# 3ο μέρος εργασίας του μαθήματος

**«Σχεδιασμός Ενσωματωμένων Συστημάτων»**

**Ομάδα 10**

**Φοιτητές:**

Ηλίας Παπαδέας 56989

Χριστόφορος Σπάρταλης 56785



Περιεχόμενα

[3ο μέρος εργασίας του μαθήματος 1](#__RefHeading___Toc1613_764338909)

[Εισαγωγή 3](#__RefHeading___Toc1615_764338909)

[Θεωρητικό σκέλος 3](#__RefHeading___Toc1617_764338909)

[Lines 4](#__RefHeading___Toc1619_764338909)

[0. Χωρίς τη βελτιστοποίηση βρόγχου 4](#__RefHeading___Toc1621_764338909)

[1. Με τη βελτιστοποίηση βρόγχου 6](#__RefHeading___Toc1623_764338909)

[Block 7](#__RefHeading___Toc1625_764338909)

[0. Χωρίς τη βελτιστοποίηση βρόγχου 8](#__RefHeading___Toc1627_764338909)

[1. Με τη βελτιστοποίηση βρόγχου 10](#__RefHeading___Toc1629_764338909)

[Συνδυασμός 11](#__RefHeading___Toc1631_764338909)

[0. Χωρίς τη βελτιστοποίηση βρόγχου 12](#__RefHeading___Toc1633_764338909)

[1. Με τη βελτιστοποίηση βρόγχου 14](#__RefHeading___Toc1635_764338909)

[Παρατηρήσεις // Συμπεράσματα 15](#__RefHeading___Toc1637_764338909)

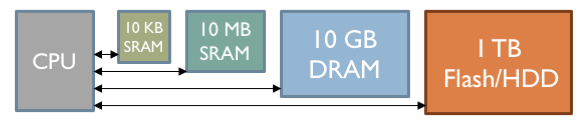
[Πηγές 17](#__RefHeading___Toc1639_764338909)

## Εισαγωγή

Στο τρίτο μέρος της εργασίας επικεντρωνόμαστε στη χρήση των μετασχηματισμών επαναχρησιμοποίησης δεδομένων. Παρόλα αυτά γίνεται χρήση της γνώσης από τα προηγούμενα μέρη της εργασίας. Για παράδειγμα η επιλογή των scatter files και των memorymaps βασίστηκε πάνω στα συμπεράσματα από το 2ο μέρος. Όμως για λόγους συντομίας δεν αναλωνόμαστε στο αναλύσουμε τις επιλογές μας (ούτως ή άλλως κάποιος μπορεί να ανατρέξει στο 2ο μέρος της εργασίας, όπου εξηγούμε αναλυτικά το σκεπτικό μας). Τώρα όσον αφορά τις βελτιστοποιήσεις βρόγχων (1ο μέρος της εργασίας) πράξαμε το ίδιο και για τους ίδιους λόγους. Όμως για να συμβαδίζουμε με το αρχείο «Ασκήσεις Ενσωματωμένων Συστημάτων» που δόθηκε ως πρότυπο για την εργασία και για να υπάρχει μια πλήρης επόπτευση, εφαρμόσαμε τις τεχνικές επαναχρησιμοποίησης δεδομένων τόσο στον βελτιστοποιημένο όσο και στον μη βελτιστοποιημένο κώδικα. Σημειώνουμε πως η βελτιστοποιήση βρόγχου που αναδείχθηκε ως η πλέον αποδοτική ήταν η loop unrolling.

## Θεωρητικό σκέλος

Σκοπός των μετασχηματισμών επαναχρησιμοποίησης δεδομένων είναι η μείωση των περιττών προσπελάσεων στην μνήμη. Αυτό γίνεται εισάγοντας μικρότερου μεγέθους μνήμες για την προσωρινή αποθήκευση δεδομένων που μπορεί να αποθηκευτούν πιο κοντά στον επεξεργαστή. Έτσι δημιουργείται μια ιεραρχία μνήμης όπου τα μικρού μεγέθους επίπεδα τοποθετούνται κοντά στον επεξεργαστή. Στα επίπεδα αυτά τοποθετούνται μεταβλητές που απαιτούν τον μεγαλύτερο αριθμό προσπελάσεων. Με αυτό τον τρόπο η μεταφορά δεδομένων από τη μνήμη στον επεξεργαστή -και αντίστροφα- γίνεται στα επίπεδα πλησίον του επεξεργαστή. Έτσι η μεταφορά δεδομένων είναι πιο γρήγορη (αφού μειώνονται οι προσπελάσεις στην εξωτερική μνήμη) και ταυτόχρονα το κόστος σε ενέργεια ανά προσπέλαση γίνεται μικρότερο.

