ΕΡΓΑΣΤΗΡΙΟ ΒΑΣΙΚΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Δούρου Βασιλική Ευαγγελία- Α.Μ.:1072633

Πεσκελίδης Παύλος- Α.Μ.:1072483

**ΕΡΓΑΣΙΑ 2:**

Σε όλα τα ερωτήματα της εργασίας, θεωρήθηκε πως τα στοιχεία όλων των πινάκων είναι δοσμένα στο δεκαδικό σύστημα, καθώς και ότι τα αποτελέσματα ζητούνταν να αναπαρασταθούν ως δεκαδικοί αριθμοί.

1. Ο κώδικας του ερωτήματος είναι ο ακόλουθος:

.arm

.text

.global main

main:

STMDB R13!, {R0-R12}

LDR R0, =Values

LDR R6, =Values +16

LDRB R1, [R0, #0]

LDRB R2, [R6, #0]

ADD R3, R0, #32

LDR R5, =Result

Loop:

ADD R4, R2, R1

STRB R4, [R5]

ADD R5, R5, #1

LDRB R1, [R0, #1]!

LDRB R2, [R6, #1]!

CMP R6, R3

BLO Loop

LDMIA R13!, {R0-R12}

MOV PC, R14

.data

Values:

.byte 0x20, 0x7F, 0xFE, 0x39, 0x16, 0x6F, 0x30, 0x0B, 0x57, 0x2D, 0x72, 0x2D, 0x42, 0x17, 0x86, 0xA8

.byte 0x13, 0x01, 0x12, 0x59, 0x5A, 0x70, 0x59, 0x20, 0x17, 0x62, 0x43, 0x53, 0x92, 0x8C, 0xC8, 0x43

Result:

.byte 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Αρχικά, αποθηκεύουμε τους καταχωρητές που θα χρησιμοποιήσουμε. Έπειτα, εισάγουμε στον καταχωρητή R0 τη διεύθυνση που σηματοδοτεί η ετικέτα Values, ενώ στον καταχωρητή R6 τη διεύθυνση που έχει η θέση Values+16, ώστε με τον καταχωρητή R0 να διατρέχουμε τα δεδομένα του πίνακα Α και με τον R6 τα δεδομένα του πίνακα Β. Μετά, εισάγουμε στον R1 το περιεχόμενο της μνήμης που δείχνει ο R0, ενώ στον R2 το περιεχόμενο της θέσης μνήμης που δείχνει ο R6. Επιπλέον, χρησιμοποιούμε τον καταχωρητή R3 για να αποθηκεύσουμε την διεύθυνση της τελευταίας θέσης μνήμης και των δύο πινάκων, ώστε να σταματήσει η δομή επανάληψης, που αναφέρεται στη συνέχεια, όταν το R3 ισούται με το R6. Επίσης, χρησιμοποιούμε και τον καταχωρητή R5 για να αποθηκεύσουμε τα αποτελέσματα σε έναν τρίτο πίνακα στη μνήμη και τον R4 για να αποθηκεύουμε τα επιμέρους αθροίσματα για κάθε τιμή των R1, R2. Μέσα στη δομή επανάληψης που ξεκινάει με την ετικέτα loop, προσθέτουμε τους R1, R2 και αποθηκεύουμε τον αποτέλεσμα στον R4, μεταφέρουμε το περιεχόμενο του R4 στη μνήμη με τον R5 και μετά αυξάνουμε τη διεύθυνση των R0, R6 και R5 κατά ένα, ώστε να δείχνουν στην αμέσως επόμενη θέση μνήμης και αποθηκεύουμε τα περιεχόμενα των θέσεων που δείχνουν οι δύο πρώτοι στους R1, R2 αντίστοιχα. Η κατάληξη B στις εντολές STRB, LDRB συμβολίζει πως θα μεταφέρεται ποσότητα ενός byte κάθε φορά. Τέλος, όταν τελειώσει η εντολή επανάληψης, επαναφέρουμε τις αρχικές τιμές στους καταχωρητές και μεταφέρουμε τη ροή της εκτέλεσης στο σημείο που κλήθηκε η main. Τα δεδομένα στη μνήμη τα δηλώνουμε ως byte με την ντιρεκτίβα .byte.

