3Η ΕΡΓΑΣΙΑ ΕΡΓΑΣΤΗΡΙΟΥ ΒΑΣΙΚΩΝ ΘΕΜΑΤΩΝ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Συγγραφείς:

Βασίλειος Τσούλος, ΑΜ: 1072605

Γεώργιος Βέργος ΑΜ :1072604

1Ο ΕΡΩΤΗΜΑ

	N	C	Z	V	Σχόλια
@11	0	0	0	0	1)
@12	0	0	0	0	2)
@13	0	0	0	0	3)
@19	0	1	0	1	4)
@20	1	0	0	0	5)
@21	0	1	0	0	6)

1)

ADDS R2, R0, R0

Αρχικά το πρόγραμμα μεταφέρει την τιμή +94 (0x0000005E) στον καταχωρητή R0.

Παίρνει το περιεχόμενο του R0 του κάνει λογική ολίσθηση δεξιά κατά μία θέση και αποθηκεύει το αποτέλεσμα στον R1.

Άρα πλέον η τιμή του R1 θα είναι το δεκαεξαδικό 0x0000002F (δεκαδικό +47).

Η εντολή **ADDS R2, R0, R0**

προσθέτει το περιεχόμενο του R0 με τον εαυτό του και αποθηκεύει το αποτέλεσμα στον R2 (0x0000005E + 0x0000005E = BC[94+94=+188]) => R2 = 0x000000BC και κρατάμε το

κρατούμενο εξόδου στη σημαία κρατούμενου του καταχωρητή κατάστασης.

Το N = 0(το bit N δηλώνει αν το αποτέλεσμα της πράξης είναι αρνητικό) όπως και το Z = 0(το bit που δηλώνει αν το αποτέλεσμα της πράξης είναι μηδέν) γιατί το αποτέλεσμα της πράξης είναι θετικό (0x000000BC > 0 αφού 0x000000BC =+ 188>0 διάφορο του μηδέν)

Το C = 0(Το bit του CPSR που δείχνει αν κατά τη πρόσθεση υπήρξε κρατούμενο εξόδου) και το V = 0(Το bit του CPSR που δείχνει αν το αποτέλεσμα της πράξης δημιούργησε υπερχείλιση) γιατί αρχικά κατά την εκτέλεση της πρόσθεσης των 5E + 5E στο δυαδικό:

(24 μηδενικά)01011110

\+ (24 μηδενικά)01011110

(24 μηδενικά)10111100=0x000000BC

Δεν προκύπτει ούτε κρατούμενο εξόδου αλλά ούτε και το αποτέλεσμα έχει περισσότερα δυαδικά ψηφία από όσα μπορούν να αποθηκευτούν οπότε δε προκύπτει υπερχείλιση.

2)

ADDS R2, R1, R1

Προσθέτει το περιεχόμενο του R1(0x0000002F) με τον εαυτό του και αποθηκεύει το αποτέλεσμα στον καταχωρητή R2 και ενημερώνει τη σημαία κρατουμένου του καταχωρητή κατάστασης (CPSR) .

Άρα ,

R2 = 0x0000002F + 0x0000002F => R2 = 0x0000005E

Και αναλυτικά από τη πράξη στο δυαδικό :

(24 μηδενικά)00101111

+(24 μηδενικά)00101111

(24 μηδενικά)01011110=0x0000005E

Δεν προκύπτει ούτε κρατούμενο εξόδου αλλά ούτε και το αποτέλεσμα έχει περισσότερα δυαδικά ψηφία από όσα μπορούν να αποθηκευτούν οπότε δε προκύπτει υπερχείλιση.

Οπότε θα ισχύει C=V=0.

Το N = 0(το bit N δηλώνει αν το αποτέλεσμα της πράξης είναι αρνητικό) όπως και το Z = 0(το bit που δηλώνει αν το αποτέλεσμα της πράξης είναι μηδέν) γιατί το αποτέλεσμα της πράξης είναι θετικό (0x0000005E > 0 αφού 0x0000005E =+ 47>0 διάφορο του μηδέν).

3)

ADDS R2, R0, R1

Αθροίζει τις τιμές των περιεχομένων των καταχωρητών R0(0x0000005E) και R1(0x0000002F), αποθηκεύει το αποτέλεσμα της πρόσθεσης στο R2, και ενημερώνει τη σημαία κρατουμένου του καταχωρητή κατάστασης (CPSR) .

