Lab1 Report

Χουλιαράς Ανδρέας ΑΕΜ:2143

Το σύστημα στο οποίο έγινε η εργασία έχει τα εξής χαρακτηριστικά:

CPU: Intel Core i7 6700 RAM: 16GB (2x8) DDR4-2133Mhz

OS: Ubuntu 16.4.5 LTS (σε Virtual Machine)

Kernel: 4.15.0-36-generic

Compiler: ICC 18.0.3

Σημείωση: Τα παρακάτω χρονομετρήθηκαν με την παραμετρο –O0 και οι παρατηρήσεις έγιναν με βάση αυτή, μέχρι να αναφερθεί διαφορετικά.

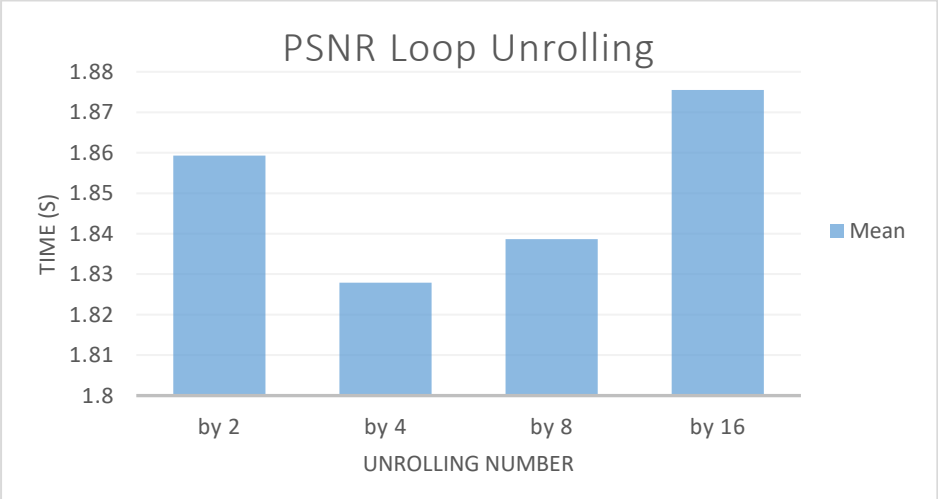
Sobel_v1: Εφαρμογή Loop interchange στο κομμάτι main computation και στην συνάρτηση της συνέλιξης. Κάναμε τις εναλλαγές στα σημεία που γινόταν διάτρεξη πρώτα κατά γραμμές ώστε μετά την αλλαγή να γίνετε διάτρεξη πρώτα κατά στήλες, έτσι ώστε να εκμεταλευόμαστε καλύτερα την cache του επεξεργαστή.

Βλέπουμε περίπου 32.8% βελτίωση στον μέσο χρόνο εκτέλεσης σε σχέση με την αρχική έκδοση.

Sobel_v2: Εφαρμογή Loop Unrolling στην συνάρτηση της συνέλιξης. Επειδή έχει μόνο 3 επαναλήψεις ανά for loop, μπορεί να γραφτεί με τρόπο τέτοιο ώστε να να μην έχουμε καθόλου for loops.

Δοκιμάζουμε την ίδια τεχνική στον υπολογισμό του PSNR. Θα ξεδιπλώσουμε μόνο το εσωτερικό loop, γιατί ξεδιπλώνοντας και το εξωτερικό θα αυξηθεί υπερβολικά ο κώδικας χωρίς ουσιαστικό όφελος.Μετά από δοκιμές το καλύτερο φαίνετε να είναι το ξεδίπλωμα loop κατά 4 οπότε και αυτό κρατάμε.

Ακολουθεί γράφημα δοκιμών Loop Unrolling για το PSNR:



Πετύχαμε περίπου 23.9% βελτίωση στον μέσο χρόνο εκτέλεσης σε σχέση με το προηγούμενο στάδιο.

