ΕΡΓΑΣΙΑ 5 ΕΡΓΑΣΤΗΡΙΟΥ ΒΑΣΙΚΩΝ ΘΕΜΑΤΩΝ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

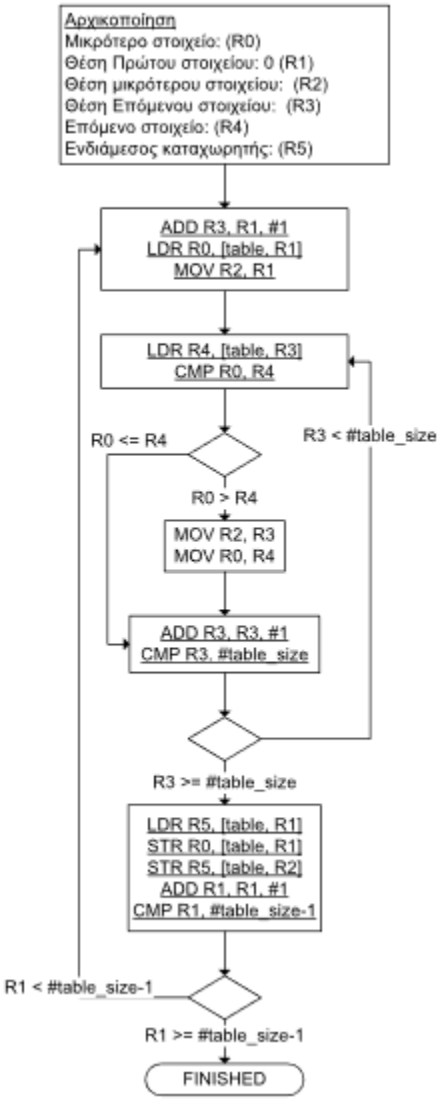
Συγγραφείς:

1) Βέργος Γεώργιος ,ΑΜ:1072604,email: up1072604@upnet.gr
2) Τσούλος Βασίλειος,ΑΜ:1072605,email: up1072605@upnet.gr

Εξάμηνο εκπόνησης της άσκησης:3ο

Ημερομηνία Παράδοσης:28/12/2020

Ερώτημα Πρώτο(i) – Υλοποίηση insertion sort / in – place



Η υπορουτίνα που χρησιμοποιήθηκε για το ερώτημα είναι ακολουθεί το παραπάνω διάγραμμα ροής είναι:

```
MOV R7,#0                                   @1
```

```
LDR R0,=table@--->''first adress of the array''   @2
MOV R1,#6@table_size                             @3
SUB R8,R1,#1                                     @4
BL Function                                      @5
Function:                                        @6
Label1:                                          @7
ADD R3,R7,#1                                     @8
LDRB R6,[R0,R7]                                  @9
MOV R2,R7                                        @10
Label2:                                          @11
LDRB R4,[R0,R3]                                  @12
CMP R6,R4                                        @13
BLS Label3                                       @14
MOV R2,R3                                        @15
MOV R6,R4                                        @16
Label3:                                          @17
ADD R3,R3,#1                                     @18
CMP R3,R1 @table_size                            @19
BCC Label2                                       @20
LDRB R5,[R0,R7]                                  @21
STRB R6,[R0,R7]                                  @22
STRB R5,[R0,R2]                                  @23
ADD R7,R7,#1                                     @24
CMP R7,R8@table_size-1                           @25
BCC Label1                                       @26
BX LR                                            @27
```

Επεξήγηση του παραπάνω κώδικα:

Εντολή @1

Αρχικοποιούμε τον καταχωρητή R7 = 0 ο οποίος σύμφωνα με το διάγραμμα ροής θα αποθηκεύει την θέση του πρώτου στοιχείου.

Εντολή @2

Αποθηκεύουμε στον καταχωρητή R0 την διεύθυνση που σηματοδοτεί η ετικέτα «Table» δηλαδή την αρχική διεύθυνση του πίνακα η οποία με αυτό το τρόπο περνιέται ως παράμετρος στην υπορουτίνα.

