ΕΡΓΑΣΤΗΡΙΟ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

Τριαντάφυλλος Πράππας ΑΜ:1067504

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 5

i. Υλοποίηση Insertion sort/in-place

```
.arm
.text
.global main
main:
STMDB R13!,{R0-R12,PC}
MOV R1,#0
LDR R6,=table
BL Loop1
end:
LDMIA R13!,{R0-R12,PC}
Loop1:
STMDB R13!,{R0-R6}
ADD R3,R1,#1
LDRB R0,[R6,R1]
MOV R2,R1
Loop2:
LDRB R4,[R6,R3]
CMP R0,R4
BLS Loop3
MOV R2,R3
MOV R0,R4
Loop3:
ADD R3,R3,#1
CMP R3,#20
BLO Loop2
LDRB R5,[R6,R1]
STRB R0,[R6,R1]
STRB R5,[R6,R2]
ADD R1,R1,#1
CMP R1,#19
BLO Loop1

BHS end


.data
table:
.byte 0x07
```

```
.byte 0x03
.byte 0x04
.byte 0x05
.byte 0x0F
.byte 0x11
.byte 0x13
.byte 0x0A
.byte 0x01
.byte 0x0D
.byte 0x02
.byte 0x0C
.byte 0x06
.byte 0x08
.byte 0x12
.byte 0x09
.byte 0x0B
.byte 0x14
.byte 0x0E
.byte 0x10
```

Για τον συγκεκριμένο κώδικα τα κυριότερα στοιχεία τα έχουμε πάρει απευθείας από το φυλλάδιο των ασκήσεων και βασισμένοι σε αυτά προσθέσαμε τα κατάλληλα Loops έτσι ώστε να μπορέσουμε να ταξινομήσουμε τους 20 τυχαίους αριθμούς που ζητάει η εκφώνηση (για λόγους ευκολίας θα χρησιμοποιήσουμε τους αριθμούς από το 1-20). Αρχικά θέτουμε στον καταχωρητή R1 την τιμή 0 ,ο R1 παίζει σημαντικό ρόλο για τον τερματισμό των Loops. Καθώς επίσης θέτουμε στον R6 τις τιμές της ετικέτας table. Στην συνέχεια με την BL Loop1 καλούμε την υπορουτίνα. Τις εντολές αυτές τις βρίσκουμε από το φυλλάδιο και απλώς κάνουμε και ένα STMD έτσι ώστε να αποθηκεύσουμε τους καταχωρητές που θα χρησιμοποιήσουμε. Το ADD R3 θα μας βοηθήσει στο Loop2 έτσι ώστε να καταφέρουμε κάθε φορά να παίρνουμε τον επόμενο αριθμό από αυτόν που έχει ο R0. Στην συνέχεια με το LDRB ο R0 παίρνει την τιμή 0x07 (δηλαδή την πρώτη τιμή της ετικέτας table αφού το R1 είναι 0). Και στην συνέχεια μετακινούμαι την τιμή του R1 στον καταχωρητή R2. Στην συνέχεια μπαίνουμε στο Loop2 στο οποίο για αρχή καλούμε με την LDRB να πάρει ο R4 την επόμενη τιμή της ετικέτας (αν πχ αν πρώτα ο R0 πάρει την 0x07 ο R4 θα πάρει την 0x03). Και ύστερα κάνουμε CMP των τιμών των R0 και R4 και ύστερα με την BLS Loop3 αν R0<=R4 θα ξεκινήσει να εκτελείται το Loop3 αλλιώς αν R0>R4 θα εκτελεστούν οι 2 εντολές MOV που δίνονται. Στην συνέχεια το Loop3 έχει την εντολή ADD η οποία θα βοηθήσει στο να σταματήσει το Loop3 λόγω του CMP που έχουμε. Με το BLO Loop2 ουσιαστικά θα κάνουμε τις συγκρίσεις που χρειάζονται έτσι ώστε να φέρουμε μπροστά τον μικρότερο αριθμό από τους οποίους έχουμε εισάγει το οποίο επιτυγχάνεται με τις εντολές που μας δίνονται. Και τέλος κάνουμε τις εντολές ADD R1,R1,#1 και CMP R1,#19 με τις οποίες ουσιαστικά θα μας επιτρέψει να κάνουμε 20 φορές τα Loops για να ταξινομηθούν όλοι οι αριθμοί. Τέλος έχουμε και την BHS end με την οποία όταν έχουν γίνει όλα τα απαραίτητα Loops θα μας πάει στην end: και θα καθαριστεί το περιεχόμενο των καταχωρητών.

Τρέχοντας την παραπάνω υπορουτίνα αν ψάξουμε στα memory την διεύθυνση που είναι αποθηκευμένα τα byte θα παρατηρήσουμε ότι μέχρι να τελειώσει ο κώδικας όλα θα έχουν τοποθετηθεί με την σωστή σειρά δηλαδή 0x1 …. 0x14 άρα ο αλγόριθμος κάνει αυτό που θα έπρεπε (απλώς θα ήθελα να πω σε αυτό το σημείο πως και σε αυτή την άσκηση το πρόγραμμά μου δεν εμφάνιζε στην θέση μνήμης το τελευταίο byte σε αυτή την περίπτωση το 0x10 και το αντικαθιστούσε με 0x00 ενώ στον συνεργάτη μου δεν συνέβαινε αυτό, θα

μπορούσατε μήπως να μου πείτε τι πιστεύετε ότι μπορεί να φταίει γιατί ένα ανάλογο θέμα το είχα και με την προηγούμενη άσκηση γιατί δεν μου εμφάνιζε τα .data και δεν μπορούσα να τεστάρω και μόνος τον κώδικα και δεν θέλω να συμβεί κάτι παρόμοιο και στην εξέταση)