i. Άθροιση bytes

```
0   .arm
1   .text
2   .global main
3
4   main:
5   STMDB R13!, {R0-R12,R14}
6
7   LDR R0, =ArrayA
8   LDR R1, =ArrayB
9   LDR R2, =ArrayC
10
11  LDR R6, =ArrayA
12  ADD R6, R6, #16
13
14  loop:
15  LDRB R4, [R0], #1
16  LDRB R5, [R1], #1
17
18  ADD R3, R4, R5
19  STRB R3, [R2], #1
20
21  CMP R0, R6
22  BLT loop
23
24  LDMIA R13!, {R0-R12,R14}
25
26  .data
27
28  ArrayA:
29  .byte 0x20,0x7F,0xFE,0x39
30  .byte 0x16,0x6F,0x30,0x0B
31  .byte 0x57,0x2D,0x72,0x2D
32  .byte 0x42,0x17,0x86,0xA8
33
34  ArrayB:
35  .byte 0x13,0x01,0x12,0x59
36  .byte 0x5A,0x70,0x39,0x20
37  .byte 0x17,0x62,0x43,0x53
38  .byte 0x92,0x8C,0xC8,0x43
39
40  ArrayC:
41  .byte 0x00,0x00,0x00,0x00
42  .byte 0x00,0x00,0x00,0x00
43  .byte 0x00,0x00,0x00,0x00
44  .byte 0x00,0x00,0x00,0x00
```

Στους R0, R1, R2 αποθηκεύονται οι διευθύνσεις των θέσεων μνήμης που σηματοδοτούνται από τις ετικέτες ArrayA, ArrayB, ArrayC αντίστοιχα. Πρακτικά οι καταχωρητές χρησιμεύουν ως δείκτες των πρώτων στοιχείων των πινάκων A, B και C.

Ο R6 αποθηκεύει την θέση μνήμης μέχρι την οποία πρέπει να φτάσει ο R0 ώστε να προσπελάσει όλον τον πίνακα A κατά τη διάρκεια της επανάληψης.

Στην επανάληψη αποθηκεύεται στον R4 το περιεχόμενο της θέσης μνήμης του ενός byte την οποία έχει αποθηκευμένη ο R0 (πρακτικά το περιεχόμενο του πίνακα A) και αντίστοιχα στον R5 για τον πίνακα B και αυξάνει τους R0 & R1 κατά ένα για την προσπέλαση των επόμενων στοιχείων.
Πχ. θα γίνει η πρόσθεση του $20_{(16)}$ και του $13_{(16)} = 33_{(16)}$

Στον R3 αποθηκεύεται το αποτέλεσμα της πρόσθεσης των στοιχείων και αποθηκεύεται ως ένα byte[1] στη θέση μνήμης που η διεύθυνσή της είναι αποθηκευμένη στον καταχωρητή R2, η οποία στη συνέχεια αυξάνεται κατά ένα για το επόμενο άθροισμα.

Κάθε φορά συγκρίνονται οι καταχωρητές R0 (δείκτης A) και R6 (αναμενόμενο τέλος του δείκτη - αρχή του πίνακα B) και όσο η εικονική διαφορά R0-R6 είναι αρνητική (δλδ υπάρχει N=1 στον CPSR), ο PC επιστρέφει στην αρχή της επανάληψης εξ αιτίας της εντολής BLT (= Branch Less Than) *(παρόμοια μπορούμε με την BNE = Branch Not Equal)*.

[1] Στη μνήμη αποθηκεύεται ένα byte μόνο, στα αποτελέσματα 110 και 14E υπάρχει overflow.

byte	Πίνακας A		Πίνακας B		Πίνακας Γ		Μη αναμεν.
	(10)	(16)	(10)	(16)	(10)	(16)	
0	32	20	19	13	51	33	
1	127	7F	1	01	128	80	
2	254	FE	18	12	272	110	X
3	57	39	89	59	146	92	
4	22	16	90	5A	112	70	
5	111	6F	112	70	223	DF	
6	48	30	89	39	137	89	
7	11	0B	32	20	43	2B	
8	87	57	23	17	110	6E	
9	45	2D	98	62	143	87	
10	114	72	67	43	181	B5	
11	45	2D	83	53	128	80	
12	66	42	146	92	212	D4	
13	23	17	140	8C	163	A3	
14	134	86	200	C8	334	14E	X
15	168	A8	67	43	235	EB	