Εντολή @3

Μεταφέρουμε την τιμή 6 στον καταχωρητή R1 (έστω το 6 , μπορούμε να το αλλάξουμε σε 20 αν θέλουμε να συγκρίνουμε 20 στοιχεία) και γενικά το περιεχόμενο του R1 είναι το μέγεθος του πίνακα “Table” και περνιέται έτσι ως παράμετρος στην υπορουτίνα.

Εντολή @4

Αποθηκεύουμε στον καταχωρητή R8 την τιμή του καταχωρητή R1 μειωμένη κατά 1 γιατί στο διάγραμμα ροής γίνεται σύγκριση του περιεχομένου του καταχωρητή R7 με το μέγεθος του πίνακα μειωμένο κατά 1.

Εντολή @5

Κλήση της υπορουτίνας “Function” που υλοποιεί το insertion sort / in – place.

Εντολή @6

Η ετικέτα “Function” συμβολίζει από αυτή και κάτω τον κώδικα της υπορουτίνας “Function”.

Εντολή @7

Η ετικέτα “Label1” είναι συγκεκριμένη περιοχή του προγράμματος στην οποία το πρόγραμμα δέχεται να επιστρέψει ανάλογα άμα εκπληρώνεται κάποια συνθήκη ή όχι.

Εντολή @8

Αυξάνουμε την τιμή του καταχωρητή R7 κατά 1 και αποθηκεύουμε το αποτέλεσμα στον καταχωρητή R3.

Εντολή @9

Αποθηκεύουμε ένα byte του περιεχομένου του καταχωρητή R6 στην περιοχή της μνήμης που σηματοδοτείται από την ετικέτα “Table” + τιμή του καταχωρητή R7 , θέσεις μπροστά.

Εντολή @10

Μεταφέρουμε το περιεχόμενο του καταχωρητή R7 στον καταχωρητή R2.

Εντολή @11

Η ετικέτα “Label2” είναι συγκεκριμένη περιοχή του προγράμματος στην οποία το πρόγραμμα δέχεται να επιστρέψει ανάλογα άμα εκπληρώνεται κάποια συνθήκη ή όχι.

Εντολή @12

Φορτώνουμε στον καταχωρητή R4 ένα byte του περιεχομένου της περιοχής στη μνήμη που σηματοδοτείται από την ετικέτα “Table” + την τιμή του καταχωρητή R3 θέσεις μπροστά.

Εντολή @13

Συγκρίνουμε το περιεχόμενο του R6 με το περιεχόμενο του R7.

Εντολή @14

Αν R6 <= R4 , η ροή του προγράμματος μεταφέρεται ακριβώς κάτω από την ετικέτα “Label3”.

Εντολή @15

Μεταφέρουμε το περιεχόμενο του καταχωρητή R3 στο περιεχόμενο του καταχωρητή R2.

Εντολή @16

Μεταφέρουμε το περιεχόμενο του καταχωρητή R6 στο περιεχόμενο του καταχωρητή R6.

Εντολή @17

Η ετικέτα “Label3” είναι συγκεκριμένη περιοχή του προγράμματος στην οποία το πρόγραμμα δέχεται να επιστρέψει ανάλογα άμα εκπληρώνεται κάποια συνθήκη ή όχι.

Εντολή @18

Αυξάνουμε την τιμή του καταχωρητή R3 κατά 1.

Εντολή @19

Συγκρίνουμε το περιεχόμενο του καταχωρητή R3 με το περιεχόμενο του καταχωρητή R1.

Εντολή @20

Αν R3 < μεγέθους του πίνακα “Table” η ροή του προγράμματος μεταφέρεται αμέσως κάτω απ’ την ετικέτα “Label2”.

Εντολή @21

Φορτώνουμε στον καταχωρητή R5 ένα byte του περιεχομένου της περιοχής στη μνήμη που σηματοδοτεί η ετικέτα “Table” + τιμή του περιεχομένου του καταχωρητή R7 θέσεις μπροστά.

