### i. Μελέτη καταχωρητή κατάστασης

| | **N** | **C** | **Z** | **V** | **Σχόλια** |
|---|---|---|---|---|---|
| **@11** | 0 | 0 | 0 | 0 | Στον R2 αποθηκεύεται το άθροισμα 5E + 5E (= BC).<br>Το αποτέλεσμα της πράξης δεν είναι αρνητικό (N=0), ούτε μηδενικό (C=0), και δεν δημιουργεί κρατούμενο ή υπερχείλιση (C=0, V=0) |
| **@12** | 0 | 0 | 0 | 0 | Στον R2 αποθηκεύεται το άθροισμα 2F + 2F (= 5E).<br>Το αποτέλεσμα της πράξης δεν είναι αρνητικό (N=0), ούτε μηδενικό (C=0), και δεν δημιουργεί κρατούμενο ή υπερχείλιση (C=0, V=0) |
| **@13** | 0 | 0 | 0 | 0 | Στον R2 αποθηκεύεται το άθροισμα 5E + 2F (= 8D).<br>Το αποτέλεσμα της πράξης δεν είναι αρνητικό (N=0), ούτε μηδενικό (C=0), και δεν δημιουργεί κρατούμενο ή υπερχείλιση (C=0, V=0) |
| **@19** | 0 | 1 | 0 | 1 | Στον R3 αποθηκεύεται η διαφορά 80000000 - 1 (= 7FFFFFFF).<br>Εμφανίζονται τα flags C=1 και V=1 γιατί η πράξη δεν επιστρέφει κρατούμενο και επειδή στο αποτέλεσμα δημιουργείται ένα επιπλέον bit, το οποίο όμως δεν μπορεί να αποθηκευτεί στα 32 bits του καταχωρητή (υπερχείλιση).<br><br>1000 0000 0000 ... 0000 0000<br>1111 1111 1111 ... 1111 1111 +<br>**1** 0111 1111 1111 ... 1111 1111 |
| **@20** | 1 | 0 | 0 | 0 | Στον R3 αποθηκεύεται η διαφορά 80000000 - 80000080 (= FFFFFF80). Είναι C=0, γιατί επιστρέφει κρατούμενο: η μετατροπή σε 2's complement του 8000080 δημιουργεί ένα έξτρα bit, επομένως χρειάζεται 33 bits (αντί για 32) για να αναπαρασταθεί σωστά, συνεπώς θα έχει κρατούμενο.<br><br>Επίσης το αποτέλεσμα είναι αρνητικό, άρα N=1.<br><br>1000 0000 0000 ... 0000 0000<br>**1** 0111 1111 1111 ... 1000 0000 +<br>1111 1111 1111 ... 1000 0000 |
| **@21** | 0 | 1 | 0 | 0 | Στον R3 αποθηκεύεται η διαφορά 80000000 - 80000080 (= 80).<br>Η πράξη δεν επιστρέφει κρατούμενο άρα C=1, δεν έχουμε υπερχείλιση άρα V=0, και το αποτέλεσμα δεν είναι ούτε αρνητικός, ούτε μηδέν. |

## ii. Προσπέλαση διαδοχικών θέσεων μνήμης

```
0   .arm
1   .text
2   .global main
3
4   main:
5   STMDB R13!, {R0-R12,R14}
6
7   LDR R1, =Store
8   MOV R0, #0
9
10  loop:
11  STRB R0, [R1, R0]
12
13  ADD R0, R0, #1
14
15  CMP R0, #0x6
16  BNE loop
17
18  LDMIA R13!, {R0-R12,R14}
19
20  .data
21
22  Stor:
23  .byte 0x00, 0x00, 0x00, 0x00
24  .byte 0x00, 0x00
25
```

Στον καταχωρητή R1 αποθηκεύεται η διεύθυνση της θέσης μνήμης στην οποία θα αποθηκευτούν οι αριθμοί 0 ... 5. Ο καταχωρητής R0 θα είναι ο μετρητής, όπου το αυξανόμενο περιεχόμενό του θα αποθηκεύεται κάθε φορά στη μνήμη.

Στην 11η γραμμή θα αποθηκεύεται κάθε φορά το περιεχόμενο του καταχωρητή R0, πρακτικά οι αριθμοί 0 μέχρι 5, στην θέση μνήμης που κρατάει ο R1. Η θέση μνήμης κάθε φορά αυξάνεται κατά ένα, διότι της προσθέτουμε το περιεχόμενο του R0.

Έτσι τη δεύτερη φορά που θα τρέξει η επανάληψη, το περιεχόμενο R0 + 1 = 1 θα αποθηκευτεί στην θέση μνήμης R1 + 1, την τρίτη το περιεχόμενο R0 + 1 = 2 στην R1+2, επιτυγχάνοντας έτσι διαδοχική τοποθέτηση byte στην μνήμη.

Τέλος συγκρίνουμε το R0 με το 6 και όχι με το 5, διότι θα αυξηθεί κατά ένα επιπλέον κατά την πέμπτη και τελευταία επανάληψη.

## iii. Υπολογισμός αριθμών Fibonacci

```
0   . arm
1   .text
2   .global main
3
4   main:
5   STMDB R13!, {R0-R12,R14}
6
7   LDR R4, =Store          @Θέση μνήμης
8   MOV R3, #0              @Μετρητής
9   MOV R0, #0              @a(n-2)
10  MOV R1, #1              @a(n-1)
11
12  loop:
13  ADD R2, R0, R1          @an
14  STRB R2, [R4, R3]
15
16  MOV R0, R1
17  MOV R1, R2
18
19  ADD R3, R3, #1
20
21  CMP R3, #0x6
22  BNE loop
23
24  LDMIA R13!, {R0-R12,R14}
25
26  .data
27
28  Stor:
29  .byte 0x00, 0x00, 0x00, 0x00
30  .byte 0x00, 0x00
```

Θέλουμε να αποθηκεύσουμε τους αριθμούς 1, 2, 3, 5, 7, 13 στις θέσεις μνήμης [Stor] ... [Store+5]:

Ο καταχωρητής R4 αποθηκεύει την θέση μνήμης Stor απ' όπου θα ξεκινήσουν να αποθηκεύονται οι αριθμοί. Ο R3 μετράει τις 6 επαναλήψεις που πρέπει να γίνουν και χρησιμεύει και ως δείκτης στη μνήμη. Θέλουμε $a_0 = 1$, άρα στους R0 και R1 είναι αποθηκευμένοι οι αριθμοί 0 και 1.

Στην επανάληψη προστίθενται οι καταχωρητές R0 & R1 ($a_{n-1} + a_{n-2}$) στον καταχωρητή R2 ($a_n$) του οποίου το περιεχόμενο αποθηκεύεται σε θέσεις μνήμης που καθορίζονται από τον R4 (που κρατάει την αρχική διεύθυνση Stor) και τον R3 που την αυξάνει διαδοχικά.

Έπειτα το περιεχόμενο του R1 ($a_{n-2}$) μεταφέρεται στον R0 ($a_{n-1}$) και το περιεχόμενο του R2 ($a_n$) στον R1 ($a_{n-1}$) και αυξάνεται κατά ένα ο μετρητής.

Ο μετρητής R3 συγκρίνεται με το 6 ώστε να γίνουν 6 επαναλήψεις.