Παράλληλος Προγραμματισμός

Προγραμματιστική Εργασία 1

Αλέξανδρος Ζερβόπουλος
ΑΜ: Π2015111

1 Περιγραφή Παραλλαγών Κώδικα

Το ζητούμενο έχει υλοποιηθεί με 3 διαφορετικές παραλλαγές. Η πρώτη είναι η απλή υλοποίηση χωρίς εντολές SSE2, η δεύτερη είναι παραλλαγή της πρώτης με προσθήκη loop unrolling, όπου το dot product γίνεται ανά 4 στοιχεία, ενώ η τρίτη χρησιμοποιεί εντολές SSE2, ώστε να υπολογίσει το dot product πάλι ανά 4 στοιχεία και στο τέλος αποθηκεύεται για κάθε στοιχείο του τρίτου πίνακα το άθροισμα μίας τελικής τετράδας.

Αναλυτικότερα, η πρώτη παραλλαγή χρησιμοποιεί 3 διαφορετικά for loops. Το πρώτο loop αλλάζει τη γραμμή του πρώτου πίνακα που χρησιμοποιείται για το dot product με τον δεύτερο και επαναφέρει το δείκτη που δείχνει σε αυτόν (το δεύτερο) στην αρχή του. Το δεύτερο loop επαναφέρει το δείκτη που δείχνει στη γραμμή του πρώτου πίνακα στην κατάλληλη θέση, που αναλαμβάνει το πρώτο loop, και αφού υπολογιστεί το dot product αυτής με την αντίστοιχη στήλη, αποθηκεύεται το αποτέλεσμα στην κατάλληλη θέση του τρίτου πίνακα.

Στη δεύτερη παραλλαγή, ακολουθείται η ίδια λογική, με εξαίρεση το τρίτο loop, στο οποίο υπολογίζεται το dot product κάνοντας unroll το εσωτερικό loop, ώστε να υπολογίζεται άνα 4 στοιχεία των πινάκτων.

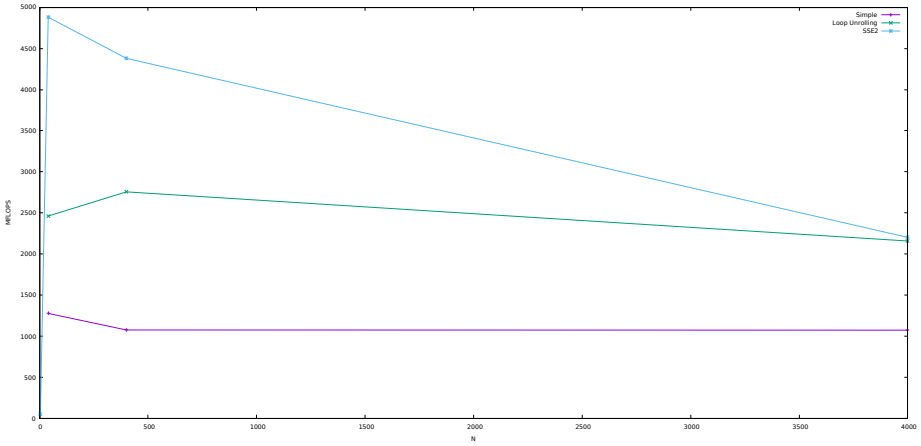
Στην τρίτη και τελευταία παραλλαγή, χρησιμοποιούνται εντολές SSE2 για τον υπολογισμό του dot product ανά 4 στοιχεία. Πριν ξεκινήσει το εσωτερικό loop αρχικοποιείται μία τετράδα με 0 και για κάθε επανάληψη αυτού, προστίθεται σε αυτή το αποτέλεσμα του dot product των 4 νέων στοιχείων, άρα καταλήγουμε σε μία τελική τετράδα την οποία πρέπει να αθροίσουμε και να αποθηκεύσουμε στο αντίστοιχο στοιχείο του τρίτου πίνακα. Αυτό μπορεί να γίνει είτε κάνοντας την τελική αυτή τετράδα να δείχνει σε κάποιον πίνακα από float με 4 θέσεις, του οποίου μπορούμε εύκολα να υπολογίσουμε το άθροισμα, είτε χρησιμοποιώντας την SSE3 εντολή hadd.

2 Αποτελέσματα

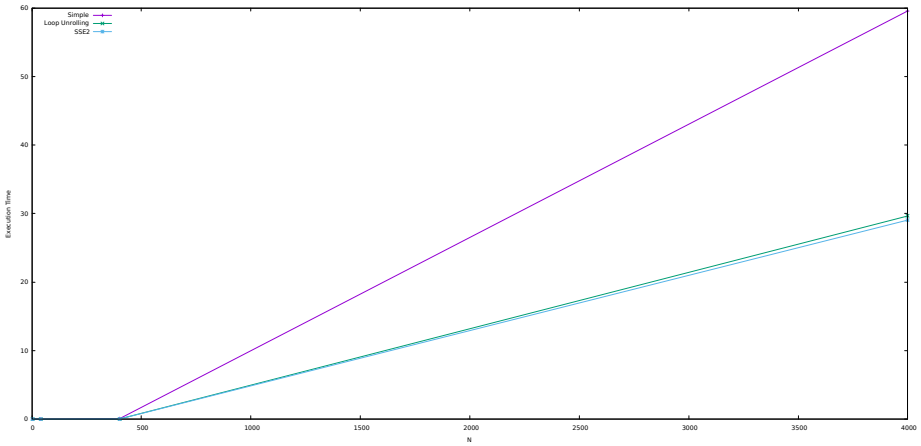
Στον Πίνακα 1 παρατίθενται αναλυτικά τα αποτελέσματα για κάθε πείραμα, ενώ στο Σχήμα 1 και Σχήμα 2 παρουσιάζονται γραφικά.

