ΕΡΓΑΣΤΗΡΙΟ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

Τριαντάφυλλος Πράππας ΑΜ:1067504

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 2

ΑΘΡΟΙΣΗ BYTES

```
.arm
 .text
 .global main

 main:
 STMDB R13!, {R0-R12}
 LDR R0, =PinakasA
 LDRB R1, [R0, #0]
 LDR R2, =PinakasB
 LDRB R3, [R2, #0]
 LDR R4, =Result
 ADD R5, R3, R1
 STRB R5, [R4]
 ADD R6, R0, #15
Loop:
 LDRB R1, [R0,#1]!
 LDRB R3, [R2,#1]!
 ADD R5, R3, R1
 STRB R5, [R4]
 CMP R0, R6
 BLO Loop
  LDMIA R13!, {R0-R12}
 MOV PC, R14

.data
 PinakasA:
.byte 0x20 ,0x7F ,0xFE ,0x39 ,0x16 ,0x6F ,0x30 ,0xB ,0x57 ,0x2D , 0x72 , 0x2D ,0x42 ,0x17
,0x86 ,0xA8
 PinakasB:
.byte 0x13 ,0x1 ,0x12 ,0x59 ,0x5A ,0x70 ,0x59 ,0x20 ,0x17 ,0x62 ,0x43 ,0x53 ,0x92 ,0x8C
,0xC8 ,0x43

Result:
.byte 0
```

Πίνακας Γ

Byte	Πίνακας Α	Πίνακας Β	Δεκαεξαδικό	Δεκαδικό	Μη αναμενόμενο
0	32	19	33	51	
1	127	1	80	128	
2	254	18	10	272	X
3	57	89	92	146	
4	22	90	70	112	
5	111	112	DF	223	
6	48	89	89	137	
7	11	32	2B	43	
8	87	23	6E	110	
9	45	98	8F	143	
10	114	67	B5	181	
11	45	83	80	128	
12	66	146	D4	212	
13	23	140	A3	163	
14	134	200	4E	334	X
15	168	67	EB	235	

Οι πράξεις στα byte 2,14 έχουν κανονικά αποτέλεσμα 3 ψηφία. Με την LDR η διεύθυνση της μνήμης δείχνει στο Result το οποίο είναι της μορφής Byte, δηλαδή 8 δυαδικών ψηφίων. Άρα, στη μνήμη θα αποθηκευτούν το 10 από το 110, και 4E από το 14E τα οποία στο δεκαδικό είναι 16 και 78 αντίστοιχα και για αυτό είναι μη αναμενόμενα. Μπορεί στον καταχωρητή να έχουμε σωστό αποτέλεσμα αλλά επειδή στη μνήμη έχουμε χώρο 1 byte το αποτέλεσμα βγαίνει λάθος λόγω υπερχείλισης (παραπάνω ψηφία από ότι χωράει η μνήμη).

Όσον αφορά τον κώδικα, με την LDR R0 =PinakasA και LDR R2, =PinakasB, εισάγουμε στους καταχωρητές R0,R2 τις διευθύνσεις που σηματοδοτούν οι ετικέτες PinakasA,PinakasB, όπου τα byte είναι σε δεκαεξαδική μορφή. Με τις LDRB R1, [R0, #0] και LDRB R2, [R0, #0] εισάγουμε στους καταχωρητές τις τιμές που βρίσκονται στις αντίστοιχες διευθύνσεις. Στη συνέχεια εκτελούμε LDR R6 εισάγουμε στον καταχωρητή R6 την διεύθυνση που σηματοδοτεί η ετικέτα Result(η Result πιάνει χώρο 1 byte).Επίσης γράφουμε την εντολή ADD R6, R0, #15 εισάγουμε στον R6 την διεύθυνση του τελευταίου byte(αυτό μας βοηθάει στο να βγούμε από το loop όταν προσπελαύνουμε όλα τα byte των 2 πινάκων και να μην κολλήσουμε σε αυτό επ' άπειρο). Εκτελώντας loop, προσπελαύνουμε τα δεδομένα των ετικετών και χρησιμοποιώντας #1 στο LDRB αλλάζουμε την διεύθυνση κάθε φορά κατά 1 για να προχωρήσουμε στην επόμενη σειρά. (π.χ με την LDRB R1, [R0,#1] αν είμασταν στο byte 0x7F τότε θα πάμε στο 0xFE).Για την πρόσθεση εκτελέσαμε την ADD R4, R3,R1, όπου στον R4 αποθηκεύεται το άθροισμα των καταχωρητών R3,R1. Τέλος η εντολή CMP αφού κάνουμε όλες τις πράξεις που θέλουμε μας επιτρέπει να βγούμε από το loop.

