8

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4 – ΟΜΑΔΑ Α5

Μηλτιάδης Μαντές AM 1084661 EMAIL up1084661@upnet.gr

Ελπίδα Κόκκαλη AM 1084648 EMAIL up1084648@upnet.gr

i. Υπολογισμός μαθηματικού τύπου

Επανάληψη	Αποτέλεσμα
1	00
2	05
3	FF
4	03

```
.arm

.text

.global main


main:

STMDB R13!, {R0-R12,R14}

LDR R1, =Const @Θέτουμε τον R1 να δείχνει στις διευθύνσεις του πίνακα Const

LDR R2, =Result @Θέτουμε τον R2 να δείχνει στις διευθύνσεις του πίνακα Result

MOV R3, #0 @Μεταφέρουμε στον R3 τη τιμή 0 και τον χρησιμοποιούμε για να διαβάζουμε τα bytes στις διευθύνσεις των πινάκων

MOV R4, #0 @Μεταφέρουμε στον R4 τη τιμή 0 και τον χρησιμοποιούμε για να διαβάζουμε τα bytes στις διευθύνσεις των πινάκων

BL Subrtn @Καλούμε την υποροutίνα 4 φορές για τον υπολογισμό των x_i

BL Subrtn

BL Subrtn

BL Subrtn

LDMIA R13!, {R0-R12,R14} @Επανακτούμε το περιεχόμενο των καταχωρητών που είχαμε σώσει

MOV PC,LR @Επιστρέφουμε από την υποροutίνα στο σημείο όπου κλήθηκε
```

Subrtn:

STMDB R13!, {R0-R11,R14}

LDR R0, =Values **@Θέτουμε τον R0 να δείχνει στις διευθύνσεις του πίνακα Values**

ADD R0,R0,R4

MOV R5, #5 **@Μεταφέρουμε στον R5 τη τιμή 5**

LDRB R6, [R0,#0] **@Μεταφέρουμε στον R6 το byte της διεύθυνσης μνήμης όπου δείχνει ο R0**

LDRB R7, [R0,#1] **@Μεταφέρουμε στον R7 το byte της επόμενης διεύθυνσης μνήμης απ'όπου δείχνει ο R0**

LDRB R8, [R0,#2] **@Μεταφέρουμε στον R8 το byte της μεθεπόμενης διεύθυνσης μνήμης απ'όπου δείχνει ο R0**

LDRB R9, [R1,#0] **@Μεταφέρουμε στον R9 το byte της διεύθυνσης μνήμης όπου δείχνει ο R1**

LDRB R10, [R1,#1] **@Μεταφέρουμε στον R10 το byte της επόμενης διεύθυνσης μνήμης απ'όπου δείχνει ο R1**

LDRB R11, [R1,#2] **@Μεταφέρουμε στον R11 το byte της μεθεπόμενης διεύθυνσης μνήμης απ'όπου δείχνει ο R1**

MUL R6,R9,R6 **@Πολλαπλασιάζουμε τα bytes των διευθύνσεων μνήμης όπου δείχνουν οι R6, R9 (a_i, z_0) και αποθηκεύουμε το αποτέλεσμα στον R6**

MUL R7,R10,R7 **@Πολλαπλασιάζουμε τα bytes των διευθύνσεων μνήμης όπου δείχνουν οι R7, R10 (b_i, z_1) και αποθηκεύουμε το αποτέλεσμα στον R7**

MUL R8,R11,R8 **@Πολλαπλασιάζουμε τα bytes των διευθύνσεων μνήμης όπου δείχνουν οι R8, R11 (c_i, z_2) και αποθηκεύουμε το αποτέλεσμα στον R8**

ADD R6,R7,R6 **@Αθροίζουμε στην ουσία το a_i*z_0 με το b_i*z_1 και αποθηκεύουμε το αποτέλεσμα στον R6**

SUB R6,R6,R8 **@Αφαιρούμε στην ουσία από το b_i*z_1 το c_i*z_2 και αποθηκεύουμε το αποτέλεσμα στον R6**