ii. Άθροιση halfwords

```
0   .arm
1   .text
2   .global main
3
4   main:
5   STMDB R13!, {R0-R12,R14}
6
7   LDR R0, =ArrayA
8   LDR R1, =ArrayB
9   LDR R2, =ArrayC
10
11  LDR R6, =ArrayA
12  ADD R6, R6, #16
13
14  loop:
15  LDRH R4, [R0], #2
16  LDRH R5, [R1], #2
17
18  ADD R3, R4, R5
19  STRH R3, [R2], #2
20
21  CMP R0, R6
22  BLT loop
23
24  LDMIA R13!, {R0-R12,R14}
25
26  .data
27
28  ArrayA:
29  .byte 0x20,0x7F,0xFE,0x39
30  .byte 0x16,0x6F,0x30,0x0B
31  .byte 0x57,0x2D,0x72,0x2D
32  .byte 0x42,0x17,0x86,0xA8
33
34  ArrayB:
35  .byte 0x13,0x01,0x12,0x59
36  .byte 0x5A,0x70,0x39,0x20
37  .byte 0x17,0x62,0x43,0x53
38  .byte 0x92,0x8C,0xC8,0x43
39
40  ArrayC:
41  .hword 0x00,0x00,0x00,0x00
42  .hword 0x00,0x00,0x00,0x00
```

Στους R0, R1, R2 αποθηκεύονται οι διευθύνσεις των θέσεων μνήμης που σηματοδοτούνται από τις ετικέτες ArrayA, ArrayB, ArrayC αντίστοιχα. Πρακτικά οι καταχωρητές χρησιμεύουν ως δείκτες των πρώτων στοιχείων των πινάκων A, B και C.

Ο R6 αποθηκεύει την θέση μνήμης μέχρι την οποία πρέπει να φτάσει ο R0 ώστε να προσπελάσει όλον τον πίνακα A κατά τη διάρκεια της επανάληψης.

Στην επανάληψη αποθηκεύεται στον R4 το περιεχόμενο της θέσης μνήμης των δύο byte (halfword) την οποία έχει αποθηκευμένη ο R0 (πρακτικά το περιεχόμενο του πίνακα A) και αντίστοιχα στον R5 για τον πίνακα B και αυξάνει τους R0 & R1 κατά δύο για την προσπέλαση των επόμενων στοιχείων.

Πχ. θα γίνει η πρόσθεση του $7F20_{(16)}$ και του $0113_{(16)} = 8033_{(16)}$

Το πρόθεμα H στο LDR καθορίζει πως πρόκειται τα 2 επόμενα bytes αποτελούν halfword, άρα τα bytes της μνήμης θα βρεθούν στη σωστή θέση με βάση την αξία τους, ανεξάρτητα της little endian αρχιτεκτονικής στη μνήμη που τα προτάσσει αντίθετα.

Στον R3 αποθηκεύεται το αποτέλεσμα της πρόσθεσης των στοιχείων (halfword) και αποθηκεύεται ως δύο byte στη θέση μνήμης που η διεύθυνσή της είναι αποθηκευμένη στον καταχωρητή R2, η οποία στη συνέχεια αυξάνεται κατά δύο για το επόμενο άθροισμα.

Κάθε φορά συγκρίνονται οι καταχωρητές R0 (δείκτης A) και R6 (αναμενόμενο τέλος του δείκτη) και όσο η εικονική διαφορά R0-R6 είναι αρνητική (N=1 στον CPSR), ο PC επιστρέφει στην επανάληψη εξ αιτίας της εντολής BLT (= Branch Less Than) *(παρόμοια μπορούμε με την BNE = Branch Not Equal)*.