Ο πίνακας συμπληρώνεται όπως ακολουθεί:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Byte | Πίνακας  Α | Πίνακας  Β | Δεκαεξαδικό | Δεκαδικό | Μη  αναμενόμενο |
| 0 | 32 | 19 | 0x33 | 51 |  |
| 1 | 127 | 1 | 0x80 | 128 |  |
| 2 | 254 | 18 | 0x110 | 272 | X |
| 3 | 57 | 89 | 0x92 | 146 |  |
| 4 | 22 | 90 | 0x70 | 112 |  |
| 5 | 111 | 112 | 0xDF | 223 |  |
| 6 | 48 | 89 | 0x89 | 137 |  |
| 7 | 11 | 32 | 0x2B | 43 |  |
| 8 | 87 | 23 | 0x6E | 110 |  |
| 9 | 45 | 98 | 0x8F | 143 |  |
| 10 | 114 | 67 | 0xB5 | 181 |  |
| 11 | 45 | 83 | 0x80 | 128 |  |
| 12 | 66 | 146 | 0xD4 | 212 |  |
| 13 | 23 | 140 | 0xA3 | 163 |  |
| 14 | 134 | 200 | 0x14E | 334 | X |
| 15 | 168 | 67 | 0xEB | 235 |  |

Τα μη αναμενόμενα αποτελέσματα οφείλονται στο γεγονός ότι το αποτέλεσμα της πρόσθεσης στις συγκεκριμένες περιπτώσεις χρειάζεται παραπάνω ψηφία για την αναπαράσταση του και δεν του αρκεί το ένα byte που του προσφέρεται σε αυτό το πρώτο ερώτημα. Έτσι, σε εκείνες τις περιπτώσεις στη μνήμη αποθηκεύεται το πρώτο byte του αποτελέσματος, δηλαδή το λιγότερο σημαντικό.

1. Ο κώδικας του ερωτήματος είναι ο ακόλουθος:

.arm

.text

.global main

main:

STMDB R13!, {R0-R12}

LDR R0, =Values

LDR R6, =Values +16

LDRH R1, [R0, #0]

LDRH R2, [R6, #0]

ADD R3, R0, #32

LDR R5, =Result

Loop:

ADD R4, R2, R1

STRH R4, [R5]

ADD R5, R5, #2

LDRH R1, [R0, #2]!

LDRH R2, [R6, #2]!

CMP R6, R3

BLO Loop

LDMIA R13!, {R0-R12}

MOV PC, R14

.data

Values:

.hword 0x7F20, 0x39FE, 0x6F16, 0x0B30, 0x2D57, 0x2D72, 0x1742, 0xA886

.hword 0x0113, 0x5912, 0x705A, 0x2059, 0x6217, 0x5343, 0x8C92, 0x43C8

Result:

.hword 0,0,0,0,0,0,0,0

Αρχικά, και σε αυτή την περίπτωση, αποθηκεύουμε τους καταχωρητές που θα χρησιμοποιήσουμε. Και εδώ, εισάγουμε στον καταχωρητή R0 τη διεύθυνση που σηματοδοτεί η ετικέτα Values, ενώ στον καταχωρητή R6 τη διεύθυνση που έχει η θέση Values+16, ώστε με τον καταχωρητή R0 να διατρέχουμε τα δεδομένα του πίνακα Α και με τον R6 τα δεδομένα του πίνακα Β. Έπειτα, εισάγουμε στον καταχωρητή R1, το περιεχόμενο της θέσης μνήμης που δείχνει ο R0, ενώ στον R2 το περιεχόμενο της θέσης μνήμης που δείχνει ο R6. Επιπλέον, χρησιμοποιούμε τον καταχωρητή R3 για να αποθηκεύσουμε την διεύθυνση της τελευταίας θέσης μνήμης και των δύο πινάκων, ώστε να σταματήσει η δομή επανάληψης, που αναφέρεται στη συνέχεια, όταν το R3 ισούται με το R6. Επίσης, χρησιμοποιούμε τον καταχωρητή R5 για να μεταφέρουμε στη μνήμη τα αποτελέσματα των πράξεων και τον αρχικοποιούμε με την διεύθυνση που σηματοδοτεί η ετικέτα Result, και τον R4 για να αποθηκεύουμε τα επιμέρους αθροίσματα των R1, R2 για κάθε τιμή τους. Μετά, αρχίζει η δομή επανάληψης, όπου προσθέτουμε τους R1, R2 και αποθηκεύουμε το αποτέλεσμα τους στον R4, μεταφέρουμε το περιεχόμενο του R4 στη μνήμη με τον καταχωρητή R5, και μετά αυξάνουμε τη διεύθυνση των R0, R6, R5 κατά δύο, αφού χρησιμοποιούμε halfwords, ώστε να δείχνουν στα αμέσως επόμενα δύο byte της μνήμης και αποθηκεύουμε στους R1, R2 τα περιεχόμενα των θέσεων μνήμης που δείχνουν οι δύο πρώτοι. Η κατάληξη H στις εντολές STRH, LDRH συμβολίζει πως θα μεταφέρεται ποσότητα ενός halfword κάθε φορά. Τέλος, όταν τελειώσει η εντολή επανάληψης, επαναφέρουμε τις αρχικές τιμές στους καταχωρητές και μεταφέρουμε τη ροή της εκτέλεσης στο σημείο που κλήθηκε η main. Τα δεδομένα στη μνήμη τα δηλώνουμε ως halfword με την ντιρεκτίβα .hword.

Ο πίνακας συμπληρώνεται (στο δεκαδικό σύστημα) όπως ακολούθως:

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

1. Ο κώδικας του ερωτήματος είναι ο ακόλουθος:

.arm

.text

.global main

main:

STMDB R13!, {R0-R12}

LDR R0, =Values

LDR R6, =Values +16

LDR R1, [R0, #0]

LDR R2, [R6, #0]

ADD R3, R0, #32

LDR R5, =Result

Loop:

ADD R4, R2, R1

STR R4, [R5]

ADD R5, R5, #4

LDR R1, [R0, #4]!

LDR R2, [R6, #4]!

CMP R6, R3

BLO Loop

LDMIA R13!, {R0-R12}

MOV PC, R14

.data

Values:

.word 0x39FE7F20, 0x0B306F16, 0x2D722D57, 0xA8861742

.word 0x59120113, 0x2059705A, 0x53436217, 0x43C88C92

Result:

.word 0,0,0,0

Ο κώδικας ακολουθεί την ίδια λογική με τα προηγούμενα δύο ερωτήματα, όμως, αυτή τη φορά επειδή έχουμε να προσθέσουμε words, χρησιμοποιούμε τις εντολές LDR και STR, χωρίς κάποια κατάληξη (όπως είχαμε πριν το B και το H), ενώ όταν αυξάνουμε τη διεύθυνση των R0, R5, R6 για να δείχνουν στην επόμενη θέση μνήμης προς αποθήκευση ή ανάγνωση, η αύξηση γίνεται κατά 4, αφού ένα word ισοδυναμεί με 4 bytes. Τα δεδομένα στη μνήμη τα δηλώνουμε ως word με την ντιρεκτίβα .word.