MUL R0,R5,R6 **@Πολλαπλασιάζουμε το 5 που βρίσκεται στον καταχωρητή R5 με την παράσταση $a_i*z_0 + b_i*z_1 - c_i*z_2$ που βρίσκεται στον καταχωρητή R6 και αποθηκεύουμε το αποτέλεσμα στον R0**

```
MOV R0,R0,LSR #6 @Κάνουμε δεξιά ολίσθηση του περιεχομένου που βρίσκεται στον R0 κατά 6 θέσεις για να διαιρέσουμε όλη τη παράσταση με το 64(2^6)

STRB R0, [R2,R3] @Φορτώνουμε στον R0 το byte της διεύθυνσης μνήμης όπου δείχνει ο R2

ADD R4,R4,#3 @Αυξάνουμε το περιεχόμενο του R4 κατά 3

ADD R3,R3,#1 @Αυξάνουμε το περιεχόμενο του R3 κατά 1

LDMIA R13!,{R1-R6}

MOV PC,LR


.data

Values:

.byte 0x02, 0x03, 0x04

.byte 0x10, 0x05, 0x06

.byte 0x0B, 0x02, 0x0D

.byte 0x01, 0x0C, 0x08

Const:

.byte 0x04, 0x07, 0x05

Result:

.byte 0,0,0,0
```

ii. Εύρεση μέγιστης τιμής σε πίνακα αποτελεσμάτων

```
.arm

.text

.global main


main:

STMDB R13!, {R0-R12,R14}

LDR R1, =Const
```

```
LDR R2, =Result

MOV R3, #0

MOV R4, #0

BL Subrtn

BL Subrtn

BL Subrtn

BL Subrtn
```

LDRB R4,[R2,#0] **@Μεταφέρουμε στον R4 το byte της διεύθυνσης μνήμης όπου δείχνει ο R2(δηλαδή το x_0)**

MOV R3,#1 **@Μεταφέρουμε στον R3 τη τιμή 1 για να μεταφερθούμε στο byte της επόμενης διεύθυνσης μνήμης όπου δείχνει ο R2 όταν μπούμε μέσα στον βρόχο επανάληψης**

Startpoint:

LDRB R12,[R2,R3] **@Μεταφέρουμε στον R12 το byte της επόμενης διεύθυνσης μνήμης όπου δείχνει ο R2 (δηλαδή το x_1)**

CMP R12,R4 **@Συγκρίνουμε το περιεχόμενο του R12 με του R4 (δηλαδή τα x_i ανά δύο κάθε φορά) με τη πράξη της αφαίρεσης R4-R12**

MOVHI R4,R12 **@Αν το περιεχόμενο του R12 είναι μεγαλύτερο από αυτό του R4 τότε αποθηκεύουμε στον R4 τη τιμή του R12**

ADD R3,R3,#1 **@Αυξάνουμε το περιεχόμενο του R3 κατά 1 για να μεταβούμε στο byte της επόμενης διεύθυνσης όπου δείχνει ο R2**

CMP R3,#3 **@Συγκρίνουμε το περιεχόμενο του R3 με το 3 ώστε να πραγματοποιηθούν 4 επαναλήψεις**

BLE Startpoint

STRB R4,[R1,#3] **@Αποθηκεύουμε στο 4° byte του πίνακα Const το περιεχόμενο του R4 ο οποίος περιέχει το μέγιστο αποτέλεσμα που δίνει η παράσταση**

STRB R3,[R1,#4] **@Αποθηκεύουμε στο 5° byte του πίνακα Const το περιεχόμενο του R3 ο οποίος περιέχει το νούμερο του συνόλου**

```
LDMIA R13!, {R0-R12,R14}

MOV PC,LR
```

```
Subrtn:
STMDB R13!, {R0-R11,R14}
LDR R0, =Values
ADD R0,R0,R4
MOV R5, #5
LDRB R6, [R0,#0]
LDRB R7, [R0,#1]
LDRB R8, [R0,#2]
LDRB R9, [R7,#0]
LDRB R10, [R7,#1]
LDRB R11, [R7,#2]
MUL R6,R9,R6
MUL R7,R10,R7
MUL R8,R11,R8
ADD R6,R7,R6
SUB R6,R6,R8
MUL R0,R5,R6
MOV R0,R0,ASR #6
STRB R0, [R2,R3]
ADD R4,R4,#3
ADD R3,R3,#1
LDMIA R13!,{R1-R6}
MOV PC,LR


.data
Values:
.byte 0x02, 0x03, 0x04
```

```
.byte 0x10, 0x05, 0x06
.byte 0x0B, 0x02, 0x0D
.byte 0x01, 0x0C, 0x08
Const:
.byte 0x04, 0x07, 0x05,0,0
Result:
.byte 0,0,0,0
```

iii. Υπολογισμός πολυωνύμου

```
.arm
.text
.global main

main:
STMDB R13!, {R0-R12, R14}
MOV R4, #0
BL Subrtn
BL Subrtn
BL Subrtn
BL Subrtn
LDMIA R13!, {R0-R12, R14}
MOV PC,LR

Subrtn:
STMDB R13!, {R1-R12, R14}
MOV R12, #0
LDR R1,=Values @Θέτουμε τον R1 να δείχνει στις διευθύνσεις του πίνακα Values που περιέχει τις τιμές του x
```