ΑΘΡΟΙΣΗ HALFWORDS

```
.arm
 .text
 .global main
 main:
 STMDB R13!, {R0-R12}
 LDR R0, =Stathera
 LDRH R1, [R0,#0]
 LDR R2, =PinakasA
 LDRH R3, [R2, #0]
 LDRH R4, [R2, #2]
 LDR R5, =PinakasB
 LDRH R6, [R5, #0]
 LDRH R7, [R5, #2]
 LDR R8, =Result
 MLA R9, R1,R4,R3
 MLA R10, R1,R7,R6
 ADD R11, R9,R10
 STRH R11, [R8]
 ADD R12, R2, #25
 Loop:
 LDRH R1, [R0,#0]!
 LDRH R3, [R2,#4]!
 LDRH R4, [R2,#2]
 LDRH R6, [R5,#4]!
 LDRH R7, [R5,#2]
 MLA R9, R1,R4,R3
 MLA R10, R1,R7,R6
 ADD R11, R10,R9
 STRH R11, [R8]
 CMP R2, R12
 BLO Loop

 LDMIA R13!, {R0-R12}
 MOV PC, R14

.data

 PinakasA:
.hword 0x20 ,0x7F ,0xFE ,0x39 ,0x16 ,0x6F ,0x30 ,0xB ,0x57 ,0x2D , 0x72 , 0x2D ,0x42 ,0x17
,0x86 ,0xA8

 PinakasB:
.hword 0x13 ,0x1 ,0x12 ,0x59 ,0x5A ,0x70 ,0x59 ,0x20 ,0x17 ,0x62 ,0x43 ,0x53 ,0x92 ,0x8C
,0xC8 ,0x43

 Stathera:
.hword 0x100

 Result:
.hword 0
```

byte	Πίνακας Α	Πίνακας Β	Πίνακας Γ
0 1	32 127	19 1	32819
2 3	254 57	18 89	37648
4 5	22 111	90 112	57200
6 7	48 11	89 32	11145
8 9	87 45	23 98	36718
10 11	114 45	67 83	32949
12 13	66 23	146 140	41940
14 15	134 168	200 67	60494

