Συστήματα Μικροϋπολογιστών – 1η Σειρά Ασκήσεων

Κυριακόπουλος Γιώργος – el18153
Τζελέπης Σεραφείμ – el18849

1η Άσκηση:

Ακολουθεί η αποκωδικοποίηση του προγράμματος που δίνεται σε γλώσσα μηχανής:

0E 08	→		MVI C,08H
3A 00 20	→		LDA 2000H
17	→	SECOND	RAL
DA 0D 08	→		JC FIRST
0D	→		DCR C
C2 05 08	→		JNZ SECOND
79	→	FIRST	MOV A,C
2F	→		CMA
32 00 30	→		STA 3000H
CF	→		RST 1

Ουσιαστικά, θεωρώντας ότι ένα LED συμβολίζει το δυαδικό 1 όταν είναι ανοιχτό και το δυαδικό 0 όταν είναι σβηστό και ότι κάθε dip switch (διακόπτης) αντιστοιχεί σε έναν αριθμό από το 8 έως το 1 από αριστερά προς τα δεξιά, τότε παρατηρούμε πώς όταν τρέχουμε το πρόγραμμα ανάβουν τα αντίστοιχα LED που αποτελούν τη δυαδική αναπαράσταση του μεγαλύτερου (δεκαδικού) αριθμού των dip switches που είναι ανοιχτός (δηλαδή τον πρώτο από τα αριστερά - MSB - λόγω της φθίνουσας αντιστοιχίας τους).

Ακολουθεί το πρόγραμμα με τις συμβολικές του διευθύνσεις, υποθέτοντας ότι είναι φορτωμένο στη μνήμη με αρχή τη διεύθυνση 0800, καθώς και η επαλήθευση αυτών κατά το τρέξιμο του παραπάνω κώδικα στο πρόγραμμα προσομοίωση MicroLab Simulator:

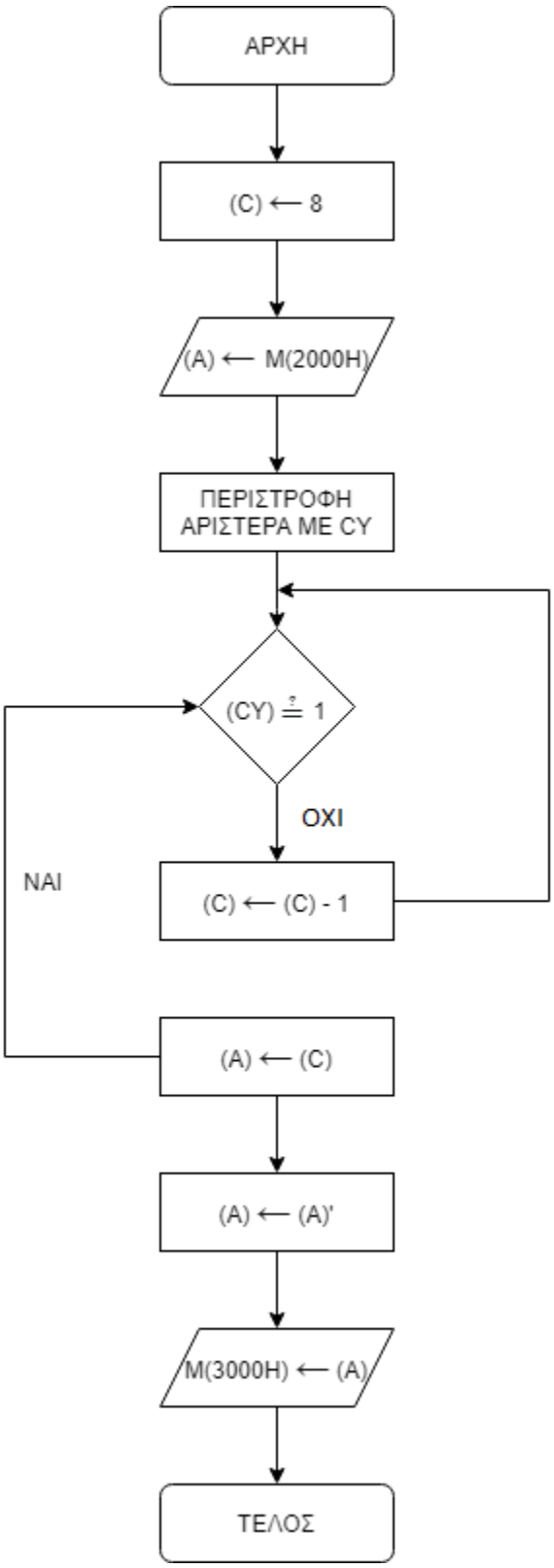
ΔΙΕΥΘΥΝΣΗ	**ΠΕΡΙΕΧΟΜΕΝΟ**	**ΕΤΙΚΕΤΑ**	**ASSEMBLY**
0800	0E – opcode		`MVI C,08H`
0801	08 – data		
0802	3A – opcode		`LDA 2000H`
0803	00 – data		
0804	20 – data		
0805	17 – opcode	`SECOND`	`RAL`
0806	DA – opcode		`JC FIRST(080D)`
0807	0D – data		
0808	08 – data		
0809	0D – opcode		`DCR C`
080A	C2 – opcode		`JNZ SECOND(0805)`
080B	05 – data		
080C	08 – data		
080D	79 – opcode	`FIRST`	`MOV A,C`
080E	2F – opcode		`CMA`
080F	32 – opcode		`STA 3000H`
0810	00 – data		
0811	30 – data		
0812	CF – opcode		`RST 1`

```
0800   0E     MVI C,08H
0801   08
0802   3A     LDA 2000H
0803   00
0804   20

SECOND:
0805   17     RAL
0806   DA     JC FIRST
0807   0D
0808   08
0809   0D     DCR C
080A   C2     JNZ SECOND
080B   05
080C   08

FIRST:
080D   79     MOV A,C
080E   2F     CMA
080F   32     STA 3000H
0810   00
0811   30
0812   CF     RST 1
```