byte	Πίνακας Α		Πίνακας Β		Πίνακας Γ	
	(10)	(16)	(10)	(16)	(10)	(16)
0 1	32 127	20 7F	19 1	13 01	32 + 127*256 + 19 + 256	80 33
2 3	254 57	FE 39	18 89	12 59	254 + 57*256 + 18 + 89*256	93 10
4 5	22 111	16 6F	90 112	5A 70	22 + 111*256 + 90 + 112*256	DF 70
6 7	48 11	30 0B	89 32	39 20	48 + 11*256 + 89 + 32*256	2B 69
8 9	87 45	57 2D	23 98	17 62	87 + 45*256 + 23 + 98*256	8F 3E
10 11	114 45	72 2D	67 83	43 53	114 + 45*256 + 67 + 83*256	80 B5
12 13	66 23	42 17	146 140	92 8C	66 + 23*256 + 146 + 140*256	A3 D4
14 15	134 168	86 A8	200 67	C8 43	134 + 168*256 + 200 + 67*256	EC 4E

iii. Άθροιση words

```
0   .arm
1   .text
2   .global main
3
4   main:
5   STMDB R13!, {R0-R12,R14}
6
7   LDR R0, =ArrayA
8   LDR R1, =ArrayB
9   LDR R2, =ArrayC
10
11  LDR R6, =ArrayA
12  ADD R6, R6, #16
13
14  loop:
15  LDR R4, [R0], #4
16  LDR R5, [R1], #4
17
18  ADD R3, R4, R5
19  STR R3, [R2], #4
20
21  CMP R0, R6
22  BLT loop
23
24  LDMIA R13!, {R0-R12,R14}
25
26  .data
27
28  ArrayA:
29  .byte 0x20,0x7F,0xFE,0x39
30  .byte 0x16,0x6F,0x30,0x0B
31  .byte 0x57,0x2D,0x72,0x2D
32  .byte 0x42,0x17,0x86,0xA8
33
34  ArrayB:
35  .byte 0x13,0x01,0x12,0x59
36  .byte 0x5A,0x70,0x39,0x20
37  .byte 0x17,0x62,0x43,0x53
38  .byte 0x92,0x8C,0xC8,0x43
39
40  ArrayC:
41  .word 0x00,0x00,0x00,0x00
```

Στους R0, R1, R2 αποθηκεύονται οι διευθύνσεις των θέσεων μνήμης που σηματοδοτούνται από τις ετικέτες ArrayA, ArrayB, ArrayC αντίστοιχα. Πρακτικά οι καταχωρητές χρησιμεύουν ως δείκτες των πρώτων στοιχείων των πινάκων A, B και C.

Ο R6 αποθηκεύει την θέση μνήμης μέχρι την οποία πρέπει να φτάσει ο R0 ώστε να προσπελάσει όλον τον πίνακα A κατά τη διάρκεια της επανάληψης.

Στην επανάληψη αποθηκεύεται στον R4 το περιεχόμενο της θέσης μνήμης των τεσσάρων byte (word) την οποία έχει αποθηκευμένη ο R0 (πρακτικά το περιεχόμενο του πίνακα A) και αντίστοιχα στον R5 για τον πίνακα B και αυξάνει τους R0 & R1 κατά τέσσερα για την προσπέλαση των επόμενων στοιχείων.

Πχ. θα γίνει η πρόσθεση του $39FE7F20_{(16)}$ και του $59120113_{(16)}$ = $93108033_{(16)}$

Στον R3 αποθηκεύεται το αποτέλεσμα της πρόσθεσης των στοιχείων (word) και αποθηκεύεται ως τέσσερα byte στη θέση μνήμης που η διεύθυνσή της είναι αποθηκευμένη στον καταχωρητή R2, η οποία στη συνέχεια αυξάνεται κατά τέσσερα για το επόμενο άθροισμα.

Κάθε φορά συγκρίνονται οι καταχωρητές R0 (δείκτης A) και R6 (αναμενόμενο τέλος του δείκτη) και όσο η εικονική διαφορά R0-R6 είναι αρνητική (N=1 στον CPSR), ο PC επιστρέφει στην επανάληψη εξ αιτίας της εντολής BLT (= Branch Less Than) *(παρόμοια μπορούμε με την BNE = Branch Not Equal)*.

