2Η ΕΡΓΑΣΙΑ ΕΡΓΑΣΤΗΡΙΟΥ ΒΑΣΙΚΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Συγγραφείς: Βέργος Γεώργιος ΑΜ :1072604 , Τσούλος Βασίλειος ΑΜ:1072605.

1ο ΕΡΩΤΗΜΑ

Το πρώτο ερώτημα της άσκησης ζητά να προσθέσουμε τα στοιχεία ενός πίνακα Α με τα αντίστοιχα στοιχεία ενός πίνακα Β και τα αποτελέσματα να αποθηκευτούν σε έναν άλλο πίνακα Γ. Τα 16 στοιχεία του πίνακα Α όπως και του Β είναι BYTES δηλαδή πληροφορίες/αριθμοί των 8 δυαδικών ψηφίων(1 byte). Η αρχιτεκτονική του συστήματος μας είναι little endian δηλαδή το λιγότερο σημαντικό byte αποθηκεύεται στη θέση μνήμης με τη μικρότερη διεύθυνση ενώ το περισσότερο σημαντικό byte αποθηκεύεται στην υψηλότερη διεύθυνση στη μνήμη. Το αποτέλεσμα της πρόσθεσης που αναμένουμε να είναι και αυτό byte θα αποθηκεύεται σε έναν πίνακα Γ, δηλαδή θα αποθηκεύω bytes στη μνήμη(ξεκινώντας από μία συγκεκριμένη διεύθυνση) διαδοχικά με το λιγότερο σημαντικό byte να αποθηκεύεται στη μικρότερη διεύθυνση στη μνήμη ενώ το πιο σημαντικό byte του πίνακα στην υψηλότερη διεύθυνση στη μνήμη. Ο κώδικας που υλοποίει το παραπάνω είναι ο παρακάτω:

```
.arm

.text

.global main

main:

STMDB R13!, {R0-R4} @ αποθηκεύω τους καταχωρητές που θα χρησιμοποιήσω στη σωρό του συστήματος.
```

MOV R3, #0 @Χρησιμοποιώ τον καταχωρητή R3 ως μετρητή για να προσπελαύνω τα στοιχεία των πινάκων Α,Β και να μπορώ να αποθηκεύω τα αντίστοιχα αποτελέσματα στον Γ.

LOOP: @Ετικέτα LOOP που τη χρησιμοποιώ για επαναλήψεις.

LDR R0,=PinakasA @Αποθηκεύω στον καταχωρητή R0 τη διεύθυνση της μνήμης που σηματοδοτεί η ετικέτα PinakasA

LDR R1,=PinakasB @Αποθηκεύω στον καταχωρητή R1 τη διεύθυνση της μνήμης που σηματοδοτεί η ετικέτα PinakasB. Εκεί θα αποθηκευτεί όλη η πληροφορία που περιέχει ο πίνακας Β.

LDR R4,=Pinakasgamma Αποθηκεύω στον καταχωρητή R4 τη διεύθυνση της μνήμης που σηματοδοτεί η ετικέτα Pinakasgamma. Εκεί θα αποθηκευτεί όλη η πληροφορία που περιέχει ο πίνακας Γ(τον αρχικοποιώ με 0) και μετά η πληροφορία τα bytes προκύπτουν από την πρόσθεση των bytes από τους Α και Β.

LDRB R0,[R0,R3] @ Αποθηκεύω στον καταχωρητή R0 ένα byte πληροφορίας από τη περιοχή στη μνήμη που δείχνει ο R0+<τιμή καταχωρητή R3> θέσεις μπροστά.

LDRB R1,[R1,R3] @ Αποθηκεύω στον καταχωρητή R1 ένα byte πληροφορίας από τη περιοχή στη μνήμη που δείχνει ο R1+<τιμή καταχωρητή R3> θέσεις μπροστά.

ADD R2,R1,R0 @Προσθέτω τα περιεχόμενα των καταχωρητών R1 και R0(τρέχον byte των πινάκων Α και Β αντίστοιχα) και αποθηκεύω το αποτέλεσμα στον καταχωρητή R2.