Εντολή @22

Αποθηκεύουμε ένα byte του περιεχομένου του καταχωρητή R6 στην περιοχή στη μνήμη που σηματοδοτεί η ετικέτα “Table” + τιμή του περιεχομένου του καταχωρητή R7 θέσεις μπροστά.

Εντολή @23

Αποθηκεύουμε ένα byte του περιεχομένου του καταχωρητή R5 περιοχής στη μνήμη που σηματοδοτεί η ετικέτα “Table” + τιμή του περιεχομένου του καταχωρητή R2 θέσεις μπροστά.

Εντολή @24

Αυξάνουμε την τιμή του περιεχομένου του καταχωρητή R7 κατά 1

Εντολή @25

Συγκρίνουμε την τιμή του περιεχομένου του καταχωρητή R7 με την τιμή του περιεχομένου του καταχωρητή R8 δηλαδή το μέγεθος του πίνακα μειωμένο κατά 1.

Εντολή @26

Αν R7 < του μεγέθους του πίνακα μειωμένο κατά 1 μεταφέρουμε την ροή του προγράμματος αμέσως κάτω από την ετικέτα “Label1”.

Εντολή @27

Αντίστοιχα με την εντολή MOV PC , LR αποθηκεύουμε το περιεχόμενο του Link Register στον program counter. Εδώ συγκεκριμένα η εντολή BX από το Branch and Exchange οδηγεί τον επεξεργαστή σε διακλάδωση στη διεύθυνση που βρίσκεται αποθηκευμένη στον καταχωρητή LR (R14).

Σημείωση!:

Ο παρακάνω κώδικας της υπορουτίνας δουλεύει πάνω σε μη προσημασμένους αριθμούς αυτό συμβαίνει για δύο λόγους:

Γενικά ο υπολογιστής για την αναπαράσταση των αρνητικών αριθμών χρησιμοποίει το συμπλήρωμα ως προς 2.

1)Χρησιμοποιούνται οι εντολές φόρτωσης LDRB και όχι LDRSB(load register with signed byte). Έτσι για παράδειγμα όταν στη μνήμη υπάρχει το byte FF μέσω της εντολής LDRB το πρόγραμμα θα φορτώσει στον καταχωρητή την τιμή 0x000000FF=255. Έτσι εφόσον κάθε byte έχει 8 bits θα αποθηκευτεί στον καταχωρητή με άλλα 24 μηδενικά αριστερά του. Το πρόγραμμα κατά την σύγκριση του με άλλους αριθμούς θα το θεωρήσει θετικό(σε συμπλήρωμα ως προς δύο) με τα ίδια αποτελέσματα αν ήταν μη προσημασμένος.

2) Χρησιμοποιούμε εντολές υπό συνθήκη :

BCC(Usigned lower than).

BLS(Unsigned lower or same).

Αν θέλουμε το πρόγραμμα να δουλεύει με προσημασμένους αριθμούς θα πρέπει να κάνουμε τις εξής αλλαγές:

1)Αλλαγή των εντολών φόρτωσης LDRB σε LDRSB. Έτσι για παράδειγμα όταν στη μνήμη υπάρχει το byte FF μέσω της εντολής LDRB το πρόγραμμα θα φορτώσει στον καταχωρητή μέσω επέκτασης προσήμου την τιμή 0xFFFFFFFF=-1. Το πρόγραμμα κατά την σύγκριση του με άλλους αριθμούς θα το θεωρήσει αρνητικό(σε συμπλήρωμα ως προς δύο). Έτσι το πρόγραμμα μας θα χειρίζεται προσημασμένους αριθμούς.

2) Μετάβαση στις ακόλουθες εντολές υπό συνθήκη:

BCC→BLT(Signed branch if lower than)

BLS→BLE(Signed branch if lower or equal than)

Άρα συνοπτικά πρέπει να γίνουν οι εξής αλλαγές:

LDRB R6,[R0,R7]→LDRSB R6,[R0,R7]

LDRB R4,[R0,R3]→LDRSB R4,[R0,R3]

LDRB R5,[R0,R7]→LDRSB R5,[R0,R7]

BLS Label3→BLE Label3

BCC Label2→BLT Label2

BCC Label1→BLT Label1

Έτσι η υπορουτίνα μας θα χειρίζεται προσημασμένους αριθμούς.

