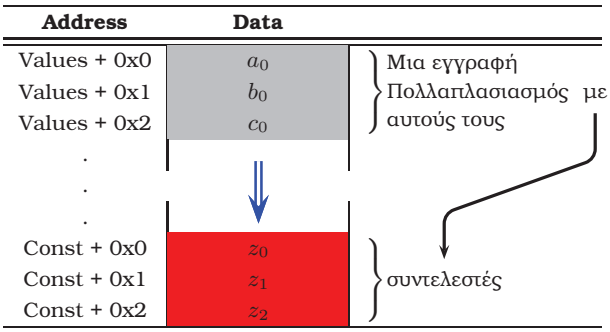
Εργασία 4

i. Υπολογισμός μαθηματικού τύπου

Δίδεται ο μαθηματικός τύπος $x = 5 * (a_i * z_0 + b_i * z_1 - c_i * z_2)/64$, όπου οι όροι a_i, b_i, c_i συμβολίζουν μεταβλητές, ενώ οι όροι z_i συμβολίζουν σταθερές. Οι μεταβλητές είναι ομαδοποιημένες στη μνήμη, με την ακόλουθη διάταξη:

Address	Variable
0x00	a_0
0x01	b_0
0x02	c_0
0x03	a_1
0x04	b_1
0x05	c_1
⋮	⋮

Παρατηρείστε πως οι μεταβλητές είναι τοποθετημένες διαδοχικά (a_i, b_i, c_i) και κάθε τριάδα ξεκινά ανά 3 θέσεις μνήμης (δηλαδή η μεταβλητή a_i θα βρίσκεται 3 θέσεις μετά από τη θέση του a_{i-1}). Επιπλέον, γνωρίζοντας τη θέση του a_i μπορεί να υπολογιστεί η θέση του b_i και c_i, διότι βρίσκονται 1 και 2 θέσεις μετά από τη θέση του a_i αντίστοιχα. Οι σταθερές z είναι τοποθετημένες με την ίδια διάταξη (δηλαδή z_0,z_1z_2) σε διαφορετική θέση μνήμης.



Καλείστε να υλοποιήσετε μια υπορουτίνα υπολογισμού του παραπάνω μαθηματικού τύπου, η οποία θα λαμβάνει στον καταχωρητή R0 τη διεύθυνση της μεταβλητής a_i (για τον υπολογισμό του τύπου με δεδομένα εισόδου την i-οστή τριάδα a_i, b_i, c_i). Πριν ολοκληρωθεί η υπορουτίνα, το αποτέλεσμα από τον υπολογισμό της i-οστής τριάδας θα πρέπει να τοποθετηθεί στον καταχωρητή R0. Ενδεικτικά βήματα του αλγορίθμου, από τη στιγμή που αρχίζει η εκτέλεση της υπορουτίνας, είναι:

1. Αποθήκευση του περιεχομένου των καταχωρητών που θα χρησιμοποιήσουμε στο σωρό.
2. Μεταφορά των δεδομένων από τη μνήμη στους καταχωρητές.

3. Εκτέλεση των πράξεων και αποθήκευση του αποτελέσματος στον R0.
4. Επαναφορά των αρχικών τιμών στους καταχωρητές. (μετά από αυτό το βήμα, οι τιμές των R1-R12 πρέπει να είναι ίδιες με αυτές που είχαν πριν ξεκινήσει η υπορουτίνα)

Για παράδειγμα ο παρακάτω κώδικας παρουσιάζει τα περιγραφόμενα βήματα:

Υπορουτίνα

```
1   .arm
2   .text
3   .global main
4
5   main:
6   STMDB R13!, {R1, R2}
7   MOV R0, =Values          @Αποθηκεύουμε τη διεύθυνση των δεδομένων
8   BL Subrtn                @Καλούμε την υπορουτίνα
9   LDMIA R13!, {R1, R2}
10
11  Subrtn:
12  STMDB R13!, {R1, R2}     @Αποθηκεύουμε στο σωρό το περιεχόμενο των καταχωρητών
13  LDRB R1, [R0, #0]        @Μεταφέρουμε στον R1 το byte της διεύθυνσης μνήμης όπου
                             δείχνει ο R0
14  LDRB R2, [R0, #1]        @Μεταφέρουμε στον R2 το byte της επόμενης διεύθυνσης μ-
                             νήμης απ'όπου δείχνει ο R0
15  MUL R1, R2, R1           @Τα πολλαπλασιάζουμε και αποθηκεύουμε το αποτέλεσμα στον
                             R1
16  STRB R1, [R0, #2]        @Μεταφέρουμε το αποτέλεσμα στη θέση μνήμης που βρίσκεται
                             2 θέσεις μετά από αυτή που δείχνει ο R0
17  LDMIA R13!, {R1, R2}     @Επανακτούμε το περιεχόμενο των καταχωρητών που είχαμε
                             σώσει
18  MOV PC,LR                @Επιστρέφουμε από την υπορουτίνα στο σημείο όπου κλήθηκε
19
20  .data
21  Values:
22  .byte 0x02, 0x03, 0x00
23
```

