

**3ο μέρος εργασίας του μαθήματος
«Σχεδιασμός Ενσωματωμένων
Συστημάτων»**

Ομάδα 10

Φοιτητές:

Ηλίας Παπαδέας 56989

Χριστόφορος Σπάρταλης 56785



Περιεχόμενα

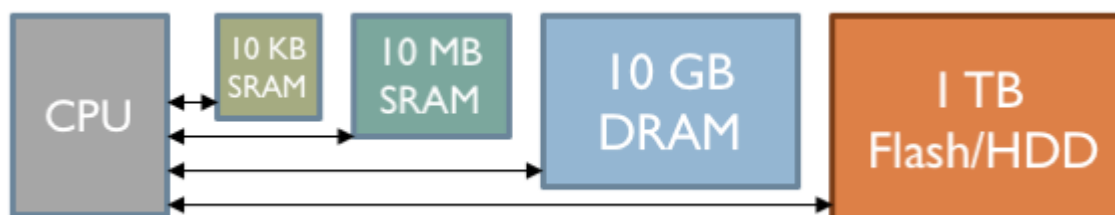
3ο μέρος εργασίας του μαθήματος.....	1
Εισαγωγή.....	3
Θεωρητικό σκέλος.....	3
Lines.....	4
0. Χωρίς τη βελτιστοποίηση βρόγχου.....	4
1. Με τη βελτιστοποίηση βρόγχου.....	6
Block.....	7
0. Χωρίς τη βελτιστοποίηση βρόγχου.....	8
1. Με τη βελτιστοποίηση βρόγχου.....	10
Συνδυασμός.....	11
0. Χωρίς τη βελτιστοποίηση βρόγχου.....	12
1. Με τη βελτιστοποίηση βρόγχου.....	14
Παρατηρήσεις // Συμπεράσματα.....	15
Πηγές.....	17

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 βελτιστοποίηση 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:

```
00000000 00002D08 ROM 4 R 250/1 250/1
00002D08 0012B95C DRAM 4 RW 150/250 150/250
0012E664 00001098 SRAM 4 RW 55/55 55/55
```

Scatter File:

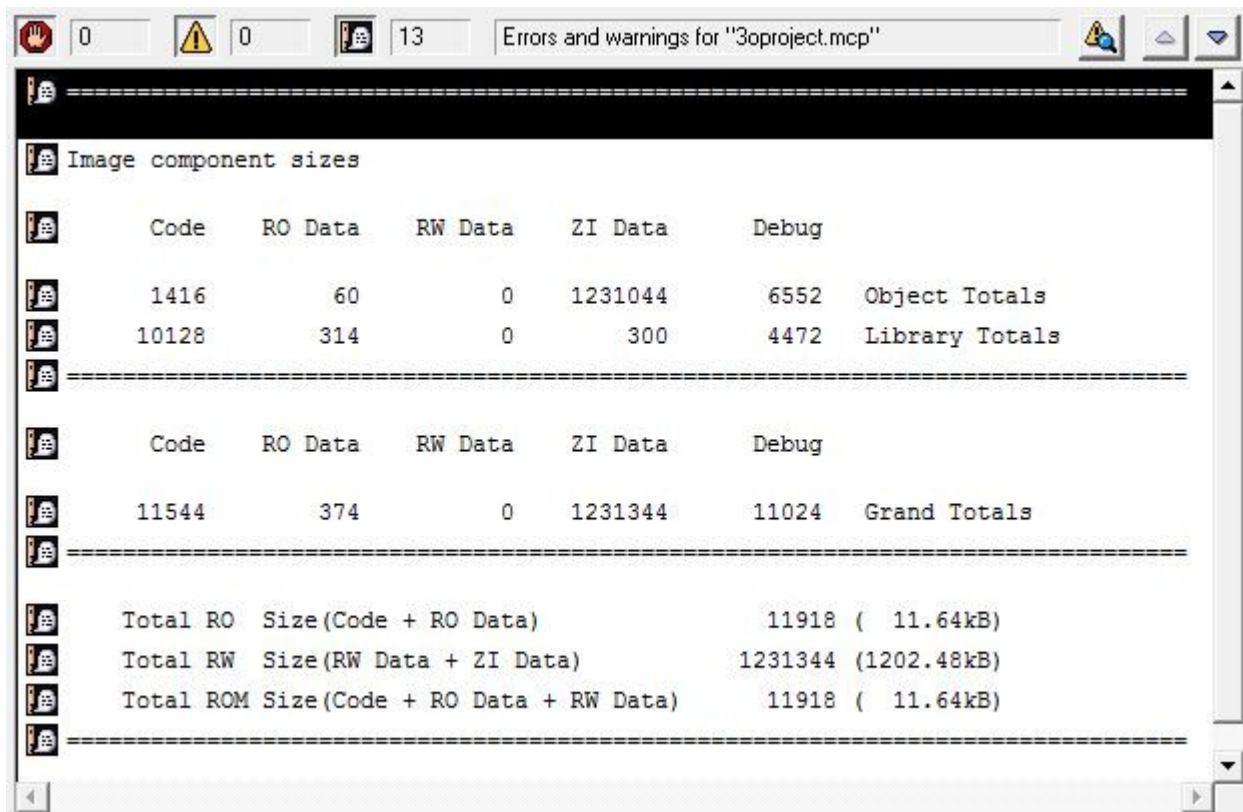
```
ROM 0x0 0x12F6FC
{
  ROM 0x0 0x2D08
  {
    *.o (+R0)
  }
  DRAM 0x2D08 0x12B95C
  {
    * (+ZI, +RW)
  }
  SRAM 0x12E664 0x1098
  {
    *(sram)
  }
}
```