Στον φάκελο με τα αρχεία της εργασίας υπάρχουν οι κώδικες αναλυτικά για το πρώτο ερώτημα με τη υπορουτίνα main , .data (βάζουμε 6 αριθμούς όπως στο παράδειγμα της εκφώνησης) .

Ερώτημα Δεύτερο(ii) – Εκτέλεση αλγορίθμου και επιβεβαίωση ορθότητας αποτελεσμάτων

Η υπορουτίνα που χρησιμοποιήθηκε για το παρόν ερώτημα:

Check_if_sorted:	@1
MOV R3,#0@-->metritis	@2
LOOP:	@3
LDRSB R1,[R0,R3]	@4
ADD R3,R3,#1	@5
LDRSB R2,[R0,R3]	@6
CMP R1,R2	@7
ADDGT R4,R4,#1	@8
TEQ R3,#R8	@9
BNE LOOP	@10
BX LR	@11

Εντολή @1

Η ετικέτα “Check _if _sorted” σηματοδοτεί την έναρξη του κώδικα της υπορουτίνας “Check_if_sorted”.

Εντολή @2

Αρχικοποιούμε τον καταχωρητή μετρητή R3 με 0 για να προσπελάσουμε τον πίνακα “Table”.

Εντολή @3

Η ετικέτα “Loop” θα χρησιμεύσει για επαναλήψεις προκειμένου να προσπελάσουμε τα στοιχεία του πίνακα “Table”.

Εντολή @4

Φορτώνουμε στον καταχωρητή R1 , 1 byte του περιεχομένου της περιοχής της μνήμης που σηματοδοτεί η ετικέτα “Table” + τιμή του καταχωρητή R3 θέσεις μπροστά με επέκταση προσήμου προκειμένου να χειριζόμαστε προσημασμένους αριθμούς και γενικά χρησιμεύει να αποθηκεύει το τρέχον στοιχείο του πίνακα που προσπελαύνουμε.

Εντολή @5

Αυξάνουμε την τιμή του καταχωρητή – μετρητή R3 κατά 1 προκειμένου να προσπελάσουμε το αμέσως επόμενο στοιχείο του “Table”.

Εντολή @6

Φορτώνουμε στον καταχωρητή R2 , 1 byte του περιεχομένου της περιοχής της μνήμης που σηματοδοτεί η ετικέτα “Table” + τιμή του καταχωρητή R3 θέσεις μπροστά με επέκταση προσήμου προκειμένου να χειριζόμαστε προσημασμένους αριθμούς και γενικά χρησιμεύει να αποθηκεύει το αμέσως επόμενο του τρέχοντος στοιχείου του πίνακα που προσπελαύνουμε.

Εντολή @7

Συγκρίνουμε τις τιμές των καταχωρητών R1 και R2 μέσο εικονικής αφαίρεσης R1 – R2.

Εντολή @8

Σε περίπτωση που ο R1 > R2 δηλαδή δεν είναι ταξινομημένα τα στοιχεία στη μνήμη των οποίων οι τιμές είναι αποθηκευμένες στους R1 και R2 αύξησε τον R4 κατά 1. Δηλαδή ο καταχωρητής R4 μετράει πόσα λάθος ταξινομημένα ζευγάρια υπάρχουν στον πίνακα. Έτσι έστω και ένα λάθος ταξινομημένο ζευγάρι να υπάρχει θα ξέρουμε ότι ο πίνακας δεν είναι ταξινομημένος.

Εντολή @9

Ελέγχουμε ως προς την ισότητα τις τιμές των καταχωρητών R3 και R8 δηλαδή ελέγχουμε αν έχουμε ελέγξει όλα τα ζευγάρια του πίνακα “Table”. Γενικά για πίνακα N στοιχείων θα συγκρίνουμε N-1 ζευγάρια

Εντολή @10

Από το Branch if Not Equal (BNE) αν δεν έχουμε ελέγξει όλα τα ζευγάρια του πίνακα ως προς την αύξουσα σειρά μεταφέρουμε την ροή του προγράμματος αμέσως κάτω απ' την ετικέτα "Loop".