	Απλή Υλοποίηση		Loop Unrolling		SSE2	
N	Χρόνος	MFLOPS	Χρόνος	MFLOPS	Χρόνος	MFLOPS
4	0.000000	$+\infty$	0.000000	$+\infty$	0.000001	53.687091
40	0.000050	1278.264076	0.000026	2462.710606	0.000013	4880.644655
100	0.000859	1164.114349	0.000373	2680.066454	0.000186	5377.312821
200	0.006970	1147.787918	0.003146	2542.965669	0.001657	4827.975827
400	0.059443	1076.661731	0.023218	2756.463649	0.014612	4379.973828
800	0.476881	1073.643049	0.188712	2713.127275	0.117231	4367.449488
1600	3.812118	1074.468299	1.614451	2537.085470	1.323678	3094.408118
3200	31.244524	1048.759776	14.295172	2292.242448	14.012390	2338.501832
4000	59.563968	1074.475093	29.646915	2158.740634	29.050265	2203.078004

Πίνακας 1: Χρόνος και MFLOPS για κάθε υλοποίηση



Σχήμα 1: MFLOPS ανά N για κάθε υλοποίηση



Σχήμα 2: Χρόνος εκτέλεσης ανά N για κάθε υλοποίηση

3 Επεξήγηση Αποτελεσμάτων

Είναι ξεκάθαρο ότι η απλή υλοποίηση πολλαπλασιασμού πινάκων είναι η πιο αργή, και μάλιστα η ταχύτητα εκτέλεσης των πράξεων παραμένει αρκετά σταθερή για τα διάφορα N. Αυτό υποδηλώνει ότι η εκτέλεση του δεν περιορίζεται τόσο από την κρυφή μνήμη, αλλά κυρίως από την επεξεργαστική ισχύ που απαιτείται για την εκτέλεση των πράξεων.

Αντίθετα, οι άλλες δύο παραλλαγές φαίνεται να διαφέρουν για σχετικά μικρά N, με την SSE2 παραλλαγή να έχει αρκετά υψηλότερη ταχύτητα εκτέλεσης των πράξεων από αυτή με το loop unrolling. Παρ΄ όλ΄ αυτά, φαίνεται να συγκλίνουν καθώς το N αυξάνει, το οποίο πιθανώς αποδίδεται στην ταχύτητα προσπέλασης των δεδομένων, που είναι περιορισμένη από την κρυφή μνήμη. Αν η υπόθεση αυτή ισχύει, οδηγούμαστε στο συμπέρασμα ότι αν η ταχύτητα εκτέλεσης των αριθμητικών πράξεων είναι επαρκής, καθώς αυξάνεται το N, η αναλογία σε σχέση με το συνολικό χρόνο εκτέλεσης είναι ολοένα και μικρότερη, με την ταχύτητα προσπέλασης των δεδομένων να παίζει το σημαντικότερο ρόλο και τελικά να περιορίζει σε μεγάλο βαθμό την ταχύτητα εκτέλεσης του προγράμματος.

Η τάση αυτή, να συγκλίνουν οι αποδόσεις για μεγάλα N, επιβεβαιώθηκε και σε εκδοχές του προγράμματος που αξιοποιούσαν τεχνικές loop unrolling για εντολές SSE2. Η μία από αυτές, χρησιμοποιούσε 4 τετράδες στο εσωτερικό loop, υπολογίζοντας έτσι 16 στοιχεία ανά επανάληψη. Αυτό βέβαια απαιτεί το μέγεθος του πίνακα να είναι πολλαπλάσιο του 16, πράγμα που δε συμφωνεί με τις απαιτήσεις της εκφώνησης. Η δεύτερη υπολόγιζε ταυτόχρονα 4 στοιχεία του τρίτου πίνακα, χρησιμοποιώντας την ίδια γραμμή του πρώτου για να υπολογίσει το dot product με 4 στήλες του δεύτερου. Όλες οι παραπάνω εκδοχές δεν απέφεραν καλύτερη απόδοση για $N = 4000$, από αυτή που επιτεύχθηκε με loop unrolling ή με την

απλή εκδοχή του προγράμματος με SSE2 εντολές.

Ως προς τον επεξεργαστή, στον οποίο έγιναν οι δοκιμές, τα χαρακτηριστικά του παρατίθεται στον Πίνακα 2.

Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
Byte Order	Little Endian
CPU(s)	4
On-line CPU(s)	list 0-3
Thread(s) per core	1
Core(s) per socket	4
Socket(s)	1
NUMA node(s)	1
Vendor ID	GenuineIntel
CPU family	6
Model	60
Model name	Intel(R) Core(TM) i5-4440 CPU @ 3.10GHz
Stepping	3
CPU MHz	3229.390
CPU max MHz	3300,0000
CPU min MHz	800,0000
BogoMIPS	6185.47
Virtualization	VT-x
L1d cache	32K
L1i cache	32K
L2 cache	256K
L3 cache	6144K
NUMA node0 CPU(s)	0-3

Πίνακας 2: Χαρακτηριστικά του επεξεργαστή, αποτέλεσμα της εντολής lscpu