STRB R2,[R4,R3] @Αποθηκεύω ένα byte πληροφορίας του περιεχομένου του καταχωρητή R2 στη περιοχή της μνήμης που δείχνει ο καταχωρητής R4+<τιμή καταχωρητή R3> θέσεις μπροστά(Πίνακας Γ).

ADD R3,R3,#1 @Αυξάνω την τιμή του καταχωρητή-μετρητή R3 κατά 1. Έστω για παράδειγμα ότι προσπελαύνω τη περιοχή στη μνήμη που δείχνει ο καταχωρητής R0. Επειδή το πρώτο byte που θα αποθηκεύσω σε καταχωρητή αποτελείται από 8 δυαδικά ψηφία(1 byte) θα καταλαμβάνει τις διευθύνσεις <τιμή καταχωρητή R0+0 οπότε στην επόμενη προσπέλαση το byte που θα επεξεργαστώ θα ξεκινά από τη διεύθυνση εκεί που δείχνει ο καταχωρητής R0 +1.

CMP R3,#16 @Επειδή όλοι οι πίνακες έχουν 16 bytes ο καθένας και αυξάνω σε κάθε επανάληψη την τιμή του R3 κατά 1 η συνθήκη τερματισμού του βρόχου είναι ο R3 να ισούται με το 16(10 HEX).

BNE LOOP @ Από το branch if not equal αν η παραπάνω σύγκριση παράγει 0(ψευδής) εκτελεί άλμα στη διεύθυνση που σημειώνεται από την ετικέτα LOOP

MOV PC,R14 @Μεταφέρουμε τη ροή της εκτέλεσης στο σημείο όπου κλήθηκε η υπορουτίνα LOOP.

.data

PinakasA:

.byte 32,127,254,57,22,111,48,11,87,45,114,45,66,23,134,168

PinakasB:

.byte 19,1,18,89,90,112,89,32,23,98,67,83,146,140,200,67

Pinakasgamma:

.byte 0

Παρατίθεται ο πίνακας των bytes που αθροίζονται και τα αντίστοιχα αποτελέσματα στον πίνακα Γ.

byte	Πίνακας A	Πίνακας B	Πίνακας Γ		Μη αναμενόμενο
			Δεκαεξαδικό	Δεκαδικό	
0	32	19	33	51	
1	127	1	80	128	
2	254	18	10	16	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	78	X
15	168	67	EB	235	

Παρατηρούμε πως δύο προσθέσεις παρουσιάζουν πρόβλημα κατά την άθροιση bytes και συγκεκριμένα οι:254+18(FE+12) και η 134+200(86+C8). Αυτό ήταν αναμενόμενο να συμβεί μιας και τα αποτελέσματα των προσθέσεων αυτών απαιτούν παραπάνω από 8 δυαδικά ψηφία(δηλαδή 1 byte) για να αναπαρασταθούν. Ωστόσο η άσκηση ζητά τα στοιχεία του πίνακα Γ είναι των 8 δυαδικών ψηφίων. Επειδή εμείς μέσω των εντολών STRB θα αποθηκεύουμε μόνο τα πρώτα 8 bits του αποτελέσματος για αποτελέσματα άνω των 8 bits φυσικό είναι το αποτέλεσμα να είναι μη αναμενόμενο. Τα σωστά αποτελέσματα των πράξεων αυτών είναι 254+18=272(110

HEX) και 134+200=334(14E HEX). Υπενθυμίζω πως ένα byte πληροφορίας(8 δυαδικά ψηφία/δύο δεκαεξαδικά ψηφία) μπορούν να αναπαραστήσουν τους αριθμούς 0-255(256 αριθμοί) δηλαδή από 00-FF.

