8

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 3 – ΟΜΑΔΑ Α5

Μηλτιάδης Μαντές AM 1084661 EMAIL up1084661@upnet.gr

Ελπίδα Κόκκαλη AM 1084648 EMAIL up1084648@upnet.gr

i. Μελέτη καταχωρητή κατάστασης

```
1 .arm
2 .text
3 .global main
4
5 main:
6 STMDB R13!, {R0-R12, R14}
7
8 MOV R0, #94 @Μεταφέρουμε στον R0 τη τιμή 94
9 MOV R1, R0, LSR #1 @Μεταφέρουμε το περιεχόμενο του R0 στον R1 ολισθημένο κατά μια θέση δεξιά, δηλαδή στην ουσία διαιρούμε το 94 με το 2 και το αποθηκεύουμε στον R1
10
11 ADDS R2, R0, R0 @Στην ουσία διπλασιάζουμε τη τιμή του R0 και την αποθηκεύουμε στον R2 ενημερώνοντας παράλληλα και τις σημαίες του καταχωρητή κατάστασης (N, C, Z, V)
12 ADDS R2, R1, R1 @Στην ουσία διπλασιάζουμε τη τιμή του R1 και την αποθηκεύουμε στον R2 ενημερώνοντας παράλληλα και τις σημαίες του καταχωρητή κατάστασης (N, C, Z, V)
13 ADDS R2, R0, R1 @Προσθέτουμε τις τιμές των R0, R1 και το αποτέλεσμα το αποθηκεύουμε στον R2 ενημερώνοντας παράλληλα και τις σημαίες του καταχωρητή κατάστασης (N, C, Z, V)
14
15 MOV R0, #0x80000000 @Μεταφέρουμε στον R0 τη τιμή 80000000 σε HEX
16 ADD R1, R0, #0x80 @Προσθέτουμε στην αρχική τιμή του R0 το 80 σε HEX και το αποτέλεσμα το αποθηκεύουμε στον R1
17 MOV R2, #1 @Μεταφέρουμε στον R2 τη τιμή 1
18
19 SUBS R3, R0, R2 @Αφαιρούμε τη τιμή του R2 από τη τιμή του R0 και το αποτέλεσμα το αποθηκεύουμε στον R3 ενημερώνοντας παράλληλα και τις σημαίες του καταχωρητή κατάστασης (N, C, Z, V)
```

20 SUBS R3, R0, R1 **@Αφαιρούμε τη τιμή του R1 από τη τιμή του R0 και το αποτέλεσμα το αποθηκεύουμε στον R3 ενημερώνοντας παράλληλα και τις σημαίες του καταχωρητή κατάστασης (N, C, Z, V)**

21 RSBS R3, R0, R1 **@Αφαιρούμε τη τιμή του R0 από τη τιμή του R1 και το αποτέλεσμα το αποθηκεύουμε στον R3 ενημερώνοντας παράλληλα και τις σημαίες του καταχωρητή κατάστασης (N, C, Z, V)**

22

23 LDMIA R13!, {R0-R12, PC}

	N	C	Z	V	Σχόλια
@11	0	0	0	0	**Όλες οι σημαίες απενεργοποιημένες.**
@12	0	0	0	0	**Όλες οι σημαίες απενεργοποιημένες.**
@13	0	0	0	0	**Όλες οι σημαίες απενεργοποιημένες.**
@19	0	1	0	1	**Προκύπτει κρατούμενο (C) κατά την αφαίρεση (80000000 – 00000001) και έτσι έχουμε υπερχείλιση (V).**
@20	1	0	0	0	**Από την αφαίρεση (80000000 –80000080) προκύπτει αρνητικός αριθμός (N).**
@21	**0**	**1**	**0**	**0**	**Προκύπτει κρατούμενο (C) κατά τηναφαίρεση (80000080 – 80000000).**

	R2 (HEX)	R2 (DEC)	R3 (HEX)	R3 (DEC)
@11	0xBC	188	0x0	0
@12	0x5E	94	0x0	0
@13	0x8D	141	0x0	0
@19	0x01	1	0x7FFFFFFF	
@20	0x01	1	0xFFFFFF80	
@21	0x01	1	0x80	128

Το φαινόμενο αποκοπής δυαδικών ψηφίων εμφανίζεται μόνο κατά την εκτέλεση της εντολής **@19** λόγω του εύρους 32 bits των καταχωρητών καθώς:

R0 – R2 = 1000 0000 0000 0000 0000 0000 0000 0000

1111 1111 1111 1111 1111 1111 1111 1111 +

1 0111 1111 1111 1111 1111 1111 1111 1111

32 bits

ii. Προσπέλαση διαδοχικών θέσεων μνήμης

```
.arm

.text

.global main


main:

STMDB R13!, {R0-R12, R14}

LDR R1, = Stor @Ο R1 δείχνει στις 6 θέσεις μνήμης του πίνακα Stor

MOV R0, #0 @Μεταφέρουμε στον R0 τη τιμή 0 και θα τον χρησιμοποιήσουμε ως μετρητή στο βρόχο επανάληψης

STR R0, [R1], #4 @Αποθηκεύουμε τη τιμή 0 του R0 στον R1 και θέλουμε μετά να δείχνουμε στο επόμενο στοιχείο του πίνακα Stor το οποίο απέχει 4 bytes από το προηγούμενο μέσα στη μνήμη


Startpoint:

ADD R0, R0, #1 @Αυξάνουμε την τιμή του R0 - μετρητή κατά 1

STR R0, [R1], #4 @Αποθηκεύουμε κάθε φορά τη τιμή του R0 στον R1 και θέλουμε μετά να δείχνουμε στο επόμενο στοιχείο του πίνακα Stor το οποίο απέχει 4 bytes από το προηγούμενο μέσα στη μνήμη

CMP R0, #5 @Όσο το περιεχόμενο του R0 δεν είναι 5 ο βρόχος επαναλαμβάνεται προκειμένου να αποθηκεύσουμε στις υπόλοιπες 5 θέσεις μνήμης του πίνακα Stor τους αριθμούς από 1 ως 6

BNE Startpoint


LDMIA R13!, {R0-R12, PC}


.data
Stor:
.word 0,0,0,0,0,0
```

iii. Υπολογισμός αριθμών Fibonacci

```
.arm
.text
.global main


main:
STMDB R13!, {R0-R12,R14}
MOV R0, #0 @Μεταφέρουμε στον R0 τη τιμή 0 και θα τον χρησιμοποιήσουμε ως μετρητή στο βρόχο επανάληψης
MOV R1, #1 @Μεταφέρουμε στον R1 τη τιμή 1 η οποία είναι ο πρώτος όρος α0 της ακολουθίας Fibonacci
MOV R2, #2 @Μεταφέρουμε στον R2 τη τιμή 2 η οποία είναι ο δεύτερος όρος α1 της ακολουθίας Fibonacci

Startpoint:

LDR R3, = Stor @Ο R3 δείχνει στις 6 θέσεις μνήμης του πίνακα Stor
LDR R4, = Stor @Ο R4 δείχνει επίσης στις 6 θέσεις μνήμης του πίνακα Stor
LDR R5, = Stor @Ο R5 δείχνει επίσης στις 6 θέσεις μνήμης του πίνακα Stor
LDR R3, [R3,R0] @Μεταφέρει το περιεχόμενο της θέσης μνήμης με διεύθυνση το άθροισμα των διευθύνσεων των R3, R0 στον καταχωρητή R3 δηλαδή στην ουσία τον αn-2 όρο της ακολουθίας
LDR R4, [R4,R1] @Μεταφέρει το περιεχόμενο της θέσης μνήμης με διεύθυνση το άθροισμα των διευθύνσεων των R4, R1 στον καταχωρητή R4 δηλαδή στην ουσία τον αn-1 όρο της ακολουθίας
ADD R4 , R4 , R3 @Προσθέτουμε τις τιμές των R3, R4 (αn-2 και αn-1) και το αποτέλεσμα το αποθηκεύουμε πάλι στον R4 δηλαδή στην ουσία τον όρο αn της ακολουθίας
STR R4, [R5,R2]
ADD R0, R0, #1 @Αυξάνουμε την τιμή του R0 - μετρητή κατά 1
ADD R1, R1, #1 @Αυξάνουμε την τιμή του R1 κατά 1
ADD R2, R2, #1 @Αυξάνουμε την τιμή του R2 κατά 1
CMP R0,#4 @Όσο το περιεχόμενο του R0 δεν είναι 4 ο βρόχος επαναλαμβάνεται προκειμένου να αποθηκεύσουμε και στις υπόλοιπες 4 θέσεις μνήμης του πίνακα Stor τους όρους α3 ως α6 της ακολουθίας Fibonacci καθώς οι πρώτοι δύο όροι είναι γνωστοί και τους έχουμε αποθηκεύσει από πριν στις πρώτες δύο θέσεις του πίνακα

BNE LOOP

LDMIA R13!, {R0-R12, PC}

.data
Stor:
.word 0,0,0,0,0,0
```