Debugger Internal Statistics:

Debugger Internals									
Internal Variables		Statistics							
Reference...	Instructions	Core_Cycles	S_Cycles	N_Cycles	I_Cycles	C_Cycles	Wait_States	Total	True_Idle_Cy..
\$statistics	12347113	24169581	11630697	9282068	3484024	0	112616378	137013167	23781
For Help, press F1							<No Pos>	ARMUL	ARM7TDMI 3oproject.axf

1. Με τη βελτιστοποίηση βρόγχου

Image Component Sizes:



The screenshot shows a window titled "Errors and warnings for '3oproject.mcp'". Inside, there is a section titled "Image component sizes" which contains two tables. The first table shows component sizes for "Object Totals" and "Library Totals". The second table shows "Grand Totals" and a summary of total sizes for RO, RW, and ROM.

	Code	RO Data	RW Data	ZI Data	Debug	
	1416	60	0	1231044	6552	Object Totals
	10128	314	0	300	4472	Library Totals

	Code	RO Data	RW Data	ZI Data	Debug	
	11544	374	0	1231344	11024	Grand Totals

Total RO	Size(Code + RO Data)	11918	(11.64kB)
Total RW	Size(RW Data + ZI Data)	1231344	(1202.48kB)
Total ROM	Size(Code + RO Data + RW Data)	11918	(11.64kB)

Memory Map:

```
00000000 00002E90 ROM 4 R 250/1 250/1
00002E90 0012B95C DRAM 4 RW 150/250 150/250
0012E7EC 00001098 SRAM 4 RW 55/55 55/55
```

Scatter File:

```
ROM 0x0 0x12F884
{
  ROM 0x0 0x2E90
  {
    *.o (+R0)
  }
  DRAM 0x2E90 0x12B95C
  {
    * (+ZI, +RW)
  }
  SRAM 0x12E7EC 0x1098
  {
    *(sram)
  }
}
```

Debugger Internal Statistics:

Debugger Internals									
Internal Variables					Statistics				
Reference...	Instructions	Core_Cycles	S_Cycles	N_Cycles	I_Cycles	C_Cycles	Wait_States	Total	True_Idle...
\$statistics	10192775	19861667	9424808	7939675	2724392	0	92713242	112802117	23781