Παρακάτω φαίνεται ο διαφοροποιημένος κώδικας για συνεχή λειτουργία του προγράμματος:

```
START:  MVI C,08H   ; Ο καταχωρητής C αρχικοποιείται με την τιμή 8.
        LDA 2000H   ; Φορτώνεται η τιμή των dip switches στον A.
SECOND: RAL         ; Αριστερή περιστροφή για να πάρουμε το MSB
                    ; στο flag κρατουμένου CY.
        JC FIRST    ; Αν CY = 1, τότε βρήκαμε τον αριστερότερο dip switch
                    ; που να είναι ON και κάνουμε άλμα στην εντολή
                    ; MOV A,C για προώθηση του C (που μετράει θέσεις μέχρι
                    ; να βρούμε το πρώτο LED = 1 από αριστερά) στον A και
                    ; στη συνέχεια στην έξοδο των LEDs.
        DCR C       ; Μείωση του C, καθώς θα προχωρήσουμε στα αριστερά
                    ; στην συνέχεια, εάν δεν ισχύει το CY = 1.
        JNZ SECOND  ; Άλμα στην αρχή της επαναληπτικής διαδικασίας.
FIRST:  MOV A,C     ; Προώθηση του μετρητή C στον A και έπειτα στην έξοδο.
        CMA         ; Αντιστροφή του A, ώστε τα LEDs αρνητικής λογικής να
                    ; ανάβουν για τους άσσους.
        STA 3000H   ; Προώθηση του A στην έξοδο που συνεπάγεται το
                    ; άναμμα των αντίστοιχων LEDs.
        JMP START   ; Άλμα στην αρχή του προγράμματος για να έχουμε
                    ; την ζητούμενη συνεχή λειτουργία.
        END         ; Τέλος του προγράμματος.
```

Τέλος, ακολουθεί το διάγραμμα ροής του πρώτου προγράμματος χωρίς τη συνεχή λειτουργία:



2η Άσκηση:

Ακολουθεί το πρόγραμμα που υλοποιεί το ζητούμενο της άσκησης με συνοπτικά σχόλια:

```
        IN 10H
        LXI B,01F4H ; Φορτώνουμε στο ζεύγος BC την καθυστέρηση (500).
        MVI D,FEH   ; Αρχικοποιούμε τον D με το hex του 11111110 που με
                    ; αρνητική λογική αντιστοιχεί στο LSB LED.
        MOV A,D     ; Φορτώνουμε τον ίδιο αριθμό και στον A.
        STA 3000H   ; Βγάζουμε στην έξοδο το πρώτο λαμπάκι (LSB).
START:  LDA 2000H   ; Φορτώνουμε στον A την είσοδο από τα dip switches.
        MOV E,A     ; Αποθηκεύουμε τον A στον E.
        RRC         ; Κάνουμε δεξιά ολίσθηση.
        JNC START   ; Έλεγχος εάν το LSB != 1, για να πάω στην START.
        CALL DELB   ; Καθυστέρηση για 0.5s (1ms·περιεχόμενο BC).
        MOV A,E     ; Επαναφορά του A μέσω του E.
        RLC         ; Ολίσθηση στα αριστερά με κρατούμενο.
        JC RIGHT    ; Αν το κρατούμενο (αρχικό MSB) είναι 1 κάνουμε
                    ; κύκλο προς τα δεξιά.
LEFT:   MOV A,D     ; Αρχικοποιούμε τον A μέσω του D.
        STA 3000H   ; Στέλνουμε τα δεδομένα του A στη θύρα εξόδου (LEDs).
        RLC         ; Αριστερή ολίσθηση με κρατούμενο.
        MOV D,A     ; Φορτώνουμε στον D τον A.
        JMP START   ; Πηγαίνουμε πάλι στην αρχή του προγράμματος.
RIGHT:  MOV A,D     ; Παρόμοια λογική με την υπορουτίνα LEFT.
        STA 3000H
        RRC
        MOV D,A
        JMP START
        END
```

3η Άσκηση:

Ακολουθεί το πρόγραμμα που υλοποιεί το ζητούμενο της άσκησης με συνοπτικά σχόλια:

```
        LXI B,01F4H ; Φορτώνουμε τον BC με την τιμή 500 = 01F4.
START:  LDA 2000H   ; Φορτώνουμε την είσοδο των switches στον A.
        CPI C8H     ; Έλεγχος εάν ο αριθμός είναι μεγαλύτερος του 199.
        JNC FLSM    ; Εάν είναι πήγαινε στην FLSM για τα MSB LEDs.
        CPI 64H     ; Έλεγχος εάν ο αριθμός είναι μεγαλύτερος του 99.
        JNC FLSL    ; Εάν είναι πήγαινε στην FLSL για τα LSB LEDs.
        MVI E,FFH   ; Αρχικοποιούμε το μετρητή δεκάδων E με -1.
        JMP DEC     ; Πήγαινε στον υπολογισμό των δεκάδων/μονάδων.
DEC:    INR E
        SUI 0AH     ; Αφαίρεσε 10 από τον A.
        JNC DEC     ; Εάν είναι θετικός, επανάληψη της αφαίρεσης.
        ADI 0AH     ; Εάν είναι αρνητικός, διόρθωσε το υπόλοιπο.
        MOV D,A     ; Αποθήκευσε τις μονάδες στον D.
        MOV A,E     ; Φέρε τον μετρητή δεκάδων στον A.
        RLC         ; Κάνε 4 shift αριστερά, άρα πολλαπλασιασμό με το 8
        RLC         ; για να ετοιμάσεις την έξοδο των LEDs.
        RLC
        RLC
        ADD D       ; Πρόσθεσε τις μονάδες στον A για την έξοδο.
        CMA         ; Συμπλήρωμα του A, λόγω των αρνητικής λογικής LEDs.
        STA 3000H   ; Στέλνουμε στην έξοδο τον A.
        JMP START   ; Επανάληψη από την αρχή.
FLSM:   MVI A,0FH   ; Αρνητική λογική, 00001111 = F για τα MSB LEDs.
        STA 3000H
        CALL DELB   ; Καθυστέρηση, ώστε να είναι ορατό το αναβόσβημα.
        MVI A,FFH   ; Αρνητική λογική, 11111111 = FF για σβηστά LEDs.
        STA 3000H
        CALL DELB   ; Καθυστέρηση, ώστε να είναι ορατό το αναβόσβημα.
        JMP START   ; Επανάληψη από την αρχή.
FLSL:   MVI A,F0H   ; Αρνητική λογική, 11110000 = F0 για τα LSB LEDs.
        STA 3000H
        CALL DELB   ; Καθυστέρηση, ώστε να είναι ορατό το αναβόσβημα.
        MVI A,FFH   ; Αρνητική λογική, 11111111 = FF για σβηστά LEDs.
        STA 3000H
        CALL DELB   ; Καθυστέρηση, ώστε να είναι ορατό το αναβόσβημα.
        JMP START   ; Επανάληψη από την αρχή.
        END
```