Αρχικά με την STMDB αποθηκεύουμε τους καταχωρητές που θέλουμε να χρησιμοποιήσουμε.Ύστερα με την LDR εισάγουμε στον R0, R2, R5 τη διεύθυνση που σηματοδοτεί η ετικέτα Stathera, PinakasA, PinakasB και στην συνέχεια κάνουμε LDRH δηλαδή να εισάγουμε στους καταχωρητές R1, R3, R4, R6, R7 το περιεχόμενο των θέσεων μνήμης που δείχνουν οι R0, R2, R5 σαν halfword(για παράδειγμα για τον PinakasB έχουμε LDRH R4, [R2, #2] δηλαδή στον καταχωρητή R4 εισάγουμε το περιεχόμενο της θέσης μνήμης που δείχνει ο R2(PinakasA) κατά 2 byte δηλαδή δείχνει την αμέσως επόμενη τιμή από το #0 και αυτό συμβαίνει επειδή ένα halfword έχει 16 δυαδικά ψηφία). Στην συνέχεια καλούμε ξανά την LDR δηλαδή θα εισάγουμε τον R8 την διεύθυνση που σηματοδοτεί η ετικέτα Result. Στην συνέχεια με τις εντολές MLA κάνουμε των πολλαπλασιασμό που μας ζητείται με την σταθερά 256 και το πιο σημαντικό byte και στην συνέχεια προσθέτουμε τον άλλον αριθμό και το αποτέλεσμα το αποθηκεύουμε σε έναν καταχωρητή. Στην συνέχεια με την ADD αποθηκεύουμε σε έναν καταχωρητή το αποτέλεσμα της πρόσθεσης των δύο προηγούμενων αποτελεσμάτων των MLA. Μετά πρέπει να δημιουργήσουμε μια Loop η οποία να μπορεί να εισάγει όλα τα δεδομένα των πινάκων χωρίς εμείς να χρειάζεται να το κάνουμε χειροκίνητα δηλαδή να κάνουμε το LDRH #0, #2, #4 μέχρι όσο χρειαστεί μέσα στην LOOP. Για τον λόγο αυτό έχουμε γράψει τα LDRH R1, [R0,#0]! Το οποίο κρατάει σταθερό τον R1 την τιμή 256 ,LDRH R3, [R2,#4]! Με την οποία αποθηκεύουμε στον R3 την τιμή της θέσης μνήμης που βρίσκεται 4 θέσης (halfword) μετά από αυτή στην οποία δείχνει ο R2 δηλαδή να παίρνει τις τιμές 57,111 κτλ. Και τέλος έχουμε το LDRH R4, [R2,#2] το οποίο κάνει το ίδιο με το προηγούμενο αλλά παρόλο που βάζουμε #2 ο R4 βγάζει σωστά αποτελέσματα αυτό συμβαίνει επειδή πχ για το δεύτερο halfword του πίνακα την τιμή 254 θα την πάρει πρώτα ο καταχωρητής R3 άρα ο R4 θα πάρει την αμέσως επόμενη δηλαδή 57. Μετά με την STRH αποθηκεύουμε την τιμή του R11 στην θέση μνήμης που δείχνει ο R8 σαν halfword (δηλαδή αν ψάξουμε στο Keil την θέση μνήμης κάτω δεξιά σε κάθε Loop θα εμφανίζεται στο συγκεκριμένο σημείο το αποτέλεσμα των πράξεων που αναφέρθηκαν πιο πάνω. Τέλος αφού τελειώσουν τα στοιχεία βγαίνουμε από το Loop.(από το loop βγαίνουμε με παρόμοιο τρόπο όπως πριν απλώς πρέπει να αλλάξουμε στο ADD τον αριθμό μετά την # έτσι ώστε να βγούμε την κατάλληλη στιγμή δηλαδή όταν προσπελάσουμε όλους τους απαραίτητους αριθμούς να φτάσουμε δηλαδή στο τελευταίο halfword).

Άθροιση words:

```
.arm
.text
.global main
main:
STMDB R13!, {R0-R12}

LDR R0, =PinakasA
LDR R1, [R0, #0]
LDR R2, =PinakasB
LDR R3, [R2, #0]
LDR R5, =Result

ADD R4, R1,R3
STR R4, [R5]
ADD R6, R0,#10

Loop:

LDR R1, [R0,#4]!
LDR R3, [R2,#4]!

ADD R4, R1,R3
STR R4, [R5]
CMP R0,R6
BLO Loop

LDMIA R13!, {R0-R12}
MOV PC, R14

.data

 PinakasA:
.byte 0x20 ,0x7F ,0xFE ,0x39  ,0x16 ,0x6F ,0x30 ,0xB ,0x57 ,0x2D , 0x72 , 0x2D ,0x42 ,0x17
,0x86 ,0xA8

PinakasB:
.byte 0x13 ,0x1 ,0x12 ,0x59  ,0x5A ,0x70 ,0x59 ,0x20 ,0x17 ,0x62 ,0x43 ,0x53 ,0x92 ,0x8C
,0xC8 ,0x43

Result:
.word 0
```

Δεκαεξαδικό(δεκαδικό)

Byte	Πίνακας Α	Πίνακας Β	Πίνακας Γ
0	32	19	
1	127	1	93108033
2	254	18	(2.467.332.147)
3	57	89	
4	22	90	
5	111	112	2B89DF70
6	48	89	(730.455.920)
7	11	32	
8	87	23	
9	45	98	80B58F6E
10	114	67	(2.159.382.382)
11	45	83	
12	66	146	
13	23	140	EC4EA3D4
14	134	200	(3.964.576.724)
15	168	67	