Εντολή @11

Αντίστοιχα με την εντολή MOV PC , LR αποθηκεύουμε το περιεχόμενο του Link Register στον program counter. Εδώ συγκεκριμένα η εντολή BX από το Branch and Exchange οδηγεί τον επεξεργαστή σε διακλάδωση στη διεύθυνση που βρίσκεται αποθηκευμένη στον καταχωρητή LR (R14).

Από κάτω παρατίθεται ολόκληρος ο κώδικας δηλαδή η συνάρτηση main,η υπορουτίνα Function που υλοποίει το Insertion Sort/In Place και η υπορουτίνα Check_if_sorted :

```
.arm

.text

.global main

main:

MOV R7,#0

LDR R0,=table@--->''first adress of the array''

MOV R1,#20@table_size για το ερώτημα ii που ζητά 20 τυχαίους αριθμούς.

SUB R8,R1,#1

BL Function

MOV R4,#0

BL Check_if_sorted

Function:

STMDB R13!,{R0-R8,R14}

Label1:

ADD R3,R7,#1

LDRSB R6,[R0,R7]

MOV R2,R7

Label2:

LDRSB R4,[R0,R3]

CMP R6,R4

BLE Label3
```

```
MOV R2,R3
MOV R6,R4
Label3:
ADD R3,R3,#1
CMP R3,R1 @table_size
BLT Label2
LDRSB R5,[R0,R7]
STRB R6,[R0,R7]
STRB R5,[R0,R2]
ADD R7,R7,#1
CMP R7,R8@table_size-1
BLT Label1
BX LR

Check_if_sorted:
MOV R3,#0@-->Counter
LOOP:
LDRSB R1,[R0,R3]
ADD R3,R3,#1
LDRSB R2,[R0,R3]
CMP R1,R2
ADDGT R4,R4,#1
TEQ R3,R8
BNE LOOP
BX LR
.data
table:
.byte .byte 0x05,0x0D,0x0D,0x0C,0x07,0x09,0x08,0x03,0x02,0x04,0x01,0x0A,0x-1,0x-3,0x-4,0x-2,0x00,0x-5,0x0B,0x0E
```

Παρακάτω παρατίθεται ο αντίστοιχος κώδικας για χειρισμό μη προσημασμένων αριθμών με 20 τυχαίους αριθμούς στον πίνακα table:

```
.arm

.text

.global main

main:

MOV R7,#0

LDR R0,=table@--->''first adress of the array''

MOV R1,#20@table_size

SUB R8,R1,#1

BL Function

MOV R4,#0

BL Check_if_sorted

Function:

STMDB R13!,{R0-R8,R14}

Label1:

ADD R3,R7,#1

LDRB R6,[R0,R7]

MOV R2,R7

Label2:

LDRB R4,[R0,R3]

CMP R6,R4

BLS Label3

MOV R2,R3

MOV R6,R4

Label3:

ADD R3,R3,#1

CMP R3,R1 @table_size

BCC Label2
```

```
LDRB R5,[R0,R7]

STRB R6,[R0,R7]

STRB R5,[R0,R2]

ADD R7,R7,#1

CMP R7,R8@table_size-1

BCC Label1

BX LR


Check_if_sorted:

MOV R3,#0@-->metritis

LOOP:

LDRB R1,[R0,R3]

ADD R3,R3,#1

LDRB R2,[R0,R3]

CMP R1,R2

ADDHI R4,R4,#1@(Από το Usigned Higher than)

TEQ R3,R8

BNE LOOP

BX LR

.data

table:

.byte
0x05,0x0D,0x0D,0x0C,0x07,0x09,0x08,0x03,0x02,0x04,0x01,0x0A,0x0F,0xFF,0xFD,0xBA,0x00,0x
AB,0x0B,0x0E
```