2ο ΕΡΩΤΗΜΑ

Το δεύτερο ερώτημα της άσκησης ζητά να προσθέσουμε τα στοιχεία ενός πίνακα Α με τα αντίστοιχα στοιχεία ενός πίνακα Β και τα αποτελέσματα να αποθηκευτούν σε έναν άλλο πίνακα Γ. Τα 8 στοιχεία του πίνακα Α όπως και του Β είναι halfwords δηλαδή πληροφορίες/αριθμοί των 16 δυαδικών ψηφίων(δύο bytes). Η αρχιτεκτονική του συστήματος μας είναι little endian δηλαδή το λιγότερο σημαντικό byte κάθε halfword αποθηκεύεται στη θέση μνήμης με τη μικρότερη διεύθυνση ενώ το περισσότερο σημαντικό byte του(το δεύτερο) αποθηκεύεται στην αμέσως υψηλότερη διεύθυνση στη μνήμη. Το αποτέλεσμα της πρόσθεσης που είναι και αυτό halfword θα αποθηκεύεται σε έναν πίνακα Γ, δηλαδή θα αποθηκεύω halfwords στη μνήμη(ξεκινώντας από μία συγκεκριμένη διεύθυνση) διαδοχικά με τα bytes τους να αποθηκεύονται κατά little endian.

Ακολουθεί ο πηγαίος κώδικας του 2ου ερωτήματος:

```
.arm

.text

.global main

main:

STMDB R13!, {R0-R4} @ αποθηκεύω τους καταχωρητές που θα χρησιμοποιήσω στη σωρό του συστήματος.
```

MOV R3, #0 @Χρησιμοποιώ τον καταχωρητή R3 ως μετρητή για να προσπελαύνω τα στοιχεία των πινάκων Α,Β και να μπορώ να αποθηκεύω τα αντίστοιχα αποτελέσματα στον Γ.

LOOP: @Ετικέτα LOOP που τη χρησιμοποιώ για επαναλήψεις.

LDR R0,=PinakasA @Αποθηκεύω στον καταχωρητή R0 τη διεύθυνση της μνήμης που σηματοδοτεί η ετικέτα PinakasA. Εκεί θα αποθηκευτεί όλη η πληροφορία που περιέχει ο πίνακας Α.

LDR R1,=PinakasB @Αποθηκεύω στον καταχωρητή R1 τη διεύθυνση της μνήμης που σηματοδοτεί η ετικέτα PinakasB. Εκεί θα αποθηκευτεί όλη η πληροφορία που περιέχει ο πίνακας Β.

LDR R4,=Pinakasgamma @Αποθηκεύω στον καταχωρητή R4 τη διεύθυνση της μνήμης που σηματοδοτεί η ετικέτα Pinakasgamma. Εκεί θα αποθηκευτεί όλη η πληροφορία που περιέχει ο πίνακας Γ(τον αρχικοποιώ με 0) και μετά η πληροφορία(τα halfwords προκύπτουν από την πρόσθεση των halfwords από τους Α και Β.

LDRH R0,[R0,R3] @ Αποθηκεύω στον καταχωρητή R0 ένα halfword πληροφορίας από τη περιοχή στη μνήμη που δείχνει ο R0+<τιμή καταχωρητή R3> θέσεις.

LDRH R1,[R1,R3] @ Αποθηκεύω στον καταχωρητή R1 ένα halfword πληροφορίας από τη περιοχή στη μνήμη που δείχνει ο R1+<τιμή καταχωρητή R3> θέσεις μπροστά. Αυτή η εντολή όπως και η παραπάνω θυμίζει από υψηλότερου επιπέδου γλώσσες προγραμματισμού(όπως η C) τη προσπέλαση πίνακα με την εξής σχέση να ισχύει: &array[i]=array+ i με το ρόλο του I εδώ να παίζει ο καταχωρητής R3.

ADD R2,R1,R0 @ Προσθέτω τα περιεχόμενα των καταχωρητών R1 και R0(τρέχον halfword των πινάκων Α και Β αντίστοιχα και αποθηκεύω το αποτέλεσμα στον καταχωρητή R2.

STRH R2,[R4,R3] @Αποθηκεύω ένα halfword πληροφορίας του περιεχομένου του καταχωρητή R2 και το αποθηκεύω στη περιοχή της μνήμης που δείχνει ο καταχωρητής R4+<τιμή καταχωρητή R3> θέσεις μπροστά(Πίνακας Γ).

