ΕΡΓΑΣΤΗΡΙΟ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

Τριαντάφυλλος Πράππας ΑΜ:1067504

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4

i)Υπολογισμός μαθηματικού τύπου

```
.arm
 .text
 .global main
main:
 STMDB R13!, {R0-R12,R14}
 MOV R5,#5
 MOV R6,#0
 MOV R7, #0
 MOV R8, #4

 BL Loop
end:
 LDMIA R13!, {R0-R12, R14}
 MOV PC, R14

Loop:
 STMDB R13!, {R2-R4}
 LDR R0, =Values
 LDR R1, =Const
 ADD R0,R0,R6
 LDRB R2,[R0,#0]
 LDRB R4,[R1,#0]
 MUL R2,R4,R2
 LDRB R3,[R0,#1]
 LDRB R4,[R1,#1]
 MUL R3, R4, R3
 ADD R2, R2, R3
 LDRB R3,[R0,#2]
 LDRB R4,[R1,#2]
 MUL R3,R4,R3
 SUB R2,R2,R3
 MUL R2, R5,R2
 MOV R0,R2,LSR #6
 CMP R0,R9
 ADD R6, R6,#3
 ADD R7,R7,#1
CMP R7,R8
BMI Loop
```

```
END:
LDMIA R13!, {R2-R4}
MOV PC, R14

.data
Values:
.byte 0x02, 0x03, 0x04
.byte 0x10, 0x05, 0x06
.byte 0x0B, 0x02, 0x0D
.byte 0x01, 0x0C, 0x08

Const:
.byte 0x04, 0x07, 0x05
```

Επανάληψη	Αποτέλεσμα
1	0x00000000
2	0x00000005
3	0x03ffffff
4	0x00000003

Για τον συγκεκριμένο κώδικα στην αρχή εκτός από την εντολή με την οποία δηλώνουμε τους καταχωρητές που θα χρησιμοποιήσουμε στην συνέχεια εισάγουμε τιμές και σε κάποιους καταχωρητές. Στον R5 εισάγουμε την τιμή 5 η οποία χρειάζεται για την αριθμητική πράξη που θα χρησιμοποιήσουμε. Στην συνέχεια στον R6 εισάγουμε την τιμή 0 ο σκοπός που χρησιμοποιούμε τον R6 είναι για να κάνουμε αυτό το +3 που θέλουμε να επιτύχουμε με τα a,b,c και για τον λόγο αυτό το Loop υπάρχουν και οι εντολές ADD R0,R0,R6 και ADD R6,R6,#3 το R7 θα το χρησιμοποιήσουμε έτσι ώστε να μπορέσουμε να βγούμε από το loop για αυτό μέσα στο Loop έχουμε την εντολή ADD R7,R7,#1 η οποία με την CMP που συγκρίνει τον R7 με τον R8 στον οποίο έχουμε εισάγει την τιμή 4 για να μπορέσουμε να βγούμε από το Loop αφού εκτελεστούν οι απαιτούμενες πράξεις. Στην συνέχεια ξεκινάμε να φτιάχνουμε την υπορουτίνα μας στην αρχή πρέπει να δημιουργήσουμε το end το οποίο θα εκτελεσθεί τελευταίο και αφού τελειώσουν όλες οι πράξεις θα μας ξεκινήσει ο κώδικας από την main: . Ύστερα πρέπει να δημιουργήσουμε το βασικό Loop σε αυτό το loop αφού ορίσουμε ποιους καταχωρητές θα χρησιμοποιήσουμε στην συνέχεια πρέπει να εισάγουμε τις διευθύνσεις που σηματοδοτούν οι ετικέτες Values, Const στην συνέχεια αφού έχουμε κατανοήσουμε πως λειτουργεί η θέση μνήμης που είναι αποθηκευμένη η Values θα πρέπει να καταλάβουμε ότι για να μπορέσουμε να εμφανίσουμε όλες τις τιμές με τον σωστό τρόπο θα πρέπει να εκτελείται σε κάθε Loop η εντολή ADD R0,R0,R6 η οποία ουσιαστικά την πρώτη φορά δεν θα αλλάξει γιατί είναι +0 και έτσι θα καταφέρουμε να πάρουμε τα a1,b1,c1 την επόμενη φορά όμως για να πάρουμε τα a2,b2,c2 θα πρέπει να εκτελέσουμε την εντολή ADD R6, R6,#3 η οποία ουσιαστικά θα «μετακινήσει» την θέση μνήμης κατά 3 με αποτέλεσμα όταν τρέξουμε τις εντολές που εισάγουν τιμές στους καταχωρητές να πάρουμε τα σωστά ai,bi,ci σε κάθε Loop.Στην συνέχεια κάνουμε όλες τις ζητούμενες