Sobel_v3: Εφαρμογή Common Subexpression Elimination.Τα πολύ προφανή σημεία είναι οι εκφράσεις i*SIZE που τις συναντάμε πολύ συχνά μέσα σε διπλά loops και στην συνάρτηση συνέλιξης, όπου για να εφαρμοστεί πρέπει να γίνει επιμεριστική ιδιότητα. Ως συνέπεια εφαρμόστηκε και η τεχνική Strenght Reduction γιατί :

το (posy ± 1(ή 0))*SIZE έγινε temp ± SIZE(ή 0) , όπου temp=posy*SIZE. Οπότε αντικαταστήσαμε αυτούς τους πολλαπλασιασμούς με προσθέσεις.

Πετύχαμε περίπου 6.3% βελτίωση στον μέσο χρόνο εκτέλεσης σε σχέση με το προηγούμενο στάδιο.

Sobel_v4: Εφαρμογή Loop Invariant Code Motion σε 2 σημεία. Στο main computation και στον υπολογισμό του PSNR, βγάζω τα Common Subexpressions από το προηγούμενο στάδιο έξω από το εκάστοτε εσωτερικό loop. Ο μέσος χρόνος εκτέλεσης είναι χειρότερος από το προηγούμενο στάδιο. Δοκιμάζοντας ποιά από τις 2 αλλαγές φταίει καταλήγουμε, ότι χωρίς την εφαρμογή της τεχνικής στο main computation πέρνουμε καλύτερο χρόνο εκτέλεσης οπότε κρατάμε αυτήν την έκδοση του κώδικα.

Πετύχαμε περίπου 0.69% βελτίωση στον μέσο χρόνο εκτέλεσης σε σχέση με το προηγούμενο στάδιο.

Sobel_v5: Εφαρμογή τεχνικών υποβοήθισης Μεταγλωττιστή. Πιο συγκεκριμένα: Βάζουμε το πεδίο register για τις μεταβλητές που αποθηκεύσαμε τα Common Subexpressions του προ- προηγούμενου σταδίου, θα σημειώσουμε ως restrict τους pointers των αρχείων που χρησιμοποιεί ο κώδικας και σημειώνουμε ως const τα ορίσματα της συνάρτησης της συνέλιξης, μιας και δεν αλλάζουν μέσα στην συνάρτηση οι τιμές τους. Κατ'επέκταση σημειώνουμε ως const και τους operators της συνέλιξης.

Τρέχοντας τον κώδικα με την επιλογή -O0 δεν περιμένουμε κάποια βελτίωση γιατί οι πληροφορίες αυτές βοηθάνε τον μεταγλωττιστή να κάνει αυτόματες βελτιστοποιήσεις και εμείς σε αυτήν την φάση τις έχουμε απενεργοποιημένες.

Σε αυτό το σημείο έχω να παρατηρήσω το εξής.Έκανα μετρήσεις πολλές φορές και ο μέσος χρόνος εκτέλεσης κάποιες φορές ήταν καλύτερος από πριν και κάποιες φορές όχι.Σε γενικές γραμμές ο μέσος χρόνος εκτέλεσης χειροτέρεψε οπότε δεν θα γίνουν περαιτέρω βελτιστοποιήσεις με αυτην την έκδοση του κώδικα.

Sobel_v6: Ώρα να ξεφορτωθούμε την συνάρτηση της συνέλιξης με την τεχνική Function Inlining. Ο κώδικας έγινε εξαιρετικά δυσανάγνωστος στο σημείο που γινόταν κλήση της συνάρτησης αλλά η βελτίωση στον χρονο εκτέλεσης είναι ιδιαίτερα σημαντικη.

Επίσης θα ξαναβάλω στις μεταβλητές που αποθηκεύσαμε τα Common Subexpressions το πεδίο register γιατί βελτιώνει λίγο τον μέσο χρόνο εκτέλεσης.

Η βελτίωση είναι περίπου 4.9% με την αλλαγή αυτη και 4.7% χωρις την αλλαγή αυτή σε σχέση με τον κώδικα sobel_v4 .