Και αναλυτικά από τη πράξη στο δυαδικό:

(24 μηδενικά)01011110

+ (24 μηδενικά) 00101111

(24 μηδενικά)10001101=0x0000008D

Παρατηρώ ότι δεν προκύπτει ούτε κρατούμενο εξόδου αλλά ούτε και το αποτέλεσμα έχει περισσότερα δυαδικά ψηφία από όσα

μπορούν να αποθηκευτούν στον R3 οπότε δε προκύπτει υπερχείλιση.

Οπότε θα ισχύει C=V=0.

Το N = 0(το bit N δηλώνει αν το αποτέλεσμα της πράξης είναι αρνητικό) όπως και το Z = 0(το bit που δηλώνει αν το αποτέλεσμα της πράξης είναι μηδέν) γιατί το αποτέλεσμα της πράξης είναι θετικό (0x0000008D > 0 αφού 0x0000008D =+ 141>0 διάφορο του μηδέν).

4)

Έπειτα μεταφέρουμε στον καταχωρητή R0 τη τιμή 0x80000000 ,έπειτα σε αυτή τη τιμή προσθέτουμε το 0x00000080 και αποθηκεύουμε το αποτέλεσμα στον R1 οπότε R1=0x80000080

Και τέλος μεταφέρουμε τη τιμή 0x00000001 στον καταχωρητή R2.

SUBS R3, R0, R2

Αφαιρεί το περιεχόμενο του R2 από το περιεχόμενο του R0 , αποθηκεύουμε το αποτέλεσμα της πράξης στον καταχωρητή R3 και ενημερώνει τον καταχωρητή κατάστασης (CPSR) για το είδος του αποτελέσματος.

Κάνει δηλαδή την αφαίρεση 0x8000000-0x00000001

- Αναλυτικά στο δυαδικό:

1000(24μηδενικά)0000

- 0000(24μηδενικά)0001

Επειδή στους σύγχρονους υπολογιστές οι αρνητικοί αριθμοί αναπαρίστανται με το συμπλήρωμα ως προς 2. Τότε θα έχω τη πρόσθεση:

1000(24 μηδενικά)0000

+1111(24 άσσοι)1111

10111(24 άσσοι)1111

Παρατηρώ ότι προέκυψαν 33 δυαδικά ψηφία(επιπλέον κόκκινος άσσος) κατά την εκτέλεση της παραπάνω πράξης τα οποία όμως δε μπορούν να αποθηκευτούν όλα(αυτός ο άσσος δεν θα αποθηκευτεί. Έγινε δηλαδή υπέρβαση της χωρητικότητας του καταχωρητή R3(32 bits).

Έχουμε δηλαδή υπερχείλιση οπότε και V=1 και ο R3 θα έχει την τιμή 0x7FFFFFFF>0 . Εφόσον το αποτέλεσμα είναι θετικό άρα ούτε αρνητικό ούτε μηδέν οι σημαίες N και Z του καταχωρητή κατάστασης θα είναι μηδέν(0). Εφόσον πρόκειται για πράξη αφαίρεσης περιεχομένων των καταχωρητών και αφού παρατηρούμε ότι δε προκύπτει δανεικό εξόδου η σημαία C του καταχωρητή κατάστασης θα πάρει τη τιμή 1. Μια άλλη ερμήνευση της τιμής της σημαίας C=1(πέρα του ότι δεν υπάρχει δανεικό εξόδου είναι ότι κατά την πρόσθεση του αντιθέτου του 1(+(-1)) προκύπτει κρατούμενο εξόδου(ο κόκκινος άσσος) οπότε και για αυτό C=1.

5)

SUBS R3, R0, R1

Αφαιρεί την τιμή του καταχωρητή R1 από την τιμή του καταχωρητή R0 και αποθηκεύει το αποτέλεσμα στον καταχωρητή R3 και ενημερώνει τον καταχωρητή κατάστασης (CPSR) για το είδος του αποτελέσματος.

Κάνει δηλαδή την αφαίρεση 0x80000000-0x80000080= -80(hex).