For Help, press F1

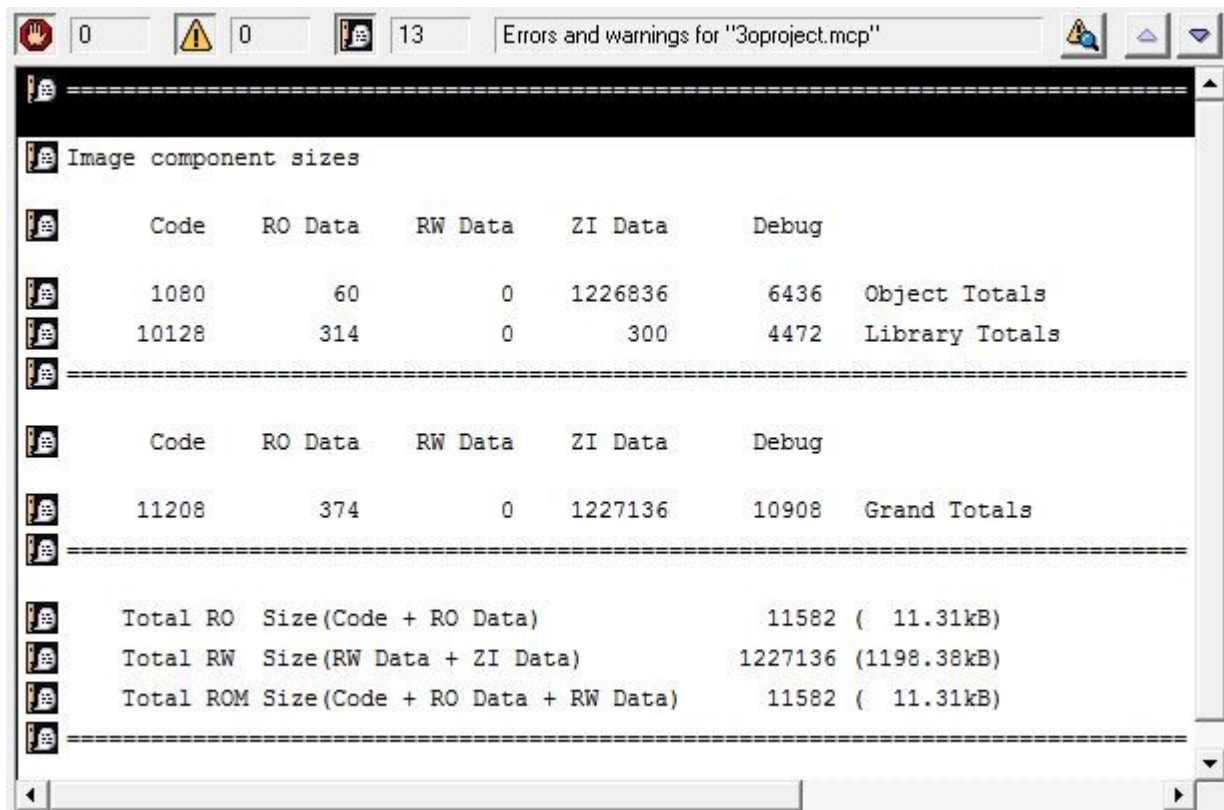
Line 129, Col 0 | ARMUL | ARM7TDMI | 3oproject.axf

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:



	Code	RO Data	RW Data	ZI Data	Debug	
	1080	60	0	1226836	6436	Object Totals
	10128	314	0	300	4472	Library Totals
=====						
	Code	RO Data	RW Data	ZI Data	Debug	
	11208	374	0	1227136	10908	Grand Totals
=====						
	Total RO Size(Code + RO Data)				11582 (11.31kB)	
	Total RW Size(RW Data + ZI Data)				1227136 (1198.38kB)	
	Total ROM Size(Code + RO Data + RW Data)				11582 (11.31kB)	

