**Παράλληλος Προγραμματισμός 2019**
**Προγραμματιστική Εργασία 2019 #1**


**Ονοματεπώνυμο:** Σπύρος Βέργης
**ΑΜ:** 2015103

## Συνοπτική Περιγραφή

matmul-normal.c

Στον κώδικα αυτό πραγματοποιείται ένας πολλαπλασιασμός 2 πινάκων. Το αποτέλεσμα αποθηκεύεται σε έναν άλλο πίνακα. Η διαδικασία γίνεται με τον τυπικό αλγοριθμικό τρόπο.

Οι τρεις πίνακες που δημιουργούνται είναι τύπου float. Για την δημιουργία τους χρησιμοποιούμε την εντολή malloc. Στην συνέχεια αρχικοποιούμε τους πίνακες.

Για την εκτέλεση του πολλαπλασιασμού δημιουργούμε τρεις for-loops για την προσπέλαση κάθε γραμμής και στήλης του κάθε πίνακα. Σε κάθε κελί του τρίτου πίνακα τοποθετούμε το άθροισμα των γινομένων των γραμμών του πρώτου πίνακα με των στηλών του δεύτερου πίνακα. Για την καταμέτρηση της απόδοσης του κώδικά μας κρατάμε την διαφορά του timestamp πριν τον πολλαπλασιασμό και του timestamp μετά τον πολλαπλασιασμό.

Τέλος απελευθερώνουμε την δεσμευμένη μνήμη στην οποία είχαμε τους πίνακες.

matmul-sse.c

Η βασική ιδέα του κώδικα παραμένει ίδια με εκείνη του matmul-normal.c. Η διαφορά είναι στην διαδικασία του πολλαπλασιασμού των πινάκων αφού σε αυτό τον κώδικα ο πολλαπλασιασμός γίνεται μέσω του SSE. Για την δέσμευση της μνήμης των πινάκων χρησιμοποιούμε την εντολή posix_memalign η οποία είναι παρόμοια της malloc μόνο που επιτρέπει πιο αυστηρή δέσμευση θέσεων. Με την χρήση των μεταβλητών _m128 μπορούμε σε πολύ low-level να πειράξουμε τις καταχωρήσεις των 128 bits.

Στις τρεις for-loop αυτή τη φορά αντί να κάνουμε τον πολλαπλασιασμό των πινάκων με τον κλασικό τρόπο, χρησιμοποιούμε την εντολή _mm_mul_ps η οποία παίρνει σαν ορίσματα δύο μεταβλητές τύπου _m128. Ουσιαστικά παίρνει δύο πίνακες από 16 bytes και τους πολλαπλασιάζει. Έτσι μπορούμε να πολλαπλασιάσουμε ταυτόχρονα 4 float τιμές του ενός πίνακα με 4 float τιμές του άλλου. Αυτό έχει ως αποτέλεσμα το πλήθος των επαναλήψεων να πέσει από N σε N/4.

Στην συνέχεια συγκρίνουμε πάλι την διαφορά του timestamp πριν τον πολλαπλασιασμό και του timestamp μετά τον πολλαπλασιασμό. Τέλος απελευθερώνουμε την μνήμη που δεσμεύσαμε.

## Αποτελέσματα

| | normal | sse | normal | sse |
|---|---|---|---|---|
| | time | time | mflops | mflops |
| 4 | 0.000000 | 0.000000 | infinite | infinite |
| 40 | 0.000076 | 0.000043 | 841.490479 | 1491.308105 |
| 400 | 0.072981 | 0.066552 | 876.939148 | 961.655151 |
| 4000 | 80.805590 | 73.989295 | 792.024414 | 864.989990 |

Η απόδοση του κώδικα σχετίζεται άμεσα με την υπολογιστική ισχή του επεξεργαστή. Ανάλογα με τις δυνατότητες του υπολογιστή στον οποίο τρέχει το πρόγραμμα θα παρατηρήσουμε και τα αντίστοιχα αποτελέσματα. Αυτό σχετίζεται με το γεγονός ότι μέσω του κώδικα χειριζόμαστε πόρους σε πολύ χαμηλό επίπεδο. Με την μείωση όμως

των επαναλήψεων που εκτελούνται στις for-loops μπορούμε και μειώνουμε τον χρόνο εκτέλεσης του προγράμματός μας αφού πλέον αντί για N επαναλήψεις η εσωτερική for-loop εκτελεί N/4 επαναλήψεις.