Μόλις ετοιμάσετε την υπορουτίνα αναπτύξτε ένα πρόγραμμα, όπου η βασική συνάρτηση main καλεί την υπορουτίνα και της περνά τα εξής δεδομένα (τοποθετώντας στον R0 τη διεύθυνση της μεταβλητής a_i):

Δεδομένα

```
1   .data
2   Values:
3   .byte 0x02, 0x02, 0x03
4   .byte 0x10, 0x07, 0x08
5   .byte 0x0B, 0x02, 0x0E
6   .byte 0x01, 0x0B, 0x08
```

```
7
8  Const:
9  .byte 0x04, 0x07, 0x05
```

Καταγράψτε στον ακόλουθο πίνακα τα 4 αποτελέσματα που παράγει η κλήση της υπορουτίνας.

Επανάληψη	Αποτέλεσμα
1	
2	
3	
4	

Υπόδειξη

Για λόγους απλότητας μπορείτε να καλέσετε την υπορουτίνα 4 φορές. Στον καταχωρητή R0 την πρώτη φορά θα αποθηκεύσετε την τιμή **=Values**, την δεύτερη φορά την τιμή **=Values** και θα προσθέσετε και 3 κλπ.

ii. Εύρεση μέγιστης τιμής σε πίνακα αποτελεσμάτων

Σε αυτό το μέρος καλείστε να μετατρέψετε τον κώδικά σας, έτσι ώστε να εντοπίζει ποιο από τα αποτελέσματα της υπορουτίνας είναι το μεγαλύτερο και ποιο σύνολο δεδομένων το παρήγαγε. Αν δηλαδή το 3ο σετ παράγει το αποτέλεσμα 0x35 θέλουμε να αποθηκευτεί στο 4ο byte μετά από την ετικέτα Const η τιμή 0x35 και στο 5ο byte το νούμερο του συνόλου, ξεκινώντας την αρίθμηση από το 0 (δηλαδή αν το πρώτο σετ δώσει το μεγαλύτερο αποτέλεσμα, στο 5ο byte θα πρέπει να αποθηκευτεί το 0).

iii. Υπολογισμός πολυωνύμου

Σκοπός του τελευταίου μέρους της εργασίας είναι ο υπολογισμός της τιμής ενός πολυωνύμου 6ου βαθμού, με γενικό τύπο $\sum_{i=0}^{6} a_i * x^i$. Οι σταθερές a_i παραμένουν ίδιες κάθε φορά (όπως οι σταθερές z_i στα προηγούμενα υποερωτήματα), αλλά η παράμετρος που εισάγεται στην υπορουτίνα μέσω του R0 είναι η τιμή του x (και όχι η διεύθυνσή του). Οι σταθεροί όροι είναι αποθηκευμένοι από τον a_0 προς τον a_6, με τον πρώτο να βρίσκεται στη διεύθυνση Const, ενώ ο τελευταίος στη διεύθυνση Const+6. Θέλει προσοχή το γεγονός πως το x δεν είναι byte, αλλά word. Υλοποιήστε την υπορουτίνα και εκτελέστε τη στα παρακάτω δεδομένα:

Δεδομένα

```
1   .data
2   Values:
3   .word 0x10
4   .word 0x50A
5   .word 0xCDCA
6   .word 0x80AB
7
8   Const:
9   .byte 0x04, 0x07, 0x05
10  .byte 0x20, 0x1A, 0x12, 0x06
```

Υπόδειξη

Ενώ φαίνεται απλό να πολλαπλασιαστεί το x με τον εαυτό του τόσες φορές όσες είναι η αντίστοιχη δύναμή του, και στη συνέχεια να το πολλαπλασιαστεί με τον σταθερό συντελεστή, κάτι τέτοιο δεν είναι πολύ αποδοτικό (για τον παραπάνω υπολογισμό θα χρειαστούμε 21 πολλαπλασιασμούς και 6 προσθέσεις). Μια εναλλακτική και πιο αποδοτική αντιμετώπιση είναι η επαναληπτική εκτέλεση των πράξεων : $b_{i-1} = b_i * x + a_{i-1}$ για i : 6...1, με $b_6 = a_6$. Ο όρος b_0 αποτελεί τη τιμή του πολυωνύμου (αν αναπτυχθεί ο επαναληπτικός τύπος μόνο σε όρους a και x, θα παρατηρείσετε ότι εμφανίζεται ο αρχικός τύπος του πολυωνύμου) και υπολογίζεται με μόνο 6 πολλαπλασιασμούς και 6 προσθέσεις.