¶ Αναλυτικά στο δυαδικό:

1000(20 μηδενικά)00000000

-1000(20 μηδενικά)10000000

Επειδή στους σύγχρονους υπολογιστές οι αρνητικοί αριθμοί αναπαρίστανται με το συμπλήρωμα ως προς 2. Τότε θα έχω τη πρόσθεση:

1000(20 μηδενικά)00000000

\+ 0111(20 άσσοι)10000000

1111(20 άσσοι) 10000000=0xffffff80=-128<0

Οπότε επειδή το αποτέλεσμα(αυτό που αποθηκεύεται στον R3) είναι αρνητικός αριθμός: το πιο σημαντικό ψηφίο είναι άσσος και αφού χρησιμοποιούμε συμπλήρωμα ως προς 2 τότε είναι αρνητικός (άρα και διάφορος του μηδέν) θα έχω N=1(από το negative), Z=0. Από τη παραπάνω αφαίρεση είναι προφανές πως θα προκύψει δανεικό ψηφίο εξόδου οπότε και C=0(προέκυψε δανεικό).Μία άλλη προσέγγιση για το C=0 είναι ότι δε προέκυψε κρατούμενο εξόδου κατά την παραπάνω πρόσθεση. Επιπλέον δε προέκυψε υπερχείλιση δηλαδή περισσότερα ψηφία από όσα μπορούν να αποθηκευτούν στον καταχωρητή R3(32 bits), οπότε και V=0.

6)

RSBS R3, R0, R1

Από το Reverse substract.

R3 - > Είναι ο καταχωρητής που θα αποθηκευτεί το αποτέλεσμα

R1 -> Πρώτος όρος της αφαίρεσης .

R0 -> Περειέχει τον δεύτερο όρο της αφαίρεσης.

Συγκεκριμένα εκτελείται η πράξη:

R3 =R1-R0=0x80000080 –0x80000000 =0x00000080=+128>0

- Αναλυτικά στο δυαδικό:
- 1000(20 μηδενικά)10000000
  -1000(20 μηδενικά)00000000

  0000(20 μηδενικά)10000000=0x00000080=+128>0

Οπότε επειδή το αποτέλεσμα είναι θετικός αριθμός(αυτό που αποθηκεύεται στον R3 έχει ως πιο σημαντικό ψηφίο το μηδέν(0) στο συμπλήρωμα ως προς 2 που χρησιμοποιείται αυτό δηλώνει θετικό αριθμό (άρα και διάφορος του μηδέν) θα έχω N=0(από το negative), Z=0. Από τη παραπάνω αφαίρεση είναι προφανές πως ΔΕΝ θα προκύψει δανεικό ψηφίο εξόδου οπότε και C=1(δε προέκυψε δανεικό στην αφαίρεση). Επιπλέον δε προέκυψε υπερχείλιση δηλαδή περισσότερα ψηφία από όσα μπορούν να αποθηκευτούν στον καταχωρητή R3(32 bits), οπότε και V=0.

Πρόκειται για μία αντίστροφη αφαίρεση (σε αντίθεση με την SUBS δηλαδή που αφαιρούσε τον δεύτερο όρο απ’ τον πρώτο , η RSBS αφαιρεί τον πρώτο όρο απ’ το δεύτερο και ενημερώνει τον CPSR για το είδος του αποτελέσματος.).

2ο ΕΡΩΤΗΜΑ

Ο κώδικας που χρησιμοποιήθηκε γι’ αυτό το ερώτημα:

```
.arm

.text

.global main

main:

STMDB R13!, {R0-R12, R14}

MOV R3,#0

LOOP: @ Ετικέτα LOOP , χρησιμοποιείται για επαναλήψεις.

LDR R0,=Stor

STRB R3,[R0,R3]

ADD R3,R3,#1

TEQ R3, #6

BNE LOOP @Αν η παραπάνω σύγκριση δε βγάζει ισότητα εκτελείται άλμα στη διεύθυνση που σημειώνεται από την ετικέτα LOOP.

MOV PC,R14
```

.data

Stor:

.byte 0

Ανάλυση και επεξήγηση του παραπάνω κώδικα:

MOV R3,#0 -> Χρήση του καταχωρητή R3 ως μετρητή για την προσπέλαση του πίνακα Stor

Αρχικοποίηση αρχικά με το 0.

LDR R0,=Stor -> Αποθηκεύουμε στον καταχωρητή R0 την διεύθυνση στη μνήμη που σηματοδοτεί η ετικέτα Stor. Δηλαδή, ο R0 είναι ένας δείκτης στον πίνακα Stor.

STRB R3,[R0,R3] -> Θα αποθηκεύσουμε 1 byte του περειεχόμενου του R3 στην θέση μνήμης που δείχνει ο R0 + <περειεχόμενο του R3> θέσεις μπροστά.

ADD R3,R3,#1 -> Αύξηση του μετρητή R3 κατά 1 για να προσπελάσουμε στην επόμενη επανάληψη το αμέσως επόμενο στοιχείο του πίνακα Stor.

