που συμμετέχει στο μετασχηματισμό. Για αποφυγή χρήσης σημείων εκτός πίνακα εικόνας ο μετασχηματισμός δεν εφαρμόζεται στην πρώτη και τελευταία γραμμή και στήλη (περίγραμμα εικόνας). Τα 2 εσωτερικά for χρησιμοποιούνται για το index των pixel και των τιμών του πίνακα μετασχηματισμού που συμμετέχουν στις πράξεις (πολλαπλασιασμοί και προσθέσεις). Η μεταβλητή sum κρατάει το άθροισμα των γινομένων, άρα στην ολοκλήρωση του διπλού εσωτερικού for-loop η τιμή της περνάει στον πίνακα της τελικής εικόνας και μηδενίζεται για τον επόμενο υπολογισμό του μετασχηματισμού.

Η απόδοση του συγκεκριμένου κώδικα είναι (για εικόνα διαστάσεων [NxM]=[600x800)]:

- **382 MFLOPS/sec** για πίνακα μετασχηματισμού 3x4
- **374 MFLOPS/sec** για πίνακα μετασχηματισμού 3x3.

(για τη δεύτερη εκδοχή πρέπει να αλλαχθεί το όριο του j-loop σε 4, το 4*i σε 3*i στο index του K πίνακα και η αντίστοιχη αρχικοποίηση και δέσμευση μνήμης.

Με τον πίνακα 3x4 η απόδοση αυξάνει καθώς στον υπολογισμό της (MFLOPS/sec) οι πράξεις που εκτελούνται είναι πλέον (3x4)xNxM αντί για (3x3)xNxM αντίστοιχα.

Για πιο ορθή σύγκριση, οι χρόνοι εκτέλεσης του κώδικα αντίστοιχα είναι:

- **0.030026sec**
- **0.024022sec**

Δηλαδή, στην περίπτωση του μεγαλύτερου πίνακα μετασχηματισμού (3x4) απαιτείται περισσότερος χρόνος λόγω μιας επιπλέον επανάληψης στο εσωτερικό for-loop.

Πριν από τη συγκεκριμένη εκδοχή κώδικα υλοποιήθηκαν και άλλοι κώδικες για την πράξη του μετασχηματισμού, με λιγότερα for-loop και πιο απλές εκφράσεις στα indexes καθώς οι πίνακες μας είναι μονοδιάστατοι. Όμως η επιλογή παραλληλισμού που κάναμε μας καθόρισε τη δομή του κώδικα.

Ανάπτυξη – Περιγραφή εκδοχής με SSE.

Όπως και στην αρχική άσκηση στην οποία βασιστήκαμε, η δέσμευση μνήμης για τους πίνακες γίνεται με ευθυγράμμιση στα 16 bytes. Πλέον δηλώνεται και η μεταβλητή sum ως πίνακας καθώς κρατάει τα 4 αθροίσματα του block δεδομένων __m128 που υπολογίζουν οι SSE εντολές.

Η λογική παραλληλισμού που ακολούθησα ήταν να εντοπίσω 4-αδες δεδομένων των πινάκων μας που χρησιμοποιούνται διαδοχικά στις πράξεις του μετασχηματισμού. Η αρχική ιδέα μου για σπάσιμο της εικόνας σε μπλοκ 4x4 ή έστω τετράδων ολόκληρων γραμμών της, δεν είχε το επιθυμητό αποτέλεσμα καθώς η σειρά χρήσης των δεδομένων δεν ακολουθούσε αυτή την μπλοκ αυτών. Έτσι επιλέχθηκε η λύση του παραλληλισμού με βάση το πλάτος του μπλοκ των δεδομένων που χρησιμοποιούνται λόγω του πίνακα μετασχηματισμού σε κάθε υπολογισμό ενός νέου pixel στην τελική εικόνα. Έτσι επέκτεινα το πλάτος του πίνακα μετασχηματισμό από 3 σε 4 ώστε να συμβαδίζει με την τετράδα πράξεων που λειτουργούν οι SSE εντολές. Η χρήση μηδενικών στην επιπλέον γραμμή του πίνακα μετασχηματισμού έχει σαν αποτέλεσμα να μην επηρεάζεται το τελικό αποτέλεσμα. Βέβαια το ¼ των πράξεων που γίνονται είναι περιττές.

Για τους πίνακές μας χρησιμοποιούνται οι SSE pointers (τύπου__m128), *va,*vb,*vc, *vsum για τους πίνακες P, K, newP και sum αντίστοιχα. Η αρχικοποίηση των 2 τελευταίων γίνεται εσωτερικά των 2 εξωτερικών for-loops του κώδικα του μετασχηματισμού, γιατί αυτοί αλλάζουν τιμή με την ολοκλήρωση κάθε πλήρους εφαρμογής μιας πράξης μετασχηματισμού και τον υπολογισμό της τιμής του νέου pixel. Αντίθετα, η αρχικοποίηση των va και vb γίνεται εσωτερικά του εσωτερικού for-loop του κώδικα του μετασχηματισμού, μόνο που με τη χρήση των SSE εντολών και του παραλληλισμού οι 4 πράξεις ανά γραμμή του πίνακα μετασχηματισμού (και των αντίστοιχων τιμών των pixels της αρχικής εικόνας) πλέον αυτό αποτελείτε μόνο από ένα for-loop (για κάθε γραμμή του πίνακα μετασχηματισμού).

Στις αρχικοποιήσεις των παραπάνω δεικτών χρησιμοποιήθηκαν οι εκφράσεις του index των πινάκων που είχαμε υπολογίσει στην εκδοχή χωρίς τις SSE εντολές, μόνο που χρειάστηκε στην διάσταση της στήλης να κάνουμε ευθυγράμμιση σε πολλαπλάσια του 4, ώστε να συμπίπτει η φόρτωση του πίνακα με την ευθυγράμμιση ανά 16 byte που έχουμε εφαρμόσει και στη μνήμη.

Η ταχύτητα του συγκεκριμένου κώδικα είναι (για εικόνα διαστάσεων [NxM]=[600x800)]:

- **717 MFLOPS/sec** για πίνακα μετασχηματισμού 3x4 (άρα (3x4)xNxM πράξεις).

Συγκρίνοντας με την αντίστοιχη εκδοχή του γραμμικού κώδικα βλέπουμε ότι πετύχαμε **x1.87**αύξηση της ταχύτητας ή **87%**