Παρατήρηση: θα έδινα και ξεχωριστή έκδοση του κώδικα που να εχει μόνο την τελευταία αλλαγή (το πεδίο register για τις μεταβλητές που αποθηκεύσαμε τα Common Subexpressions) αλλά για κάποιο ανεξήγητο λόγο, μόνη της η αλλαγή αυτή χειροτερεύει το μέσο χρόνο εκτέλεσης σαν το sobel_v5 .

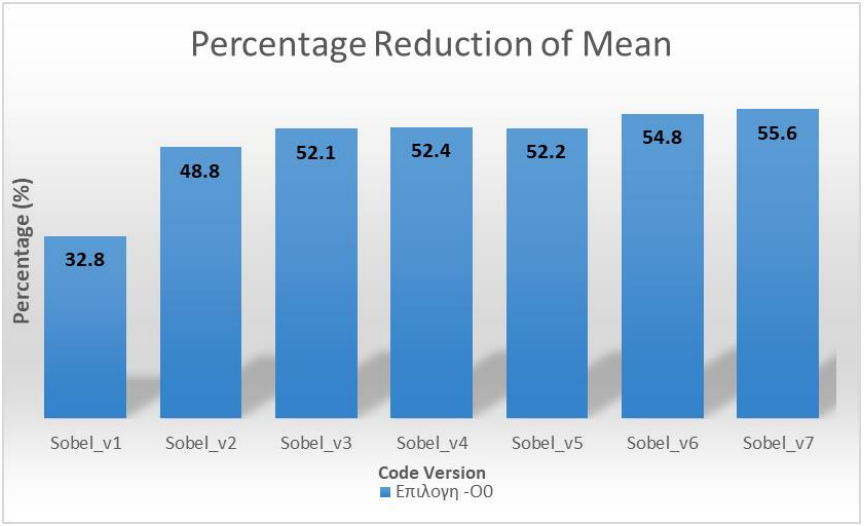
Sobel_v7: Εφαρμόζω την τεχνική loop invariant code motion για το κομμάτι του κώδικα που δεν κράτησα από το στάδιο 4(sobel_v4).

Πετύχαμε 1.84% βελτίωση σε σχέση με το προηγούμενο στάδιο.

Η τεχνικη του Loop Fusion δεν γινετε να εφαρμοσει για το συγκεκριμενο κωδικα ή τουλάχιστον εγώ δεν βρήκα κάποιο σημείο που να μπορώ να την εφαρμόσω.

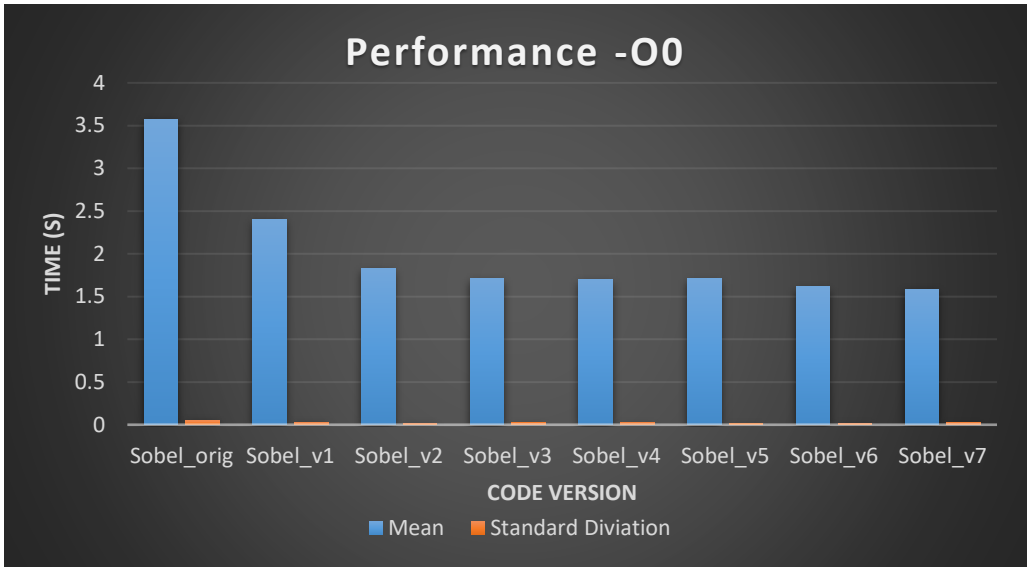
Μέχρι στιγμής έχουμε μειώσει τον χρόνο εκτέλεσης συνολικά κατά 55.6% χωρίς να εφαρμόσει ο μεταγλωττιστής καμία βελτιστοποίηση.