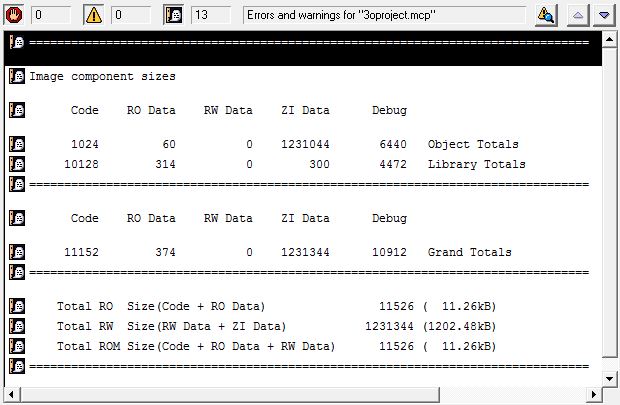


## Lines

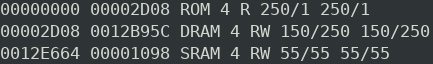
Σε αυτή την περίπτωση χρησιμοποιούμε τρεις buffers όπου ο καθένας τους είναι ένα μονοδιάστατος πίνακας 354 στοιχείων (ή θα μπορούσαμε να χρησιμοποιήσουμε έναν δισδιάστατο buffer 3x354 στοιχείων). Οι buffers αυτοί χρησιμοποιούνται κατά τη διάρκεια του φιλταρίσματος, όπου έχει νόημα η επαναχρησιμοποίηση δεδομένων. Πιο συγκεκριμένα, κάθε φορά που γίνεται το φιλτράρισμα στις 3 γραμμές (αφού η μάσκα έχει 3 γραμμές) σε κάθε buffer φορτώνονται τα περιεχόμενα της αμέσως επόμενης γραμμής.

### 0. Χωρίς τη βελτιστοποίηση βρόγχου

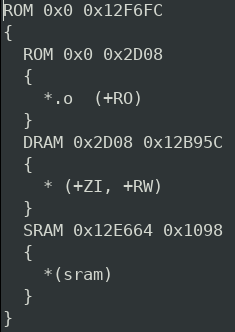
Image Component Sizes:



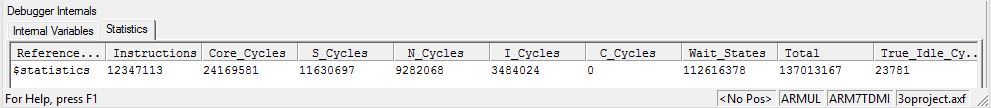
Memory Map:



Scatter File:

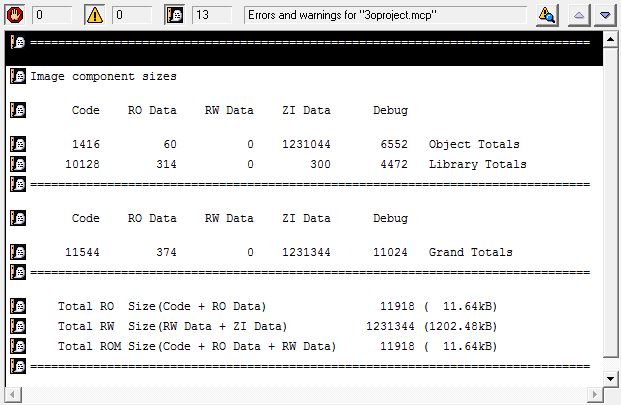


Debugger Internal Statistics:

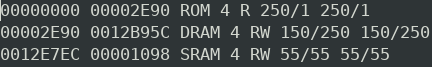


### 1. Με τη βελτιστοποίηση βρόγχου

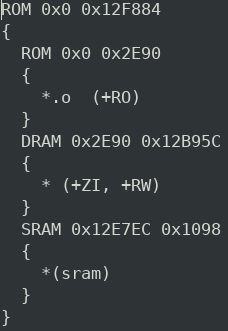
Image Component Sizes:



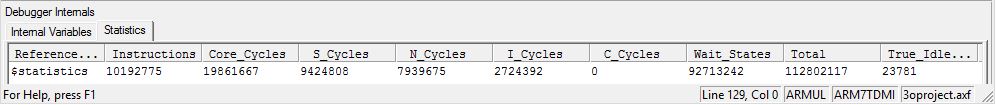
Memory Map:



Scatter File:



Debugger Internal Statistics:

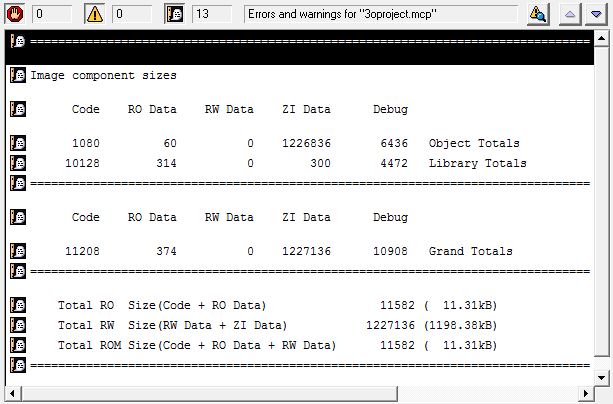


## Block

Σε αυτή την περίπτωση χρησιμοποιούμε ένα block 3x3 (όσο είναι και η μάσκα μας). Απλά όπως και στην προηγούμενη περίπτωση θεωρούμε ότι έχουμε 3 blocks των 3 στοιχείων, τα οποία κάνουν -όπως και παραπάνω- shift ανά γραμμή και όταν φτάσουν στο τέλος του μεγάλου πίνακα τότε ξαναγυρνάνε στην αρχή shiftαρισμένα κατά μία στήλη. Προφανώς περιμένουμε τα αποτελέσματα να είναι χειρότερα από προηγουμένως, αλλά παρόλα αυτά η μνήμη η οποία χρησιμοποιούμε έχει κατά πολύ μικρότερη χωρητικότητα. Οπότε σαν τεχνική μπορεί να φανεί χρήσιμη όταν έχουμε περιορισμένη μνήμη SRAM.

### 0. Χωρίς τη βελτιστοποίηση βρόγχου

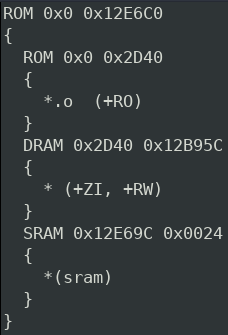
Image Component Sizes:



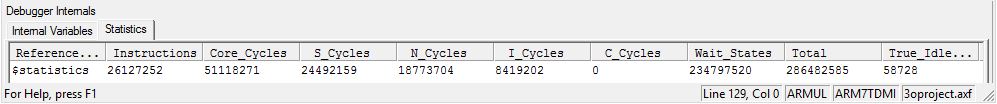
Memory Map:



Scatter File:

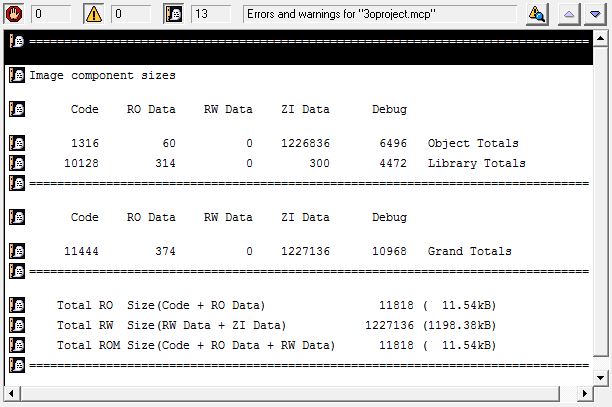


Debugger Internal Statistics:



### 1. Με τη βελτιστοποίηση βρόγχου

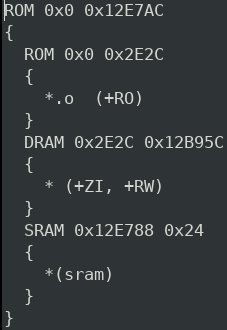
Image Component Sizes:



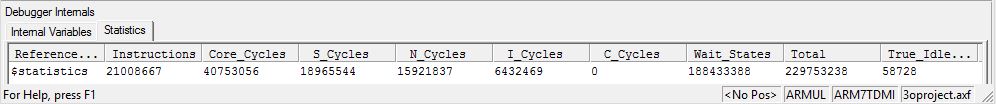
Memory Map:



Scatter File:



Debugger Internal Statistics:

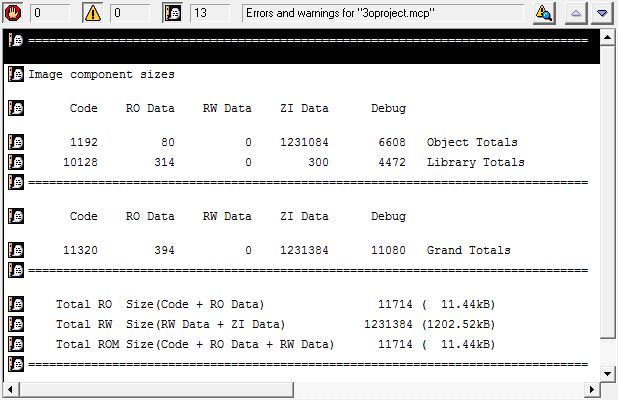


## Συνδυασμός

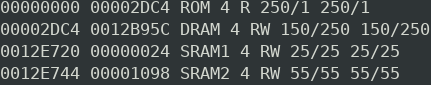
Τέλος συνδυάζουμε τις δύο προηγούμενες πετυχαίνοντας την μορφή που παρουσιάζεται στο θεωρητικό σκέλος. Δηλαδή έχουμε μια γρήγορη μνήμη στην οποία αποθηκεύονται κάθε φορά 3 γραμμές του κώδικα και μία ακόμα γρηγορότερη και μικρότερη μνήμη στην οποία αποθηκεύεται κάθε φορά ένα block 3x3 από τις γραμμές της προαναφερόμενης μνήμη. Όπως και στις προηγούμενες δύο περιπτώσεις το shiftάρισμα των γραμμών -και αντίστοιχα του block- γίνεται κάθε φορά ανά μία γραμμή, αφού κατ’ αυτό τον τρόπο τα στοιχεία είναι οργανωμένα στην μνήμη.

### 0. Χωρίς τη βελτιστοποίηση βρόγχου

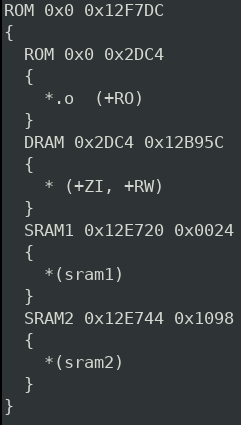
Image Component Sizes:



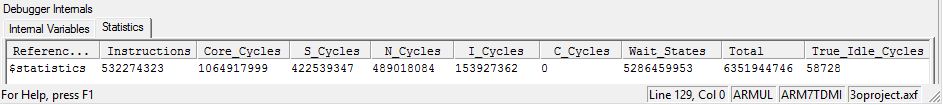
Memory Map:



Scatter File:

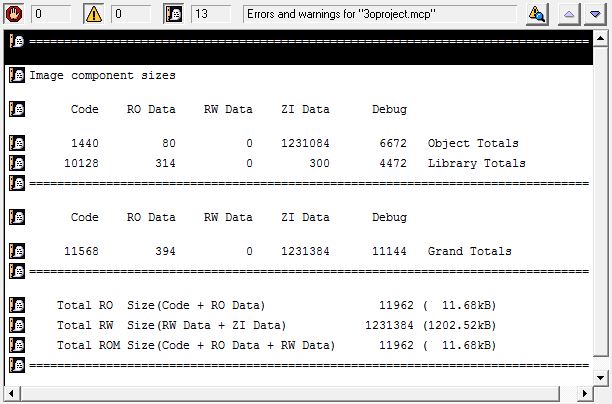


Debugger Internal Statistics:

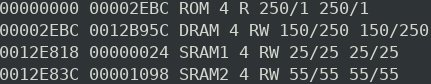


### 1. Με τη βελτιστοποίηση βρόγχου

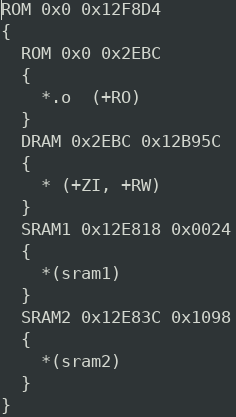
Image Component Sizes:



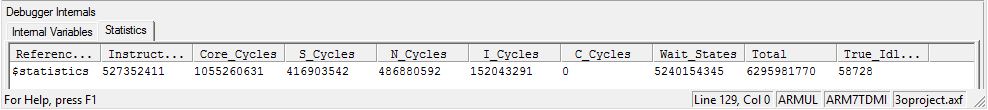
Memory Map:



Scatter File:

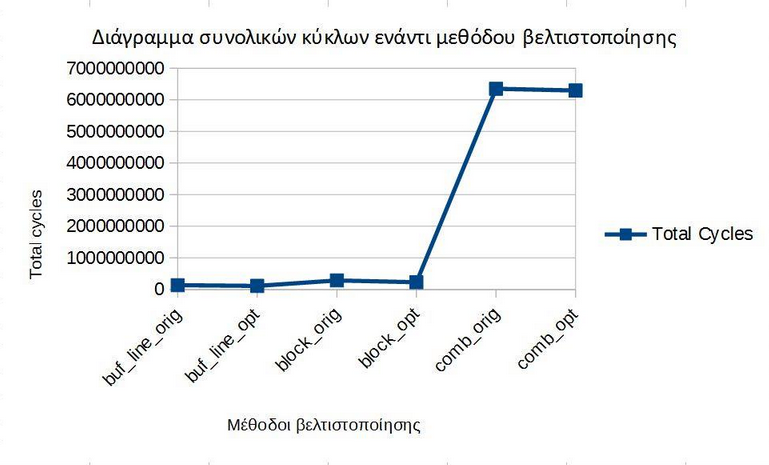


Debugger Internal Statistics:



## Παρατηρήσεις // Συμπεράσματα





Παρατηρούμε πως οι λιγότεροι συνολικοί κύκλοι επιτυγχάνονται στην περίπτωση του του buffer 3x354 με τον κώδικα να έχει υποστεί βελτιστοποίηση βρόγχου.

Προφανώς στην περίπτωση που χρησιμοποιούμε block 3x3 έχουμε περισσότερους κύκλους, αφού είναι λιγότερα τα δεδομένα τα οποία αποθηκεύονται στην μνήμη. Βέβαια στο παράδειγμα που εφαρμόσαμε, τόσο η μνήμη που έχει χωρητικότητα 3x354, όσο και αυτή που έχει 3x3 έχουν την ίδια ταχύτητα προσπέλασης και γραφής τόσο για τα σειριακά, όσο και για τα μη σειριακά δεδομένα. Ενώ θα ήταν ρεαλιστικό -όπως αναλύσαμε και στο 2ο μέρος της εργασίας- η μικρότερη μνήμη να έχει και μεγαλύτερη ταχύτητα.

Βλέπουμε πως στην περίπτωση που χρησιμοποιούμε δύο επίπεδα γρήγορης μνήμης (π.χ. cache) οι συνολικοί κύκλοι εκτινάσσονται. Αυτό μπορεί να οφείλεται στο γεγονός ότι το shiftάρισμα του buffer 3x354 από την αρχή έως και το τέλος του μεγάλου πίνακα δε γίνεται μία, αλλά 354 φορές.

## Πηγές

Αρχείο «Ασκήσεις Ενσωματωμένων Συστημάτων» από το eclass του μαθήματος «Σχεδιασμός Ενσωματωμένων Συστημάτων».

Διαφάνειες και κώδικας του 3ου εργαστηρίου από το μάθημα «Σχεδιασμός Ενσωματωμένων Συστημάτων».