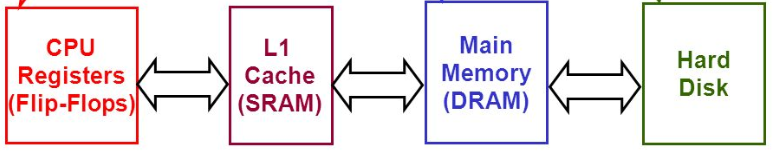
|  |
| --- |
| Σχεδιασμός Ενσωματωμένων Συστημάτων |
| 3ο μέρος της εργασίας |
| Μαρουλάκος-Σεφεριάδης Πάρης : 57128 Μπαντήλας Παναγιώτης : 57138 |

Ομάδα 34

**Εισαγωγή**

Σ’ αυτό το μέρος της εργασίας θα ασχοληθούμε στην μελέτη και χρήση των μετασχηματισμών επαναχρησιμοποίησης δεδομένων, όπως αυτές παρουσιάζονται στο αρχείο «Ασκήσεις Ενσωματωμένων Συστημάτων». Η χρήση αυτών τον τεχνικών δεν μας είναι τελείως άγνωστη καθώς εφαρμόσαμε μια απ’ αυτές στο δεύτερο μέρος της εργασίας (χρήση buffers).

Η χρήση αυτών των τεχνικών αποσκοπεί στην αύξηση της ταχύτητας και της απόδοσης του συστήματος μας, καθώς μειώνει της περιττές προσπελάσεις στην μνήμη. Αυτό επιτυγχάνετε τοποθετώντας μικρού μεγέθους και υψηλής ταχύτητας μνήμες (SRAM) κοντά στην CPU, οι οποίες αποθηκεύουν τις μεταβλητές που απαιτούν πολλές προσπελάσεις κατά την διάρκεια του προγράμματος μας. Έτσι δημιουργούμε μια ιεραρχία μνήμης, η οποία μας γλιτώνει από τις προσπελάσεις με τα πιο αργά κομμάτια της μνήμης(DRAM),μειώνοντας δραστικά τον χρόνο εκτέλεσης.



-Διάγραμμα ιεράρχησης μνήμης

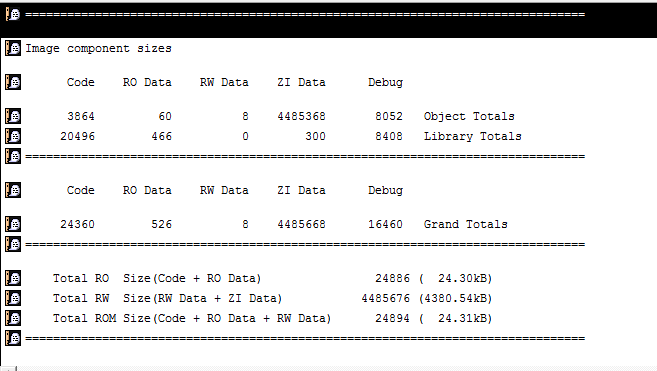
**ΜΕΘΟΔΟΙ**

**1.Lines**

Σε αυτή την μέθοδο χρησιμοποιούμε 3 Buffers, όπου ο κάθε ένας απ’αυτούς αποτελεί έναν πινακα 1x354.Τα στοιχεία που θα αποθηκευτούν στους Buffers είναι τα εικονοστοιχεία της εικόνας κατά την διάρκεια του φιλτραρίσματος, τα οποία ξαναχρησιμοποιούνται.Πιο συγκεκριμένα,κάθε φορά που γίνεται το φιλτράρισμα στις 3 γραμμές(kernel 3x3)σε κάθε Buffer φορτώνονται τα περιεχόμενα της αμέσως επόμενης γραμμής.

Armulator:

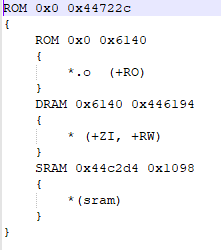
-Image component sizes:



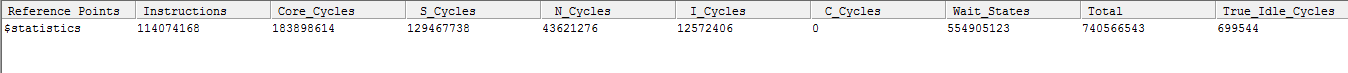
-Memory Map:



-Scatter File:



- Debugger Internals Statistics:

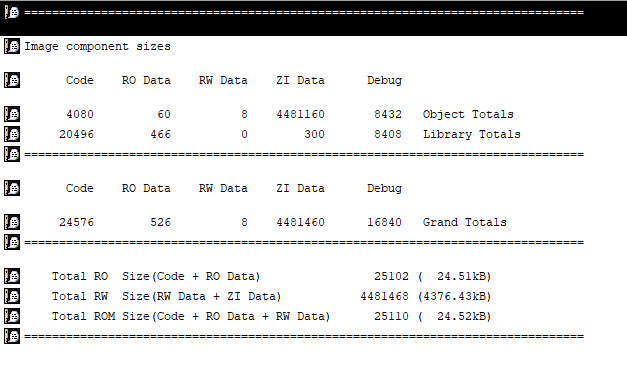


**2.Block**

Σε αυτή την μέθοδο χρησιμοποιούμε ένα Block 3x3.Οπως και στην προηγούμενη περίπτωση θεωρούμε ότι έχουμε 3 Blocks των 3ων στοιχείων το καθένα, τα οποία κάνουν shift ανά γραμμή και όταν φτάσουν στο τέλος του μεγάλου πίνακα τότε ξαναγυρνάνε στην αρχή shifted κατά μια στήλη. Αυτή η μέθοδος επί της ουσίας είναι ίδια με την προηγούμενη αλλά αντί οι Buffers να έχουν το μέγεθος της εικόνας, έχουν το μέγεθος του kernel. Είναι φανερό ότι η απόδοση αυτής της μεθόδου είναι μικρότερη από της προηγούμενης, όμως χρησιμοποιεί πολύ λιγότερη μνήμη πράγμα που την κάνει χρήσιμη σε εφαρμογές με περιορισμένη SRAM.

Armulator:

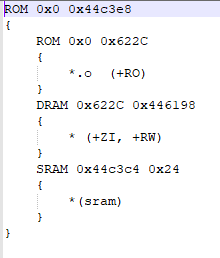
-Image component sizes:



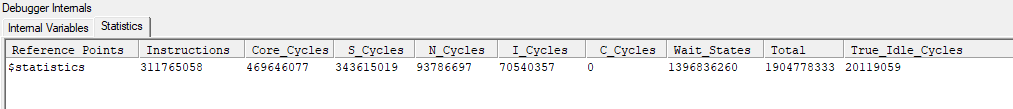
-Memory Map:



-Scatter File:



- Debugger Internals Statistics:

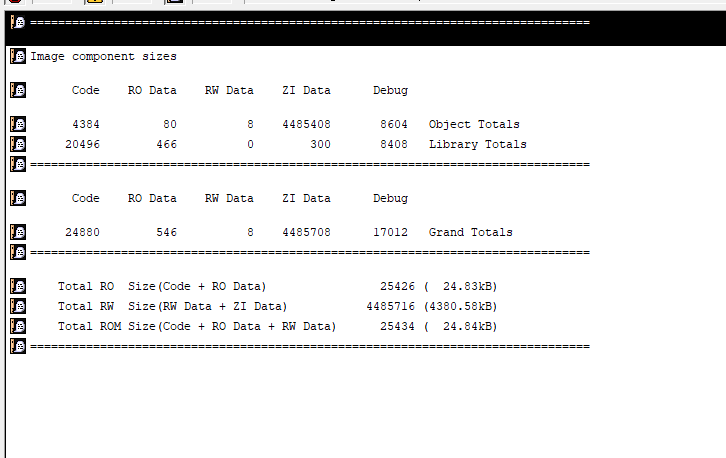


**3.Combination**

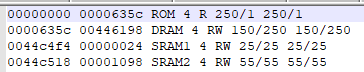
Ο συνδυασμός των δυο προηγούμενων μεθόδων βασίζεται στην ιεράρχηση μνήμης που παρατίθεται παραπάνω με την διαφορά ότι υπάρχει μια ακόμα SRAM(cache L2).Ο τροπος που λειτουργεί είναι ότι οι buffers αποθηκεύονται στην cache L2(αρκετά μεγαλύτερη και με μικρότερη ταχύτητα από την cache L1),ενώ τα Blocks στην cache L1.Οντας συνδυασμός των μεθόδων το shift γίνεται με τον ίδιο τρόπο, δηλαδή ανά μια γραμμή κάθε φορά, αφού έτσι είναι αποθηκευμένα τα στοιχεία στην μνήμη.

Armulator:

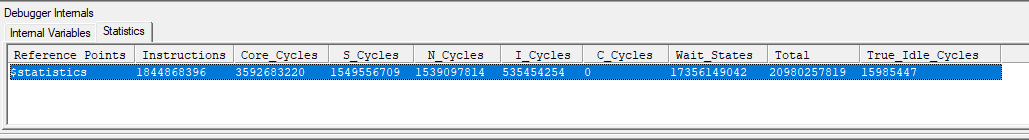
-Image component sizes:



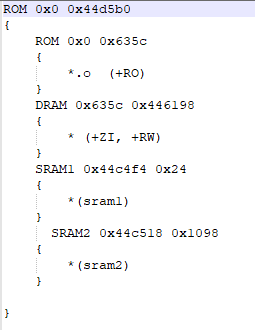
-Memory Map:



- Debugger Internals Statistics:



-Scatter File:



**Συμπεράσματα**

|  |  |
| --- | --- |
| Μέθοδος | Κύκλοι |
| Buffers | 740.566.543 |
| Block | 1.904.778.333 |
| Combination | 20.980.257.819 |

Από τα αποτελέσματα παρατηρούμε ότι οι λιγότεροι συνολικοί κύκλοι επιτυγχάνονται με την μέθοδο των Buffers.Στην μέθοδο του Block οι κύκλοι προφανώς και είναι περισσότεροι καθώς αποθηκεύονται λιγότερα από τα στοιχεία προς επεξεργασία στην SRAM.Ενώ στην τελευταία περίπτωση βλέπουμε ότι οι κύκλοι αυξάνονται δραματικά,αυτό πιθανώς οφείλεται στο ότι το shift του buffer από την αρχή ως το τέλος του μεγάλου πίνακα δεν γίνεται μια αλλά 354 φορές.