Στο ακόλουθο γράφημα φαίνεται η ποσοστιαία μείωση του μέσου όρου των χρόνων εκτέλεσης ανά στάδιο σε σχέση με την αρχική έκδοση:



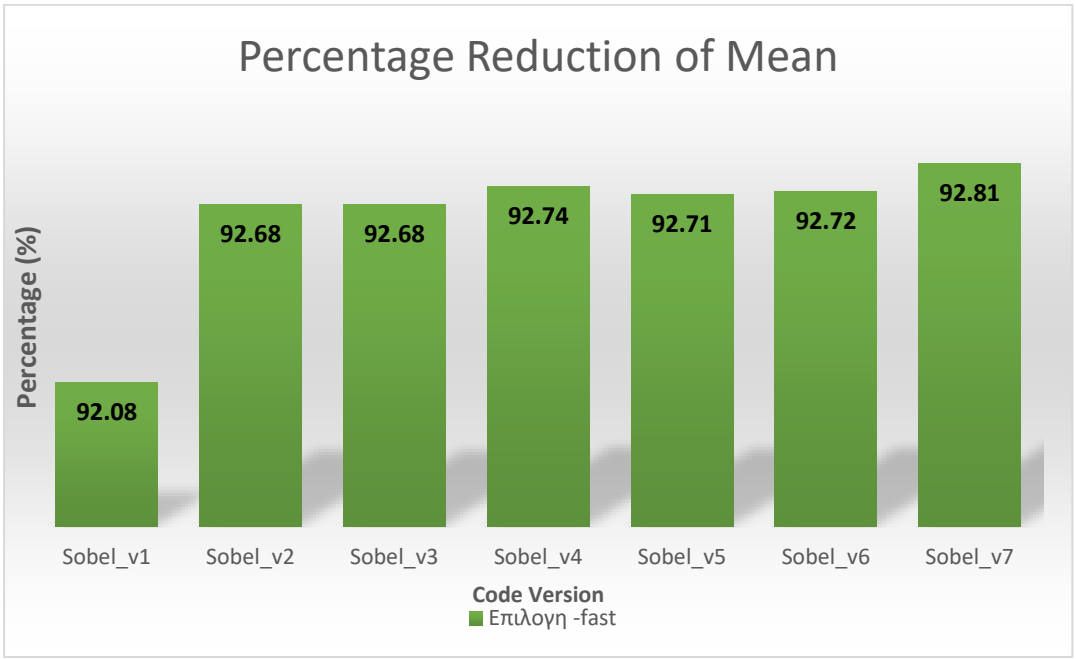
Όλες οι χρονομετρήσεις βρίσκονται αναλυτικά στο αρχείο excel Lab1_Statistics. Έκανα 12 πειράματα για κάθε έκδοση κώδικα όπως τις οδηγίες και τα έβαλα στο αρχείο χωρίς τη μεγαλύτερη και τη μικρότερη τιμή.

Στο ακόλουθο γράφημα φαίνεται η επίδοση κάθε έκδοσης με την παράμετρο –O0.