Memory Map:

```
00000000 00002D40 ROM 4 R 250/1 250/1
00002D40 0012B95C DRAM 4 RW 150/250 150/250
0012E69C 00000024 SRAM 4 RW 55/55 55/55
```


Scatter File:

```
ROM 0x0 0x12E6C0
{
  ROM 0x0 0x2D40
  {
    *.o (+R0)
  }
  DRAM 0x2D40 0x12B95C
  {
    * (+ZI, +RW)
  }
  SRAM 0x12E69C 0x0024
  {
    *(sram)
  }
}
```

Debugger Internal Statistics:

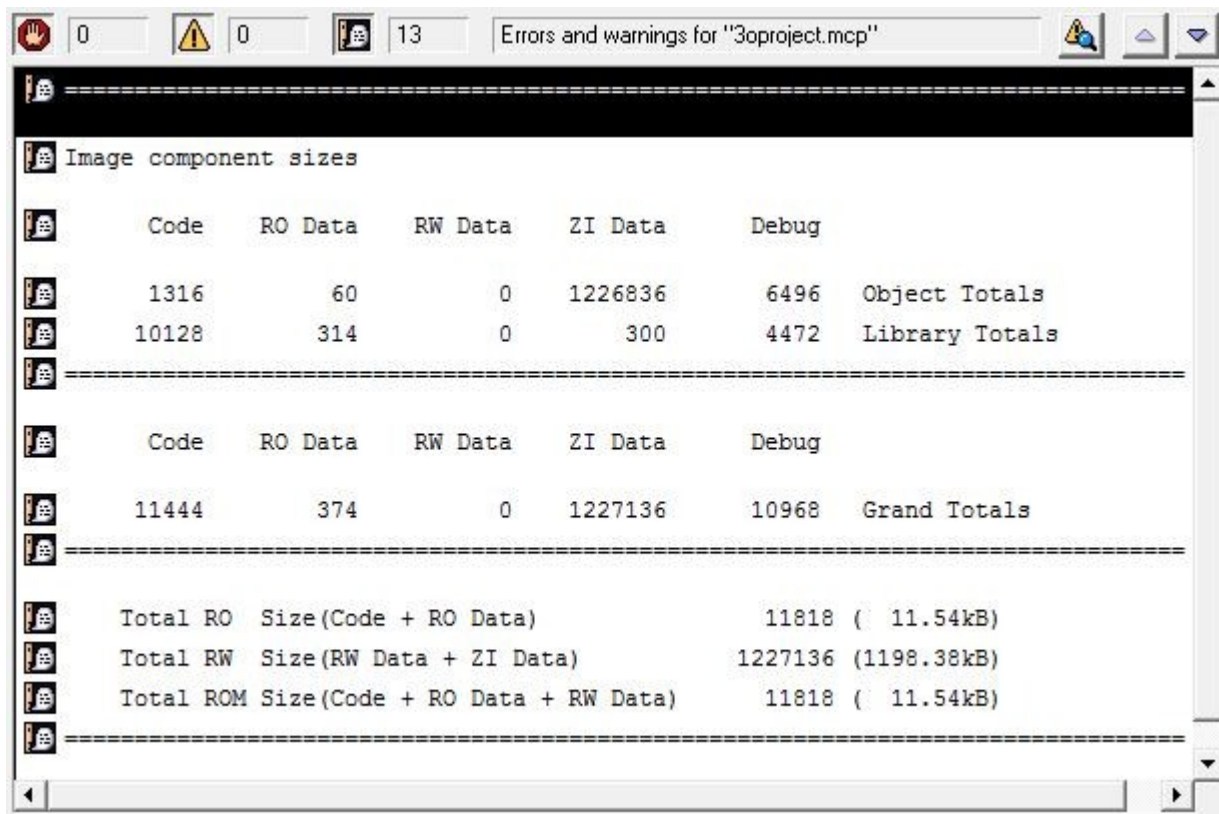
Debugger Internals									
Internal Variables		Statistics							
Reference...	Instructions	Core_Cycles	S_Cycles	N_Cycles	I_Cycles	C_Cycles	Wait_States	Total	True_Idle...
\$statistics	26127252	51118271	24492159	18773704	8419202	0	234797520	286482585	58728