πράξεις και ταυτόχρονα αλλάζουμε τις τιμές των καταχωρητών. Τέλος για να κάνουμε την διαίρεση που μας ζητάει θα πρέπει να κάνουμε μια ολίσθηση προς τα δεξιά κατά 6 θέσεις και ύστερα να ελέγξουμε με το CMP αν ο αριθμητής είναι ίσος με το 0. (Μετά από κάποιες δοκιμές παρατηρήσαμε ότι για τις θετικές διαιρέσεις εμφάνιζε το σωστό αποτέλεσμα ενώ πχ για την 3 επανάληψη που βγαίνει διαίρεση αρνητικού με θετικού βγαίνει λάθος αποτέλεσμα). Τέλος αφού εκτελεστεί το Loop 4 φορές περνάμε στην END: η οποία αφού εκτελεστεί θα μας πάει στην end: και το πρόγραμμα θα ξανά αρχίσει από την main όπως αναφέρθηκε και πριν.

ii) Εύρεση μέγιστης τιμής σε πίνακα αποτελεσμάτων

```
.arm
.text
.global main
main:
STMDB R13!, {R0-R12,R14}
MOV R5,#5
MOV R6,#0
MOV R7, #0
MOV R8, #4

BL Loop
end:
LDMIA R13!, {R0-R12, R14}
MOV PC, R14

Loop:
STMDB R13!, {R2-R4}
LDR R0, =Values
LDR R1, =Const
ADD R0,R0,R6
LDRB R2, [R0]
LDRB R4,[R1]
MUL R2,R4,R2
LDRB R3,[R0,#1]
LDRB R4,[R1,#1]
MUL R3, R4, R3
ADD R2, R2, R3
LDRB R3,[R0,#2]
LDRB R4,[R1,#2]
MUL R3,R4,R3
SUB R2,R2,R3
MUL R2, R5,R2
MOV R0,R2,LSR #6
CMP R0,R9
MOVGT R9,R0
MOVGT R10,R7
ADD R6, R6,#3
ADD R7,R7,#1
```

```
CMP R7,R8
BMI Loop
STRB R9,[R1,#3]
STRB R10,[R1,#4]

END:
LDMIA R13!, {R2-R4}
MOV PC, R14

.data
Values:
.byte 0x02, 0x03, 0x04
.byte 0x10, 0x05, 0x06
.byte 0x0B, 0x02, 0x0D
.byte 0x01, 0x0C, 0x08

Const:
.byte 0x04, 0x07, 0x05
```

Για τον συγκεκριμένο κώδικα δεν έχουμε να πούμε πολλά ισχύουν πάνω κάτω τα ίδια με τον προηγούμενο κώδικα με την διαφορά ότι έχουμε προσθέσει κάποιες παραπάνω εντολές με τις οποίες από τα αποτελέσματα που παράγει παίρνει το πιο μεγάλο και το εμφανίζει στον καταχωρητή 9. Οι εντολές οι οποίες χρησιμοποιούμε είναι η MOVGT η οποία ουσιαστικά είναι συνδυασμός της εντολής MOV και του GT (GREATER THAN). Και με το το MOVGT R10,R7 επιλέγει ποια επανάληψη είναι αυτή με την μεγαλύτερη τιμή. (σαν παρατήρηση θα πω ότι λόγο του ότι η διαίρεση με τον αρνητικό βγάζει αριθμό μεγαλύτερο οπότε ο καταχωρητής παίρνει την συγκεκριμένη στιγμή). Επίσης χρησιμοποιούμε και 2 STRB για να αποθηκεύσουμε τα αποτελέσματα.

