i. Υπολογισμός μαθηματικού τύπου

```
0   .arm
1   .text
2   .global main
3
4   main:
5   STMDB R13!, {R2-R9}
6
7   LDR R0, =Values
8   LDR R1, =Const
9   LDR R10, =Results
10
11  BL Subrtn
12  ADD R0, R0, #3
13  ADD R10, R10, #1
14  BL Subrtn
15  ADD R0, R0, #3
16  ADD R10, R10, #1
17  BL Subrtn
18  ADD R0, R0, #3
19  ADD R10, R10, #1
20  BL Subrtn
21
22  LDMIA R13!, {R2-R9}
23
24
25  Subrtn:
26  STMDB R13!, {R2-R9}
27
28  LDRB R2, [R0]     @a0
29  LDRB R3, [R1]     @z0
30
31  MUL R4, R2, R3    @a0*z0
32
33  LDRB R2, [R0, #1] @b0
34  LDRB R3, [R1, #1] @z1
35
36  MUL R5, R2, R3    @a1*z1
37  ADD R4, R4, R5    @a0*z0 + b0*z1
38
39  LDRB R2, [R0, #2] @c0
40  LDRB R3, [R1, #2] @z2
41
42  MUL R5, R2, R3    @c0*z2
43
44  SUB R4, R4, R5    @a0*z0 + b0*z1 - c0*z2
45
46  MOV R8, #5
47  MUL R9, R4, R8
48  MOV R9, R9, LSR #6
49
50  STRB R9, [R10]
51
52  LDMIA R13!, {R2-R9}
53
54  MOV PC, LR
55
56
57  .data
58
59  Values:
60  .byte 0x02, 0x03, 0x04
61  .byte 0x10, 0x05, 0x06
62  .byte 0x0B, 0x02, 0x0D
63  .byte 0x01, 0x0C, 0x08
64
65  Const:
66  .byte 0x04, 0x07, 0x05
67
68  Results:
69  .byte 0x00, 0x00, 0x00, 0x00
```

Στους R0, R1, R10 αποθηκεύονται οι διευθύνσεις των θέσεων μνήμης που σηματοδοτούνται από τις ετικέτες Values, Const & Results. Στο Results θα αποθηκευτούν τα αποτελέσματα του προγράμματος.

Το πρόγραμμα τρέχει 4 φορές την υποροutίνα Subrtn, αυξάνοντας κάθε φορά την διεύθυνση της Values κατά 3 (προχωρώντας δλδ στην επόμενη τριάδα αριθμών) και την διεύθυνση της Results κατά 1 (για να αποθηκευτεί ο επόμενος αριθμός).

Στην Subrtn αποθηκεύονται στους καταχωρητές R2 και R3 τα περιεχόμενα των θέσεων μνήμης που έχουν αποθηκευτεί στους R0 και R1, δλδ η πρώτη τιμή των Values (a_0 και z_0) και το γινόμενό τους αποθηκεύεται στον R4. Θα χρησιμοποιήσουμε τον R4 ως τον καταχωρητή που θα αποθηκευτεί διαδοχικά το τελικό αποτέλεσμα, γιατί μπορούμε.

Στην συνέχεια αποθηκεύονται οι επόμενες τιμές b_0 και z_1 πάλι στους R2, R3 αυξάνοντας τις διευθύνσεις κατά 1 και αποθηκεύουμε το γινόμενό τους στον R5. Προσθέτουμε τα δύο προηγούμενα γινόμενα (του R4 & R5) στον R5. Με την ίδια διαδικασία δημιουργούμε και το τελευταίο γινόμενο c_0*z_2 που το αποθηκεύουμε στον R5 και το αφαιρούμε από το R4, δημιουργώντας στον καταχωρητή R4 τη παράσταση $a_0*z_0 + b_0*z_1 - c_0*z_2$.

Στον R8 αποθηκεύουμε τον αριθμό 5 ώστε να το πολλαπλασιάσουμε με την παραπάνω παράσταση. Το τελικό γινόμενο το αποθηκεύουμε στον καταχωρητή 9, τον οποίο ολισθαίνουμε λογικά προς τα δεξιά κατά 6 θέσεις το οποίο ισοδυναμεί με την διαίρεση ÷ 64.

Τέλος αποθηκεύουμε το αποτέλεσμα στην θέση μνήμης που είναι αποθηκευμένη στον R10 και επιστέφουμε στην main.