For Help, press F1

Line 129, Col 0 | ARMUL | ARM7TDMI | 3oproject.axf

1. Με τη βελτιστοποίηση βρόγχου

Image Component Sizes:



	Code	RO Data	RW Data	ZI Data	Debug	
	1316	60	0	1226836	6496	Object Totals
	10128	314	0	300	4472	Library Totals
=====						
	Code	RO Data	RW Data	ZI Data	Debug	
	11444	374	0	1227136	10968	Grand Totals
=====						
Total RO	Size(Code + RO Data)				11818 (11.54kB)	
Total RW	Size(RW Data + ZI Data)				1227136 (1198.38kB)	
Total ROM	Size(Code + RO Data + RW Data)				11818 (11.54kB)	

Memory Map:

```
00000000 00002E2C ROM 4 R 250/1 250/1
00002E2C 0012B95C DRAM 4 RW 150/250 150/250
0012E788 00000024 SRAM 4 RW 55/55 55/55
```

Scatter File:

```
ROM 0x0 0x12E7AC
{
  ROM 0x0 0x2E2C
  {
    *.o (+R0)
  }
  DRAM 0x2E2C 0x12B95C
  {
    * (+ZI, +RW)
  }
  SRAM 0x12E788 0x24
  {
    *(sram)
  }
}
```

Debugger Internal Statistics:

Debugger Internals									
Internal Variables		Statistics							
Reference...	Instructions	Core_Cycles	S_Cycles	N_Cycles	I_Cycles	C_Cycles	Wait_States	Total	True_Idle...
\$statistics	21008667	40753056	18965544	15921837	6432469	0	188433388	229753238	58728
For Help, press F1							<No Pos>	ARMUL	ARM7TDMI 3oproject.axf

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:



0 0 13 Errors and warnings for "3oproject.mcp"

=====

Image component sizes

	Code	RO Data	RW Data	ZI Data	Debug	
	1192	80	0	1231084	6608	Object Totals
	10128	314	0	300	4472	Library Totals

=====

	Code	RO Data	RW Data	ZI Data	Debug	
	11320	394	0	1231384	11080	Grand Totals

=====

Total RO	Size (Code + RO Data)	11714 (11.44kB)
Total RW	Size (RW Data + ZI Data)	1231384 (1202.52kB)
Total ROM	Size (Code + RO Data + RW Data)	11714 (11.44kB)

=====

Memory Map:

```
00000000 00002DC4 ROM 4 R 250/1 250/1
00002DC4 0012B95C DRAM 4 RW 150/250 150/250
0012E720 00000024 SRAM1 4 RW 25/25 25/25
0012E744 00001098 SRAM2 4 RW 55/55 55/55
```

Scatter File:

```
ROM 0x0 0x12F7DC
{
  ROM 0x0 0x2DC4
  {
    *.o (+R0)
  }
  DRAM 0x2DC4 0x12B95C
  {
    * (+ZI, +RW)
  }
  SRAM1 0x12E720 0x0024
  {
    *(sram1)
  }
  SRAM2 0x12E744 0x1098
  {
    *(sram2)
  }
}
```

Debugger Internal Statistics:

Debugger Internals

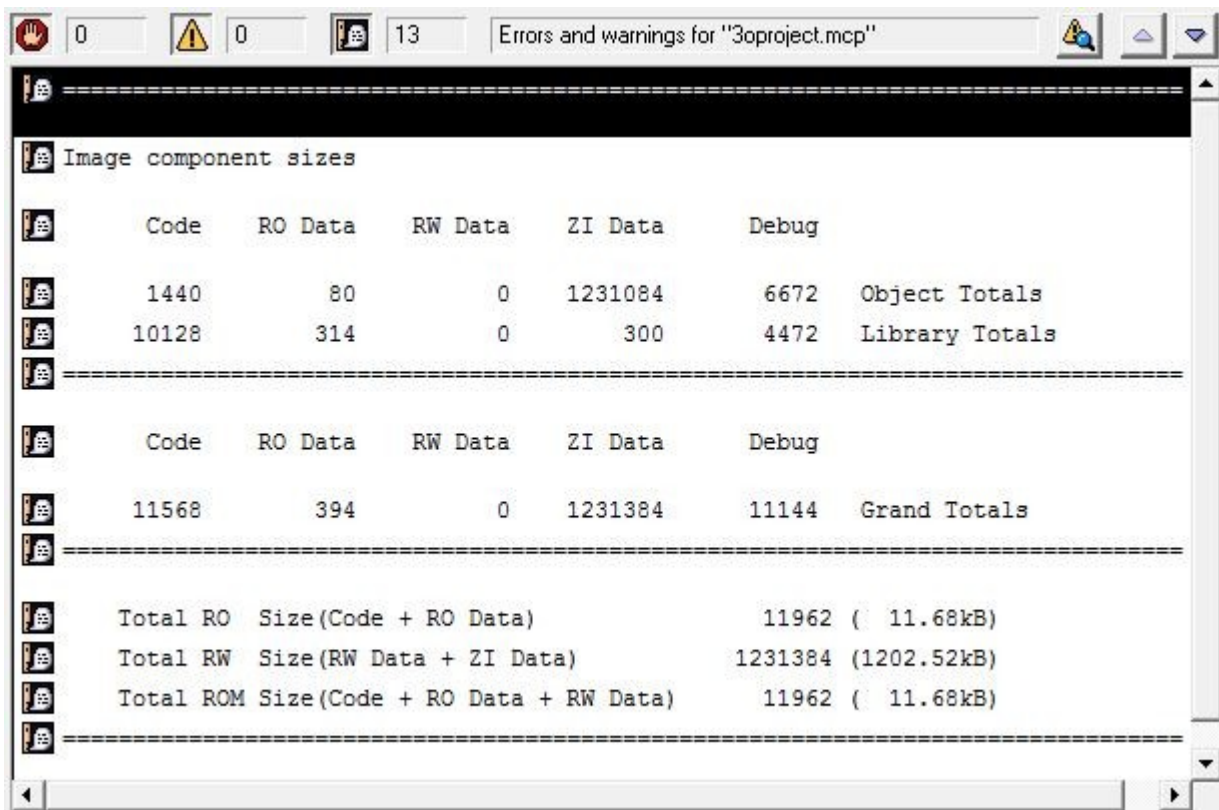
Internal Variables		Statistics								
Referenc...	Instructions	Core_Cycles	S_Cycles	N_Cycles	I_Cycles	C_Cycles	Wait_States	Total	True_Idle_Cycles	
\$statistics	532274323	1064917999	422539347	489018084	153927362	0	5286459953	6351944746	58728	

For Help, press F1

Line 129, Col 0 | ARMUL | ARM7TDMI | 3oproject.axf

1. Με τη βελτιστοποίηση βρόγχου

Image Component Sizes:



	Code	RO Data	RW Data	ZI Data	Debug	
	1440	80	0	1231084	6672	Object Totals
	10128	314	0	300	4472	Library Totals
=====						
	Code	RO Data	RW Data	ZI Data	Debug	
	11568	394	0	1231384	11144	Grand Totals
=====						
Total RO	Size(Code + RO Data)				11962 (11.68kB)	
Total RW	Size(RW Data + ZI Data)				1231384 (1202.52kB)	
Total ROM	Size(Code + RO Data + RW Data)				11962 (11.68kB)	

Memory Map:

```
00000000 00002EBC ROM 4 R 250/1 250/1
00002EBC 0012B95C DRAM 4 RW 150/250 150/250
0012E818 00000024 SRAM1 4 RW 25/25 25/25
0012E83C 00001098 SRAM2 4 RW 55/55 55/55
```