ADD R3,R3,#2 @Αυξάνω την τιμή του καταχωρητή-μετρητή R3 κατά δύο. Το 2 χρησιμοποιείται γιατί κάθε halfword αποτελείται από 2 bytes. Έστω για παράδειγμα ότι προσπελαύνω τη περιοχή στη μνήμη που δείχνει ο καταχωρητής R0. Επειδή το πρώτο halfword που θα αποθηκεύσω σε καταχωρητή αποτελείται από δύο bytes , τα 2 bytes του θα καταλαμβάνουν τις διευθύνσεις <τιμή καταχωρητή R0+0>(το λιγότερο σημαντικό) και <τιμή καταχωρητή R0 +1>(το πιο σημαντικό του) οπότε στην επόμενη προσπέλαση το halfword που θα επεξεργαστώ θα ξεκινά από τη διεύθυνση εκεί που δείχνει ο καταχωρητής R0 +2.

CMP R3,#16 @Επειδή όλοι οι πίνακες έχουν 8 halfwords ο καθένας και αυξάνω σε κάθε επανάληψη την τιμή του R3 κατά 2 η συνθήκη τερματισμού του βρόχου είναι ο R3 να ισούται με το 16(10 HEX).

BNE LOOP @ Από το branch if not equal αν η παραπάνω σύγκριση παράγει 0(ψευδής) εκτελεί άλμα στη διεύθυνση που σημειώνεται από την ετικέτα LOOP.

MOV PC,R14 @Μεταφέρουμε τη ροή της εκτέλεσης στο σημείο όπου κλήθηκε η υπορουτίνα LOOP.

.data

PinakasA:

.byte 32,127,254,57,22,111,48,11,87,45,114,45,66,23,134,168

@ Τοποθετώ σε διαδοχικές θέσεις στη περιοχή της μνήμης που σηματοδοτείται από την ετικέτα PinakasA τα παραπάνω bytes βάσει little endian.

PinakasB:

.byte 19,1,18,89,90,112,89,32,23,98,67,83,146,140,200,67

@ Τοποθετώ σε διαδοχικές θέσεις στη περιοχή της μνήμης που σηματοδοτείται από την ετικέτα PinakasA τα παραπάνω bytes βάσει little endian.

Pinakasgamma:

.byte 0

Στις περιοχές της μνήμης που αντιπροσωπεύουν τους πίνακες Α,Β,Γ τοποθετώ bytes γιατί είναι η μικρότερη μονάδα πληροφορίας που πραγματεύεται η άσκηση και είναι εύκολος ο χειρισμός της σε σχέση με το να αποθήκευα τις πληροφορίες ως words ή halfwords.

3^{Ο} ΕΡΩΤΗΜΑ

Το τρίτο ερώτημα της άσκησης ζητά να προσθέσουμε τα στοιχεία ενός πίνακα Α με τα αντίστοιχα στοιχεία ενός πίνακα Β και τα αποτελέσματα να αποθηκευτούν σε έναν άλλο πίνακα Γ. Τα 4 στοιχεία του πίνακα Α όπως και του Β είναι words δηλαδή πληροφορίες/αριθμοί των 32 δυαδικών ψηφίων(4 bytes). Η αρχιτεκτονική του συστήματος μας είναι little endian δηλαδή το λιγότερο σημαντικό byte κάθε word αποθηκεύεται στη θέση μνήμης με τη μικρότερη διεύθυνση ενώ το αμέσως πιο σημαντικό byte του(το δεύτερο) αποθηκεύεται στην αμέσως υψηλότερη διεύθυνση στη μνήμη κι ούτω καθεξής. Το αποτέλεσμα της πρόσθεσης που είναι και αυτό word θα αποθηκεύεται σε έναν πίνακα Γ, δηλαδή θα αποθηκεύω words στη μνήμη(ξεκινώντας από μία συγκεκριμένη διεύθυνση)

διαδοχικά με το λιγότερο σημαντικό word να αποθηκεύεται στις 4 μικρότερες διεύθυνσεις στη μνήμη ενώ το πιο σημαντικό word του πίνακα στις 4 υψηλότερες διεύθυνσεις στη μνήμη.