byte	Πίνακας Α		Πίνακας Β		Πίνακας Γ	
	(10)	(16)	(10)	(16)	(10)	(16)
0	32	20	19	13	(για 0-3 bytes): (32 + 127*256 + 254*256^2 + 57*256^3) + (19 + 1*256 + 18*256^2 + 89*256^3) (αντίστοιχα για τα επόμενα)	93 10 80 33
1	127	7F	1	01		
2	254	FE	18	12		
3	57	39	89	59		
4	22	16	90	5A		2B 69 DF 70
5	111	6F	112	70		
6	48	30	89	39		
7	11	0B	32	20		
8	87	57	23	17		80 B5 8F 6E
9	45	2D	98	62		
10	114	72	67	43		
11	45	2D	83	53		
12	66	42	146	92		EC 4E A3 D4
13	23	17	140	8C		
14	134	86	200	C8		
15	168	A8	67	43		

iv. Άθροιση longwords

```
0     .arm
1     .text
2     .global main
3
4     main:
5     STMDB R13!, {R0-R12,R14}
6
7     LDR R0, =ArrayA
8     LDR R1, =ArrayB
9     LDR R2, =ArrayC
10
11    LDR R6, =ArrayA
12    ADD R6, R6, #16
13
14    MOV R10, #0
15
16    loop:
17    LDR R4, [R0], #4
18    LDR R5, [R1], #4
19
20    ADDS R3, R4, R5
21
22    ADD R3, R3, R10
23
24    MOVCS R10, #1
25    MOVCC R10, #0
26
27    STR R3, [R2], #4
28
29    CMP R0, R6
30    BLT loop
31
32    LDMIA R13!, {R0-R12,R14}
33
34    .data
35
36    ArrayA:
37    .word 0x39FE7F20
38    .word 0x0B306F16
39    .word 0xFFFFFFFF
40    .word 0xA8861742
41
42    ArrayB:
43    .word 0x59120113
44    .word 0x2039705A
45    .word 0x53436217
46    .word 0x43C88C92
47
48    ArrayC:
49    .word 0x00,0x00,0x00,0x00
50
```

Στους R0, R1, R2 αποθηκεύονται οι διευθύνσεις των θέσεων μνήμης που σηματοδοτούνται από τις ετικέτες ArrayA, ArrayB, ArrayC αντίστοιχα. Πρακτικά οι καταχωρητές χρησιμεύουν ως δείκτες των πρώτων στοιχείων των πινάκων A, B και C.

Ο R6 αποθηκεύει την θέση μνήμης μέχρι την οποία πρέπει να φτάσει ο R0 ώστε να προσπελάσει όλον τον πίνακα A κατά τη διάρκεια της επανάληψης. Στον R10 θα αποθηκευτούν τα κρατούμενα των προσθέσεων.

Στην επανάληψη αποθηκεύονται στον R4 τα περιεχόμενα της θέσης μνήμης (words) που έχει αποθηκεύσει ο R0 (δλδ το περιεχόμενο του πίνακα A) και αντίστοιχα στον R5 για τον πίνακα B και αυξάνει τον R0 & R1 κατά 4 (μιας και ένα word αποθηκεύεται σε τέσσερις θέσεις μνήμης) για την προσπέλαση των επόμενων στοιχείων.

Στον R3 αποθηκεύεται το αποτέλεσμα της πρόσθεσης των στοιχείων (words) και το C flag του CPSR τίθεται ως 1 ή 0 ανάλογα με το αν η πράξη παρήγαγε κρατούμενο. Η τιμή αυτή αποθηκεύεται αναλόγως στον R10: αν C=1, ισχύει η CS συνθήκη και αν C=0, η CC. Στο αποτέλεσμα του R3 προστίθεται ο καταχωρητής R10, και το συνολικό αποτέλεσμα αποθηκεύεται στη θέση μνήμης που η διεύθυνσή της είναι αποθηκευμένη στον καταχωρητή R2, η οποία στη συνέχεια αυξάνεται κατά 4 για το επόμενο άθροισμα.

* Δεν χρησιμοποιήσαμε την ADC(S) αλλά αποθηκεύσαμε χειροκίνητα το C, διότι λόγω της σύγκρισης του CMP τα κρατούμενα των προηγούμενων προσθέσεων χάνονταν.

Κάθε φορά συγκρίνονται οι καταχωρητές R0 (δείκτης A) και R6 (αναμενόμενο τέλος του δείκτη) και όσο η εικονική διαφορά R0-R6 είναι αρνητική (N=1 στον CPSR), ο PC επιστρέφει στην επανάληψη εξ αιτίας της εντολής BLT (= Branch Less Than) *(* ή BNE = Branch Not Equal) (παρόμοια μπορούμε με την BNE = Branch Not Equal)*.

Αφού το 1 word = 4 bytes θα χρειάζονταν $4 \cdot 4 = 16$ προσθέσεις χωρίς να περιλαμβάνουμε τα κρατούμενα.