Ο πίνακας συμπληρώνεται (στο δεκαδικό σύστημα) όπως φαίνεται παρακάτω:

|  |  |  |  |
| --- | --- | --- | --- |
| Byte | Πίνακας Α | Πίνακας Β | Πίνακας Γ |
| 0  1  2  3 | 32  127  254  57 | 19  1  18  89 | 2467332147 |
| 4  5  6  7 | 22  111  48  11 | 90  112  89  32 | 730455920 |
| 8  9  10  11 | 87  45  114  45 | 23  98  67  83 | 2159382382 |
| 12  13  14  15 | 66  23  134  168 | 146  140  200  67 | 3964576724 |

1. Ο κώδικας είναι ο ακόλουθος:

.arm

.text

.global main

main:

STMDB R13!, {R0-R12}

LDR R0, =Values

LDR R6, =Values +16

LDR R1, [R0, #0]

LDR R2, [R6, #0]

ADD R3, R0, #32

LDR R5, =Result

ADDS R4, R2, R1

STR R4, [R5]

Loop:

ADDS R4, R2, R1

ADD R5, R5, #4

LDR R1, [R0, #4]!

LDR R2, [R6, #4]!

ADC R4, R2, R1

STR R4, [R5]

CMP R6, R3

BLO Loop

LDMIA R13!, {R0-R12}

MOV PC, R14

.data

Values:

.word 0x39FE7F20, 0x0B306F16, 0x2D722D57, 0xA8861742

.word 0x59120113, 0x2059705A, 0x53436217, 0x43C88C92

Result:

.word 0,0,0,0

Και σε αυτή την περίπτωση, ο κώδικας παρουσιάζει αρκετές ομοιότητες με τους άλλους τρεις. Όμως, εδώ πριν την έναρξη της επαναληπτικής δομής γίνεται η πρόσθεση των 4 λιγότερων σημαντικών byte του κάθε πίνακα (δηλαδή των λιγότερων σημαντικών word), κρατιέται το κρατούμενο εξόδου στη σημαία κρατουμένου του καταχωρητή κατάστασης και αποθηκεύεται το αποτέλεσμα στον καταχωρητή R4 και μετά μεταφέρεται με τον R5 στα 4 λιγότερα σημαντικά bytes του αποτελέσματος στη θέση μνήμης. Έπειτα, ξεκινάει η επαναληπτική δομή, όπου έχουμε βάλει να ξαναγίνεται αυτή η διαδικασία, χωρίς τη μεταφορά στη μνήμη όμως, για να μπορούμε να ελέγχουμε για κρατούμενο ανάμεσα και στις υπόλοιπες δυάδες words. Μετά, αυξάνεται η διεύθυνση των R0, R6, R5 όπως και στους προηγούμενους κώδικες, και γίνεται η πρόσθεση ανάμεσα στο R1, στο R2, αλλά και στη σημαία κρατουμένου του καταχωρητή κατάστασης και το αποτέλεσμα αποθηκεύεται στον R4. Το loop συνεχίζει έως ότου R3=R6 και οι υπόλοιπες διαδικασίες γίνονται όπως και στους προηγούμενους κώδικες. Επίσης, στη μνήμη τα δεδομένα και των Values και του αποτελέσματος αποθηκεύονται ως words, αφού δεν υποστηρίζονται ποσότητες μεγαλύτερες των 32 bit.

Ο πίνακας συμπληρώνεται (στο δεκαδικό σύστημα) όπως φαίνεται στη συνέχεια:

|  |  |  |  |
| --- | --- | --- | --- |
| Bytes | Πίνακας Α | Πίνακας Β | Πίνακας Γ |
| 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | 32  127  254  57  22  111  48  11  87  45  114  45  66  23  134  168 | 19  1  18  89  90  112  89  32  23  98  67  83  146  140  200  67 | 314106129029175284967811117607864533043 |

Αν εκτελούσαμε προσθέσεις bytes και όχι words, τότε θα χρειάζονταν να προσθέσουμε κάθε γραμμή του πίνακα Α με κάθε αντίστοιχη γραμμή του πίνακα Β και έτσι θα γινόντουσαν 16 προσθέσεις.