TEQ R3, #6 -> Συγκρίνουμε το περιεχόμενο του R3 με το 6, όπου αποτελεί συνθήκη τερματισμού του βρόχου γιατί ο πίνακας έχει 6 στοιχεία.(Ελέγχουμε ισότητα του περιεχομένου του R3 με το 6).

MOV PC,R14 -> Μεταφέρουμε την ροή του προγράμματος στο σημείο που κλήθηκε η υπορουτίνα LOOP.

Stor:

Πρόκειται για την ετικέτα Stor. Εφόσον βρίσκεται στη περιοχή data του προγράμματος μας (ντιρεκτίβα .data) σηματοδοτεί τη διεύθυνση στη μνήμη που θα αποθηκεύονται τα δεδομένα μας δηλαδή πρόκειται για τον πίνακα Stor.

3^{Ο} ΕΡΩΤΗΜΑ

Ο κώδικας που χρησιμοποιήθηκε για το ερώτημα αυτό

```
.arm
.text
.global main
main:
STMDB R13!,{R0-R4,R14}
MOV R0,#1
MOV R4,#2
MOV R1,#2
LDR R3,=Stor
STRB R0,[R3]
STRB R1,[R3,#1]
LOOP:
LDR R3,=Stor
ADD R2,R1,R0
STRB R2,[R3,R4]
MOV R0,R1
```

```
MOV R1,R2
ADD R4,R4,#1
TEQ R4,#6
BNE LOOP
MOV PC,R14
.data
Stor:
.byte 0
```

Ανάλυση και επεξήγηση του παραπάνω κώδικα:

STMDB R13!,{R0-R4,R14} -> Αποθηκεύουμε τους καταχωρητές που θα χρησιμοποιήσουμε στη σωρό του συστήματος.

MOV R0,#1 -> Μεταφέρουμε την τιμή 1 στον καταχωρητή R0. Ο R0 είναι ο καταχωρητής που θα παίρνεις τις τιμές του α_{n-2}.

MOV R4,#2 -> Μεταφέρουμε την τιμή 2 στον καταχωρητή R4. Ο R4 είναι ο μετρητής με τον οποίο θα προσπελάσουμε τον πίνακα Stor προκειμένου να αποθηκεύσουμε σ' αυτόν τους όρους Fibonacci. Τον αρχικοποιούμε με 2 και όχι με 0, γιατί οι πρώτοι 2 όροι είναι ορισμένοι από υπόθεση.

MOV R1,#2 -> Μεταφέρουμε στον καταχωρητή R1 την τιμή 2. Ο R1 είναι ο καταχωρητής που θα χρησιμοποιηθεί για να αποθηκεύουμε τις τιμές $\alpha_{v-1.}$

LDR R3,=Stor -> Μεταφέρουμε στον R3 την διεύθυνση μνήμης που σηματοδοτεί η ετικέτα Stor. Άρα ο R3 αποτελεί δείκτη στον πίνακα Stor.

STRB R0,[R3] -> Θα αποθηκεύσουμε το πρώτο όρο της ακολουθίας στην πρώτη θέση του πίνακα.

STRB R1,[R3,#1] -> Θα αποθηκεύσουμε το δεύτερο όρο της ακολουθίας στην δεύτερη θέση του πίνακα.

ADD R2,R1,R0 -> Προσθέτουμε τα περιεχόμενα των R1 (α_{n-1}) με τον R0(α_{n-2}) και αποθηκεύουμε το αποτέλεσμα στον καταχωρητή R2 (α_n). Υλοποιούμε δηλαδή τον τύπο $a_n=a_{n-1}+a_{n-2.}$

STRB R2,[R3,R4] -> Αποθηκεύουμε 1 byte(που επαρκεί για την αναπαράσταση) του n-οστό όρο της ακολουθίας στην αντίστοιχη n-οστή θέση του πίνακα.

MOV R0,R1 -> Μεταφέρουμε τα περιεχόμενα του α_{n-1} στον α_{n-2}

MOV R1,R2 -> Μεταφέρουμε τα περιεχόμενα του α_n στον α_{n-1}

ADD R4,R4,#1 -> Αυξάνουμε την τιμή του μετρητή κατά 1.

TEQ R4,#6 -> Συνθήκη τερματισμού του βρόχου(η τιμή του μετρητή ίση με 6) γιατί 6 στοιχεία έχει ο πίνακας και 6 είναι οι όροι της ακολουθίας που θέλουμε να υπολογίσουμε.