Ακολουθεί ο πηγαίος κώδικας του 3ου ερωτήματος:

.arm

.text

.global main

main:

STMDB R13!, {R0-R4} @ αποθηκεύω τους καταχωρητές που θα χρησιμοποιήσω στη σωρό του συστήματος.

MOV R3, #0 @Χρησιμοποιώ τον καταχωρητή R3 ως μετρητή για να προσπελαύνω τα στοιχεία των πινάκων Α,Β και να μπορώ να αποθηκεύω τα αντίστοιχα αποτελέσματα στον Γ.

LOOP: @Ετικέτα LOOP που τη χρησιμοποιώ για επαναλήψεις.

LDR R0,=PinakasA @Αποθηκεύω στον καταχωρητή R0 τη διεύθυνση της μνήμης που σηματοδοτεί η ετικέτα PinakasA. Εκεί θα αποθηκευτεί όλη η πληροφορία που περιέχει ο πίνακας Α.

LDR R1,=PinakasB @Αποθηκεύω στον καταχωρητή R1 τη διεύθυνση της μνήμης που σηματοδοτεί η ετικέτα PinakasB. Εκεί θα αποθηκευτεί όλη η πληροφορία που περιέχει ο πίνακας Β.

LDR R4,=Pinakasgamma @Αποθηκεύω στον καταχωρητή R4 τη διεύθυνση της μνήμης που σηματοδοτεί η ετικέτα Pinakasgamma. Εκεί θα αποθηκευτεί όλη η πληροφορία που περιέχει ο πίνακας Γ(τον αρχικοποιώ με 0) και μετά η

πληροφορία(τα words προκύπτουν από την πρόσθεση των words από τους A και B).

LDR R0,[R0,R3] @ Αποθηκεύω στον καταχωρητή R0 ένα word πληροφορίας από τη περιοχή στη μνήμη που δείχνει ο R0+<τιμή καταχωρητή R3> θέσεις.

LDR R1,[R1,R3] @ Αποθηκεύω στον καταχωρητή R1 ένα word πληροφορίας από τη περιοχή στη μνήμη που δείχνει ο R1+<τιμή καταχωρητή R3> θέσεις μπροστά. Αυτή η εντολή όπως και η παραπάνω θυμίζει από υψηλότερου επιπέδου γλώσσες προγραμματισμού(όπως η C) τη προσπέλαση πίνακα με την εξής σχέση να ισχύει: &array[i]=array+ i με το ρόλο του I εδώ να παίζει ο καταχωρητής R3.

ADD R2,R1,R0 @ Προσθέτω τα περιεχόμενα των καταχωρητών R1 και R0(τρέχον word των πινάκων A και B αντίστοιχα και αποθηκεύω το αποτέλεσμα στον καταχωρητή R2.

STR R2,[R4,R3] @Αποθηκεύω ένα word πληροφορίας του περιεχομένου του καταχωρητή R2 και το αποθηκεύω στη περιοχή της μνήμης που δείχνει ο καταχωρητής R4+<τιμή καταχωρητή R3> θέσεις μπροστά(Πίνακας Γ).

ADD R3,R3,#4 @Αυξάνω την τιμή του καταχωρητή-μετρητή R3 κατά 4. Το 4 χρησιμοποιείται γιατί κάθε word αποτελείται από 4 bytes. Έστω για παράδειγμα ότι προσπελαύνω τη περιοχή στη μνήμη που δείχνει ο καταχωρητής R0. Επειδή το πρώτο word που θα αποθηκεύσω σε καταχωρητή αποτελείται από 4 bytes , τα 4 bytes του θα καταλαμβάνουν διαδοχικα τις διευθύνσεις <τιμή καταχωρητή R0+0>(το λιγότερο σημαντικό) , <τιμή καταχωρητή R0 +1>,<τιμή καταχωρητή R0+2>,<τιμή καταχωρητή R0+3>, οπότε στην επόμενη προσπέλαση το word

που θα επεξεργαστώ θα ξεκινά από τη διεύθυνση εκεί που δείχνει ο καταχωρητής R0 +4.