ii. Εύρεση μέγιστης τιμής σε πίνακα αποτελεσμάτων

```
0   .arm
1   .text
2   .global main
3
4   main:
5   STMDB R13!, {R0-R12}
6
7   LDR R0, =Values
8   LDR R1, =Const
9   LDR R10, =Results
10
11  MOV R11, #0     @μέγιστος
12  MOV R12, #0     @θέση
13
14  Loop:
15  BL Subrtn
16
17  LDRB R7, [R10]
18  CMP R7, R11
19  MOVHI R11, R7
20
21  STRHIB R11, [R1, #3]
22  STRHIB R12, [R1, #4]
23
24  ADD R0, R0, #3
25  ADD R10, R10, #1
26  ADD R12, R12, #1
27
28  CMP R12, #4
29  BNE Loop
30
31  STRB R2, [R1, #3]
32  STRB R12,[R1, #4]
33
34  LDMIA R13!, {R0-R12}
35
36  Subrtn:
37
38
39  LDRB R2, [R0]     @a0
40  LDRB R3, [R1]     @z0
41  MUL R4, R2, R3    @a0*z0
42  LDRB R2, [R0, #1] @b0
43  LDRB R3, [R1, #1] @z1
44  MUL R5, R2, R3    @a1*z1
45  ADD R4, R4, R5    @a0*z0 + b0*z1
46  LDRB R2, [R0, #2] @c0
47  LDRB R3, [R1, #2] @z2
48  MUL R5, R2, R3    @c0*z2
49  SUB R4, R4, R5    @a0*z0 + b0*z1 - c0*z2
50  MOV R8, #5
51  MUL R9, R4, R8
52  MOV R9, R9, LSR #6
53  STRB R9, [R10]
54
55
56  MOV PC, LR
57
58
59  .data
60  Values:
61  .byte 0x02, 0x03, 0x04
62  .byte 0x10, 0x05, 0x06
63  .byte 0x0B, 0x02, 0x0D
64  .byte 0x01, 0x0C, 0x08
65
66  Const:
67  .byte 0x04, 0x07, 0x05, 0x00, 0x00
68
69  Results:
70  .byte 0x00, 0x00, 0x00, 0x00
```

Προσθέτουμε τον R11 και R12 στους οποίους θα αποθηκεύονται ο τρέχων μέγιστος και η θέση του. Μετατρέπουμε την main έτσι ώστε να αποτελείται από μία επανάληψη στην οποία αρχικά τρέχει η υπορουτίνα Subrtn.

Στην συνέχεια ο καταχωρητής R7 διαβάζει από τη μνήμη το αποτέλεσμα της υπορουτίνας και το συγκρίνει με τον μέγιστο μέχρι στιγμής αριθμό. Αν ο τρέχων είναι μεγαλύτερος, αποθηκεύει αυτόν στον καταχωρητή R11.

Στην συνέχει αυξάνονται κατά 1 οι καταχωρητές R0, R10 όπως και στο i) αλλά και ο R12 που μετράει το πλήθος των τριάδων.

iii. Υπολογισμός πολυωνύμου

```
0    .arm
1    .text
2    .global main
3
4    main:
5
6    STMDB R13!, {R0, R2-R8}
7    LDR R10, =Values
8    LDR R11, =Results
9
10   Loop:
11
12   LDR R4, [R10], #4   @x
13   BL Subrtn
14
15   ADD R9, R9, #1
16   CMP R9, #4
17   BNE Loop
18
19   LDMIA R13!, {R0, R2-R8}
20
21
22       Subrtn:
23       STMDB R13, {R0, R2-R8}
24
25       LDR R0, =Const
26       MOV R2, #6
27
28       LDRB R1, [R0, #6]
29
30           SubLoop:
31           SUB R2, R2, #1
32           LDRB R3, [R0, R2]  @ai
33           MLA R1, R4, R1, R3   @bi
34
35           ADD R7, R7, #1
36           CMP R7, #6
37           BNE SubLoop
38
39       STR R1, [R11], #4
40
41       LDMIA R13!, {R0, R2-R8}
42       MOV PC, LR
43
44   .data
45
46   Values:
47   .word 0x10
48   .word 0x50A
49   .word 0xCDCA
50   .word 0x80AB
51
52   Const:
53   .byte 0x04, 0x07, 0x05
54   .byte 0x20, 0x1A, 0x12, 0x06
55   .byte 0x00, 0x00, 0x00
56
57   Results:
58   .word 0x00, 0x00, 0x00, 0x00
59
60
61
62
63
64
65
66
67
68
69
```

Στους καταχωρητές R10 & R11 αποθηκεύονται οι διευθύνσεις των Values και Results αντίστοιχα. Στον R4 αποθηκεύεται η πρώτη τιμή του x και προστίθεται 4 στην διεύθυνσή της μνήμης για την επόμενη τιμή.

Στην υπορουτίνα αποθηκεύεται στον R0 η διεύθυνση των Const και στον R2 η τιμή 6. Ο R2 λειτουργεί ως δείκτης για να προσπελάσουμε τα a_6 με a_0. Στον R1 αποθηκεύεται η πρώτη τιμή (a_6). Ο R1 δρα ως το b που αναδρομικά θα υπολογίζουμε.

Στην επανάληψη της υπορουτίνας ο δείκτης R2 μειώνεται κατά 1 για την επόμενη τιμή, και στον R3 αποθηκεύεται η επόμενη τιμή a_5. Στη συνέχεια υπολογίζεται η παράσταση $b_5 = R1 \cdot x + R3 = a_6 \cdot x + a_5$ και αποθηκεύεται στον καταχωρητή R1.

Την επόμενη φορά από τις 6 συνολικά (ώστε να προσπελαστούν όλες οι τιμές $a_{6\text{-}0}$) θα αποθηκευτεί στον R1 η τιμή $b_4 = b_5 \cdot x + a_4$ κ.ο.κ.

Στο τέλος αποθηκεύεται στο Results το αποτέλεσμα, δλδ ο καταχωρητής R1, και τρέχει η υπορουτίνα άλλες 3 φορές λόγω της επανάληψης στη main.