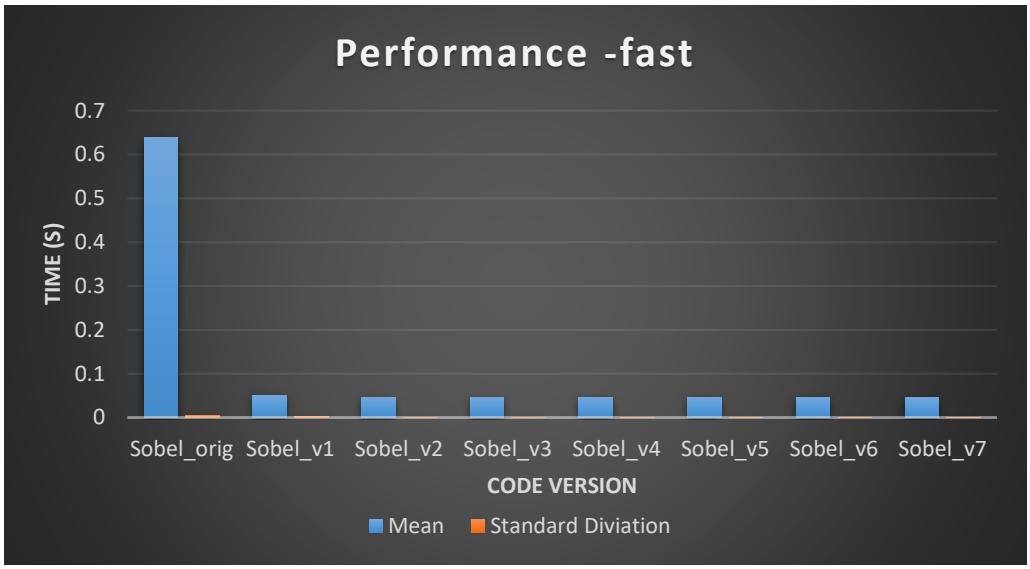


Κάνουμε χρονομέτρηση στους κώδικες από όλα τα στάδια και με την παράμετρο –fast.

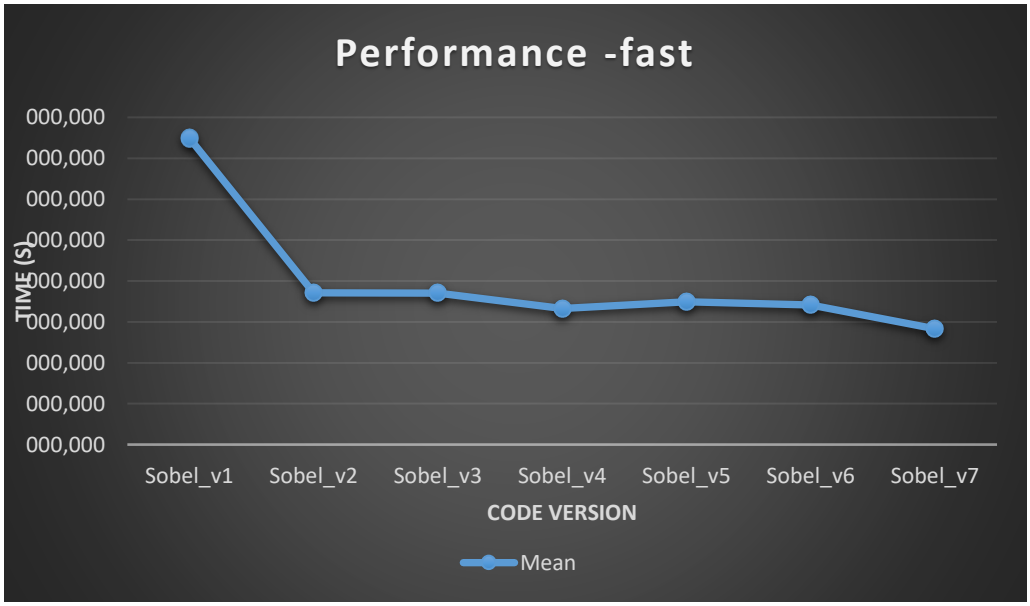
Στο ακόλουθο γράφημα φαίνεται η ποσοστιαία μείωση του μέσου όρου των χρόνων εκτέλεσης ανά στάδιο σε σχέση με την αρχική έκδοση :