Scatter File:

```
ROM 0x0 0x12F8D4
{
  ROM 0x0 0x2EBC
  {
    *.o (+R0)
  }
  DRAM 0x2EBC 0x12B95C
  {
    * (+ZI, +RW)
  }
  SRAM1 0x12E818 0x0024
  {
    *(sram1)
  }
  SRAM2 0x12E83C 0x1098
  {
    *(sram2)
  }
}
```

Debugger Internal Statistics:

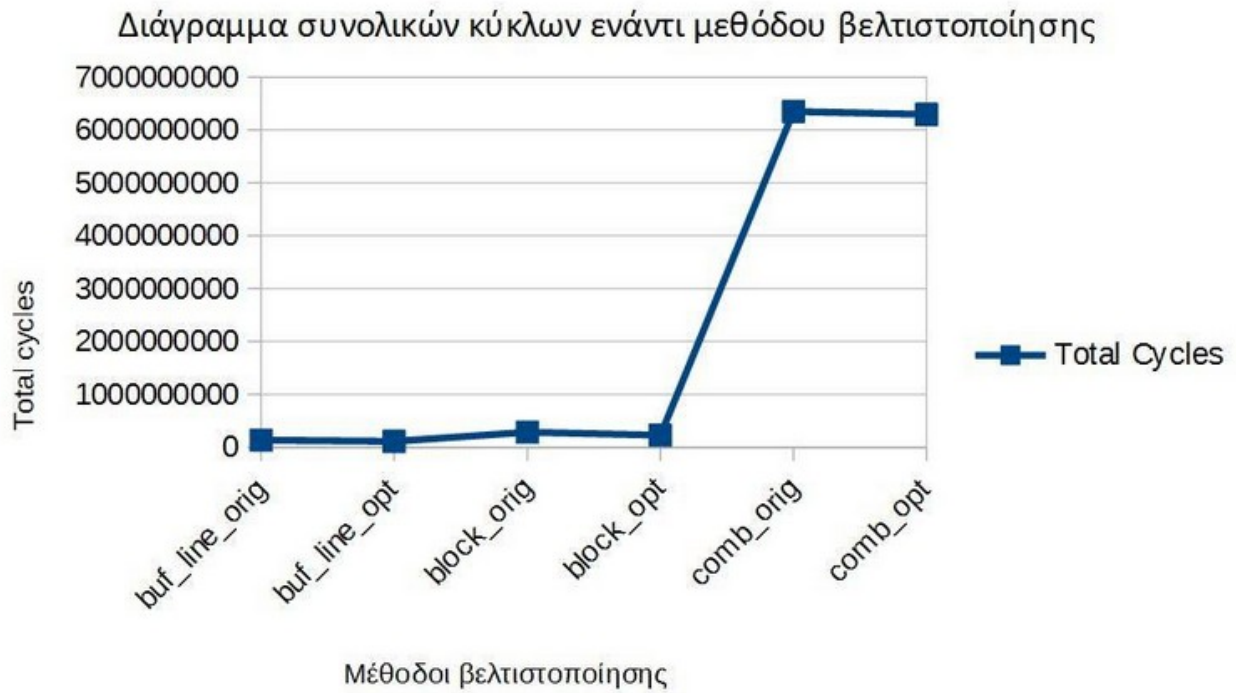
Debugger Internals										
Internal Variables		Statistics								
Referenc...	Instruct...	Core_Cycles	S_Cycles	N_Cycles	I_Cycles	C_Cycles	Wait_States	Total	True_Idl...	
\$statistics	527352411	1055260631	416903542	486880592	152043291	0	5240154345	6295981770	58728	

For Help, press F1

Line 129, Col 0 | ARMUL | ARM7TDMI | 3oproject.axf

Παρατηρήσεις // Συμπεράσματα

Methods	Total Cycles
buf_line_orig	137013167
buf_line_opt	112802117
block_orig	286482585
block_opt	229753238
comb_orig	6351944746
comb_opt	6295981770



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».