iii)Υπολογισμός πολυωνύμου

```
.arm
.text
.global main

main:
STMDB R13!, {R0-R12,R14}
LDR R0,=Values
LDR R1,=Const
LDR R2,=Result
MOV R7,#4

BL Loop
end:
LDMIA R13!,{R0-R12,PC}

Loop:
STMDB R13!, {R3-R6}
MOV R6,#6
LDR R3,[R0]
```

```
ADD R1,R1,#6
LDRB R4,[R1]
MOV R5,R4

ContLoop:
MUL R5,R3,R5
SUB R1,R1,#1
LDRB R4,[R1]
ADD R5,R5,R4
STR R5,[R2]
ADD R2,R2,#4
SUBS R6,R6,#1
BHI ContLoop
ADD R0,R0,#4
LDR R3,[R0]
SUBS R7,R7,#1
BHI Loop
LDMIA R13!,{R3-R6}
MOV PC, R14

.data
Result:
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0
.word 0

Values:
.word 0x10
.word 0x50A
```

```
.word 0xCDCA
.word 0x80AB

Const:
.byte 0x04, 0x07, 0x05
.byte 0x20, 0x1A, 0x12, 0x06
```

Για τον συγκεκριμένο κώδικα πρέπει να παρατηρήσουμε την σημείωση που υπάρχει και να καταλάβουμε τον τρόπο με τον οποίο πρέπει να γίνει η πράξη. Αρχικά πρέπει να δηλώσουμε τις ετικέτες με τις διευθύνσεις των μεταβλητών. Ύστερα πρέπει να δώσουμε στον R7 τιμή 4 έτσι ώστε να γίνει ο απαραίτητος αριθμός επαναλήψεων. Στην συνέχεια κάνουμε ένα end: για να λήξει ο κώδικας και να αρχίσει από το main: . Μετά πρέπει να κάνουμε ένα MOV με τιμή #6 το οποίο θα μας βοηθήσει να κάνουμε τις 6 απαραίτητες πράξεις που πρέπει να γίνουν. Στην συνέχεια με την πρώτη LDR παίρνουμε την πρώτη τιμή που έχουμε στο Values και μετά με το ADD παίρνουμε την τελευταία τιμή που έχουμε στο Const γιατί για την πράξη πρέπει να πάμε από το 6 προς το 1. Στην συνέχεια κάνουμε τον πολλαπλασιασμό και μειώνουμε το R1 κατά 1 για να πάρουμε την αμέσως επομένη τιμή και κάνουμε την πρόσθεση και αποθηκεύουμε την πράξη στο Results και προσθέτουμε στο R2 το #4 για να πάμε στην αμέσως επόμενη θέση για να αποθηκεύσουμε το αμέσως επόμενο word που θα πάρουμε και κάνουμε ξανα SUB για να πάρουμε την επόμενη τιμή του Const. Και αυτό θα γίνει όσα είναι τα Const. Ύστερα με το ADD R0 θα πάρουμε το επόμενο word και με το subs 7 ουσιαστικά θα μας βγάζει από το Loop όταν τελειώσουν τα Values. Τέλος αφού γίνουν τα Loop γίνεται καθαρισμός καταχωρητών και πηγαίνουμε στο end: για να τελειώσει ο κώδικας

Επίσης θα πρέπει να αναφέρω ότι είχε κάποιο θέμα το KEIL για την τελευταία άσκηση και δεν είμαι και πολύ σίγουρος για κάποιο λόγο στο KEIL μου δεν μου δεχόταν όλα τα Const.