Στο ακόλουθο γράφημα φαίνεται η επίδοση κάθε έκδοσης με την παράμετρο – fast



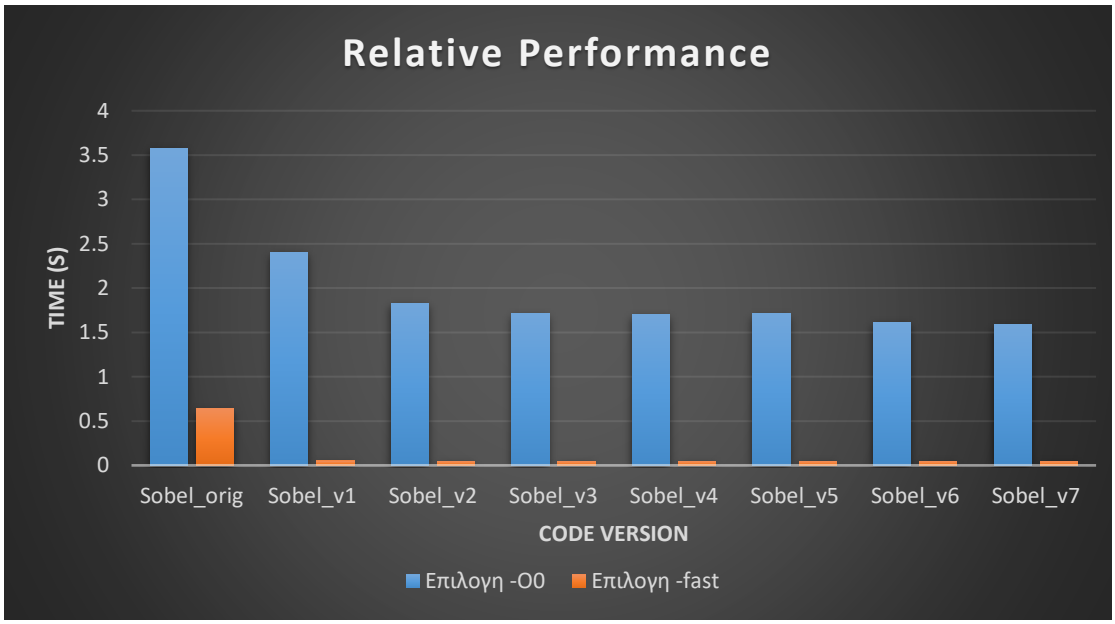
Στο επόμενο γράφημα αφαίρεσα τον αρχικό κώδικα γιατί τώρα δεν φαίνονται οι διαφορές στις διάφορες εκδόσεις βελτιστοποίησης:



Αν κοιτάξουμε τι προσέφερε η αυτόματη βελτιστοποίηση του μεταγλωττιστή παρατηρούμε 82.1% βελτίωση στον αρχικό κώδικα σε σχέση με πριν. Αν συγκρίνουμε τους χρόνους μεταξύ αρχικού κώδικα και του σταδίου 7 με την επιλογή –fast και στα δύο βλέπουμε μια βελτίωση κατά 92.8% στον χρόνο εκτέλεσης!!! Αυτό σημαίνει ότι οι αλλαγές που κάναμε βοήθησαν τον μεταγλωττιστή στις αυτόματες βελτιστοποιήσεις που κάνει, κατά 92.8%.

Συνολικά οι αλλαγές που κάναμε στον αρχικό κώδικα, με ενεργοποιημένες τις αυτόματες βελτιστοποιήσεις του μεταγλωττιστή, μείωσαν τον χρονο εκτέλεσης κατά 98.7%!!!

Ακολουθεί και το συγκρητικό γράφημα επίδοσης μεταξύ –O0 και –fast :



Στο αρχειο excel Lab1_Statistics ,πατώντας πάνω στα γραφήματα υπάρχει η επιλογή φίλτρου.Μπορείτε να αποκρύψετε ή να εμφανίσετε στοιχεία για να παρατηρήσετε καλύτερα τις διαφορές μεταξύ εκδόσεων του κώδικα.Για παράδειγμα, στα γραφήματα επίδοσης οι τυπικές αποκλήσεις σχεδόν δεν φαίνονται.Φιλτράροντας τους μέσους όρους γίνονται πιο ευδιάκριτες.