CMP R3,#16 @Επειδή όλοι οι πίνακες έχουν 4 words ο καθένας και αυξάνω σε κάθε επανάληψη την τιμή του R3 κατά 4 η συνθήκη τερματισμού του βρόχου είναι ο R3 να ισούται με το 16(10 HEX).

BNE LOOP @ Από το branch if not equal αν η παραπάνω σύγκριση παράγει 0(ψευδής) εκτελεί άλμα στη διεύθυνση που σημειώνεται από την ετικέτα LOOP.

MOV PC,R14 @Μεταφέρουμε τη ροή της εκτέλεσης στο σημείο όπου κλήθηκε η υπορουτίνα LOOP.

.data

PinakasA:

.byte 32,127,254,57,22,111,48,11,87,45,114,45,66,23,134,168

@ Τοποθετώ σε διαδοχικές θέσεις στη περιοχή της μνήμης που σηματοδοτείται από την ετικέτα PinakasA τα παραπάνω bytes βάσει little endian.

PinakasB:

.byte 19,1,18,89,90,112,89,32,23,98,67,83,146,140,200,67

@ Τοποθετώ σε διαδοχικές θέσεις στη περιοχή της μνήμης που σηματοδοτείται από την ετικέτα PinakasA τα παραπάνω bytes βάσει little endian.

Pinakasgamma:

.byte 0

Στις περιοχές της μνήμης που αντιπροσωπεύουν τους πίνακες Α,Β,Γ τοποθετώ bytes γιατί είναι η μικρότερη μονάδα πληροφορίας που πραγματεύεται η άσκηση(σε σχέση με halfwords,words,longwords) και είναι εύκολος ο χειρισμός της

σε σχέση με το να αποθήκευα τις πληροφορίες ως words ή halfwords.

4Ο ΕΡΩΤΗΜΑ

Το 4ο ερώτημα της άσκησης ζητά να προσθέσω δύο longwords και συγκεκριμένα δύο αριθμούς των 16 bytes ο καθένας(128 bits). Δυστυχώς η assembly δεν διαθέτει εντολές αποθήκευσης είτε σε καταχωρητές είτε σε μνήμη απευθείας τέτοιας ποσότητας πληροφορίας οπότε ακολουθούμε συγκεκριμένη μεθοδολογία προκειμένου να αποθηκευτεί το τελικό αποτέλεσμα στη μνήμη. Η μεθοδολογία είναι η εξής:

Θα φορτώσω αρχικά σε καταχωρητές τα 32 λιγότερο σημαντικά bits(word) του longword κάθε πίνακα ,θα τα προσθέσω μεταξύ τους ,θα αποθηκεύσω το αποτέλεσμα στη θέση μνήμης που σηματοδοτείται από την ετικέτα Pinakasgamma και θα αποθηκεύσω το κρατούμενο που προκύπτει από την πρόσθεση στον καταχωρητή CPSR. Έπειτα θα φορτώσω σε καταχωρητές τα αμέσως 32 πιο σημαντικά bits(words) του longword του κάθε πίνακα και θα προσθέσω αυτά και το κρατούμενο που είχε προκύψει από τη προηγούμενη πρόσθεση και το αποτέλεσμα το αποθηκεύω στη μνήμη καθώς και θα αποθηκεύσω το κραοτύμενο που προέκυψε στον CPSR. Η διαδικασία επαναλαμβάνεται μέχρι να εξαντλήσω όλο το longword. Ακολουθεί ο κώδικας:

```
.arm

.text

.global main

main:

STMDB R13!,{R0-R4} @ αποθηκεύω τους καταχωρητές που θα χρησιμοποιήσω στη σωρό του συστήματος.

MOV R3,#0 @ Αρχικοποιώ τον καταχωρητή/μετρητή R3 με 0.
```

LOOP: @Ετικέτα LOOP που τη χρησιμοποιώ για επαναλήψεις.