Αρχικά με την STMDB αποθηκεύουμε τους καταχωρητές που θα χρησιμοποιήσουμε. Ύστερα με την LDR εισάγουμε στον R0 και R2 τις διευθύνσεις που σηματοδοτούν οι ετικέτες PinakasA και PinakasB. Και με τις LDR R1 και R3 εισάγουμε τα αντίστοιχα word(εδώ πρέπει να τονιστεί ότι στο word το MSB πρέπει να βρίσκεται πιο αριστερά και το LSB πρέπει να βρίσκεται δεξιά ,άρα το πρώτο word θα είναι το 0x39fe7f20 (στο δεκαεξαδικό), επίσης παρατηρούμε ότι αν στις ετικέτες PinakasA και PinakasB τα δεδομένα τα δηλώσουμε σαν .byte και στην συνέχεια κάνουμε LDR αντί για LDRB τότε στον καταχωρητή θα μπουν όσα δεδομένα χρειάζονται έτσι ώστε να γεμίσει πχ στην πρώτη περίπτωση για LDR R1, [R0, #0] Τα δεδομένα 0x20 ,0x7F ,0xFE ,0x39 θα γραφτούν όπως το word που αναφέρθηκε πιο πάνω). Στην συνέχεια με την ADD κάνουμε την πρόσθεση των word που είναι αποθηκευμένα στους δυο καταχωρητές και αποθηκεύουμε το αποτέλεσμα σε έναν άλλον καταχωρητή. Τέλος κάνουμε ένα loop έτσι ώστε να εισάγουμε «αυτόματα» όλα τα δεδομένα και να γίνουν οι απαραίτητες πράξεις το μόνο που πρέπει να προσέξουμε είναι θα πρέπει τα LDR στο loop να έχουν #4 έτσι ώστε σε κάθε loop να μπαίνει το επόμενο word.

Άθροιση longword:

```
 .arm
 .text
 .global main
 main:
 STMDB R13!, {R0-R12}

 LDR R0, =PinakasA
 LDR R1, [R0, #0]
 LDR R2, =PinakasB
 LDR R3, [R2, #0]
 LDR R5, =Result

 ADDS R4, R1,R3
 STR R4, [R5]
 ADD R6, R0,#10
 Loop:
 ADDS R4, R1,R3

 LDR R1, [R0,#4]!
 LDR R3, [R2,#4]!
 ADC R4, R1,R3

 STR R4, [R5, #4]!
 CMP R0,R6
 BLO Loop

 LDMIA R13!, {R0-R12}
 MOV PC, R14

.data

 PinakasA:

.byte 0x20 ,0x7F ,0xFE ,0x39  ,0x16 ,0x6F ,0x30 ,0xB ,0x57 ,0x2D , 0x72 , 0x2D ,0x42 ,0x17
,0x86 ,0xA8
 PinakasB:
.byte 0x13 ,0x1 ,0x12 ,0x59  ,0x5A ,0x70 ,0x59 ,0x20 ,0x17 ,0x62 ,0x43 ,0x53 ,0x92 ,0x8C
,0xC8 ,0x43
 Result:
.word 0,0,0,0
```

byte	Πίνακας Α	Πίνακας Β	Πίνακας Γ (δεκαεξαδικό)
0	32	19	EC4EA3D4 80B58F6E 2B89DF70 93108033
1	127	1	
2	254	18	
3	57	89	
4	22	90	
5	111	112	
6	48	89	
7	11	32	
8	87	23	
9	45	98	
10	114	67	
11	45	83	
12	66	146	
13	23	140	
14	134	200	
15	168	67	

Η λογική για την πραγματοποίηση μιας τέτοιας πράξεις είναι να χωρίσουμε τον πίνακα όπως τον πίνακα της άθροισης words και να κάνουμε κάθε πράξη ξεχωριστά και σε περίπτωση κρατουμένου να το προσθέταμε στην επόμενη πράξη πχ.

(παράδειγμα ΑΤ)

Carry In	**1**	**0**
Number1	**0xFF000000**	**0x00000001**
Number2	**0x00FFFFFE**	**0xFFFFFFFF**
Result	**0xFFFFFFFF**	**0x00000000**

Στην συγκεκριμένη περίπτωση σε καμία πρόσθεση δεν μας μένει κρατούμενο.

Στον κώδικα εκτός από τις εντολές που αναλύσαμε και στα προηγούμενα θα πρέπει να προσθέσουμε ότι χρησιμοποιήσαμε την εντολή ADDS έτσι ώστε σε περίπτωση που έχουμε περισσευούμενο ψηφίο να το αποθηκεύσει στο Result και στην συνέχεια μέσα στο loop με την ADC να κάνει την επόμενη πράξη και να προσθέσει το ψηφίο αυτό και να αποθηκεύει το αποτέλεσμα στα τελευταία byte της ετικέτας Result(η οποία αποτελείται από 4 μηδενικά words).