LDR R1,[R1,R4] **@Μεταφέρουμε στον R1 το byte της διεύθυνσης μνήμης όπου δείχνει ο R1**

LDR R0,=Const **@Θέτουμε τον R0 να δείχνει στις διευθύνσεις του πίνακα Const**

LDRB R0,[R0,R12] **@Μεταφέρουμε στον R0 το byte της διεύθυνσης μνήμης όπου δείχνει ο R0**

ADD R8,R0,#0 **@Στην ουσία αποθηκευούμε στον R8 το a_0**

ADD R12,R12,#1 **@Αυξάνουμε το περιεχόμενο του R12 κατά 1 για να μεταβούμε στο byte της επόμενης διεύθυνσης όπου δείχνει ο R0 ($2^η$ τιμή του x)**

LDR R0,=Const **@Θέτουμε τον R0 να δείχνει στις διευθύνσεις του πίνακα Const**

LDRB R0,[R0,R12] **@Μεταφέρουμε στον R0 το byte της διεύθυνσης μνήμης όπου δείχνει ο R0**

MUL R9,R1,R0 **@Πολλαπλασιάζουμε τα περιεχόμενα των R1, R0 και το αποτέλεσμα το αποθηκεύουμε στον R9 (δηλαδή κάνουμε $x*a_1$)**

ADD R8,R8,R9 **@Προσθέτουμε τα περιεχόμενα των R8, R9 και το αποτέλεσμα (a_1*x+a_0) το αποθηκεύουμε στον R8**

ADD R12,R12,#1 **@Αυξάνουμε το περιεχόμενο του R12 κατά 1 για να μεταβούμε στο byte της επόμενης διεύθυνσης όπου δείχνει ο R0 ($3^η$ τιμή του x)**

LDR R0,=Const **@Θέτουμε τον R0 να δείχνει στις διευθύνσεις του πίνακα Const**

LDRB R0,[R0,R12] **@Μεταφέρουμε στον R0 το byte της διεύθυνσης μνήμης όπου δείχνει ο R0**

MUL R9,R1,R1 **@Πολλαπλασιάζουμε τα περιεχόμενα των R1, R1 και το αποτέλεσμα το αποθηκεύουμε στον R9 (δηλαδή κάνουμε $x*x=x^2$)**

MUL R0,R9,R0 **@Πολλαπλασιάζουμε τα περιεχόμενα των R9, R0 και το αποτέλεσμα το αποθηκεύουμε στον R0 (δηλαδή κάνουμε $x*x*a_2 = x^2*a_2$)**

ADD R8,R8,R0 **@Προσθέτουμε τα περιεχόμενα των R8, R0 και το αποτέλεσμα ($a_2*x^2+a_1*x+a_0$) το αποθηκεύουμε στον R8**

ADD R12,R12,#1 **@Αυξάνουμε το περιεχόμενο του R12 κατά 1 για να μεταβούμε στο byte της επόμενης διεύθυνσης όπου δείχνει ο R0 ($4^η$ τιμή του x)**

```
Startpoint:

LDR R0,=Const @Θέτουμε τον R0 να δείχνει στις διευθύνσεις του πίνακα Const

LDRB R0,[R0,R12] @Μεταφέρουμε στον R0 το byte της διεύθυνσης μνήμης όπου δείχνει ο R0

MUL R9,R1,R9 @Πολλαπλασιάζουμε τα περιεχόμενα των R9, R1 και το αποτέλεσμα το αποθηκεύουμε στον R9 (δηλαδή κάνουμε x*x^2=x^3)

MUL R0,R9,R0 @Πολλαπλασιάζουμε τα περιεχόμενα των R9, R0 και το αποτέλεσμα το αποθηκεύουμε στον R0 (δηλαδή κάνουμε x*x^2*a3=x^3*a3)

ADD R8,R8,R0 @Προσθέτουμε τα περιεχόμενα των R8, R0 και το αποτέλεσμα (a3*x^3+a2*x^2+a1*x+a0) το αποθηκεύουμε στον R8

ADD R12,R12,#1

CMP R12,#6 @Ο βρόχος επαναλαμβάνβεται άλλες τρεις φορές για να υπολογίσουμε τα a4*x^4, a5*x^5, a6*x^6 και να υπολογίσουμε τη τελική τιμή του πολυωνύμου

BLE Startpoint


ADD R4,R4,#4

LDR R6,=Results

STR R8,[R6,R4] @Αποθηκεύουμε στον R6 που δείχνει στον πίνακα Results το περιεχόμενο του R8, δηλαδή τη τελική τιμή του πολυωνύμου

LDMIA R13!,{R0-R10}

MOV PC,LR


.data

Values:

.word 0x10

.word 0x50A

.word 0xCDCA

.word 0x80AB

Const:
```

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