LDR R0,=PinakasA @Αποθηκεύω στον καταχωρητή R0 τη διεύθυνση της μνήμης που σηματοδοτεί η ετικέτα PinakasA. Εκεί θα αποθηκευτεί όλη η πληροφορία που περιέχει ο πίνακας Α.

LDR R1,=PinakasB @Αποθηκεύω στον καταχωρητή R1 τη διεύθυνση της μνήμης που σηματοδοτεί η ετικέτα PinakasB. Εκεί θα αποθηκευτεί όλη η πληροφορία που περιέχει ο πίνακας Β.

LDR R2,=PinakasGamma @Αποθηκεύω στον καταχωρητή R4 τη διεύθυνση της μνήμης που σηματοδοτεί η ετικέτα PinakasGamma. Εκεί θα αποθηκευτεί όλη η πληροφορία που περιέχει ο πίνακας Γ(τον αρχικοποιώ με 0) και μετά η πληροφορία(τα words που προκύπτουν από την πρόσθεση των words από τους Α και Β.

LDR R0,[R0,R3] @ Αποθηκεύω στον καταχωρητή R0 ένα word πληροφορίας από τη περιοχή στη μνήμη που δείχνει ο R0+<τιμή καταχωρητή R3> θέσεις.

LDR R1,[R1,R3] @ Αποθηκεύω στον καταχωρητή R1 ένα word πληροφορίας από τη περιοχή στη μνήμη που δείχνει ο R1+<τιμή καταχωρητή R3> θέσεις.

ADCS R4,R1,R0 Προσθέτω τα περιεχόμενα των καταχωρητών R1 και R0 και την τιμή που υπάρχει αποθηκευμένη στη σημαία κράτούμενου του καταχωρητή κατάστασης αποθηκεύουμε το αποτέλεσμα στον R4 και κρατάμε το κρατούμενο εξόδου στη σημαία κρατούμενου του καταχωρητή κατάστασης.

STR R4,[R2,R3] @Αποθηκεύω ένα word πληροφορίας του περιεχομένου του καταχωρητή R4 και το αποθηκεύω στη περιοχή της μνήμης που δείχνει ο καταχωρητής R2+<τιμή καταχωρητή R3> θέσεις μπροστά(Πίνακας Γ).

ADD R3,R3,#4 @Αυξάνω την τιμή του καταχωρητή/μετρητή R3 κατά 4. Κατά 4 γιατί σε κάθε προσπέλαση του πίνακα Α όπως και Β επεξεργάζομαι ένα word πληροφορίας δηλαδή 4 bytes. Έστω για παράδειγμα ότι προσπελαύνω τη περιοχή στη μνήμη που δείχνει ο καταχωρητής R0. Επειδή το πρώτο word που θα αποθηκεύσω σε καταχωρητή αποτελείται από 4 bytes , τα 4 bytes του θα καταλαμβάνουν διαδοχικά τις διευθύνσεις <τιμή καταχωρητή R0+0>(το λιγότερο σημαντικό) , <τιμή καταχωρητή R0 +1>,<τιμή καταχωρητή R0+2>,<τιμή καταχωρητή R0+3>(το πιο σημαντικό), οπότε στην επόμενη προσπέλαση το word που θα επεξεργαστώ θα ξεκινά από τη διεύθυνση εκεί που δείχνει ο καταχωρητής R0 +4.

CMP R3,#16 @Αφού ο μετρητής σε κάθε επανάληψη αυξάνεται κατά 4 και συνολικά για να καλυφθεί όλο το longword(128 bits) επεξεργάζομαι 1 word(4 bytes ανά επανάληψη) η συνθήκη τερματισμού του βρόχου είναι R3=16(10 HEX).

BNE LOOP @ Από το branch if not equal αν η παραπάνω σύγκριση παράγει 0(ψευδής) εκτελεί άλμα στη διεύθυνση που σημειώνεται από την ετικέτα LOOP.

.data

PinakasA:

.byte 32,127,254,57,22,111,48,11,87,45,114,45,66,23,134,168

PinakasB:

.byte 19,1,18,89,90,112,89,32,23,98,67,83,146,140,200,67

PinakasGamma:

.byte 0