4η Άσκηση:

Για τις 4 τεχνολογίες που περιγράφονται στην εκφώνηση έχουμε τις παρακάτω πληροφορίες:

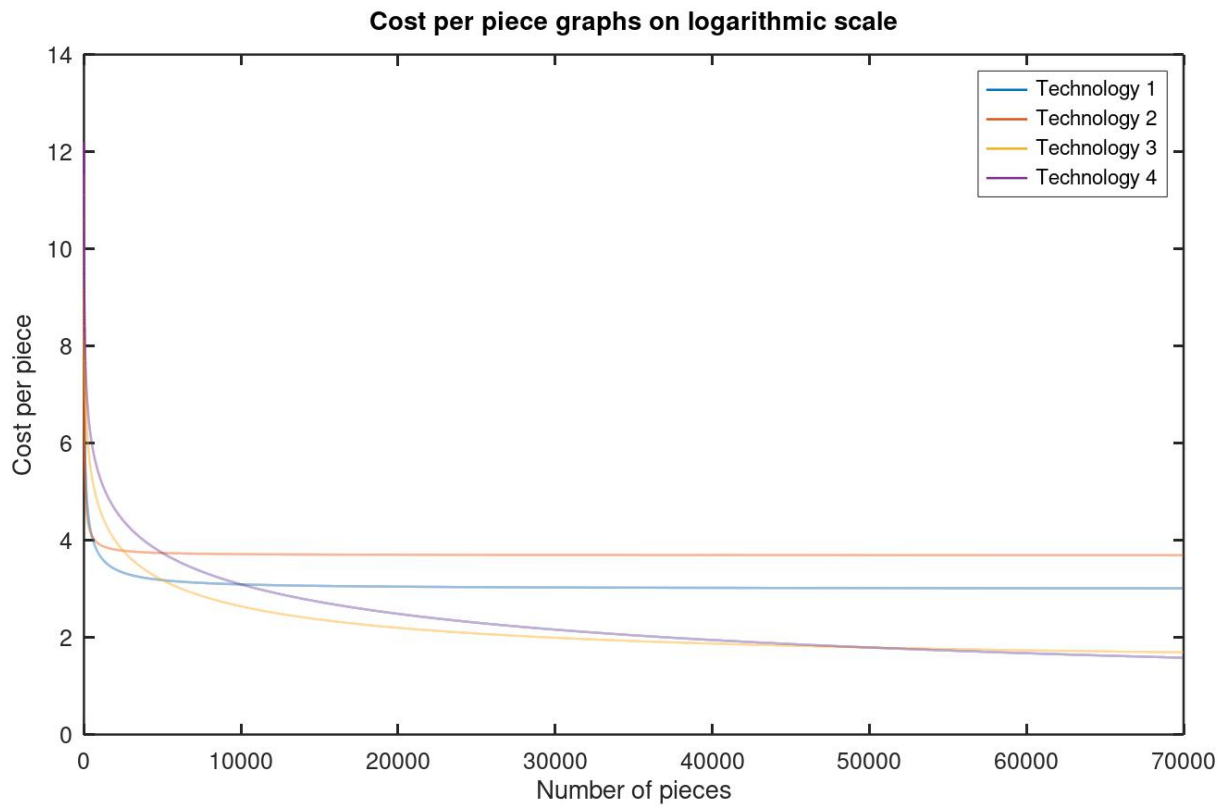
Τεχνολογία 1: Αρχικό κόστος σχεδίασης: 20.000€
Κόστος Ι.Σ.: 10€ ανά τεμάχιο
Κόστος πλακέτας και συναρμολόγησης: 10€ ανά τεμάχιο
Καμπύλη κόστους: K1(x) = 20x + 20.000
Καμπύλη κόστους ανά τεμάχιο: f1(x) = 20.000/x + 20

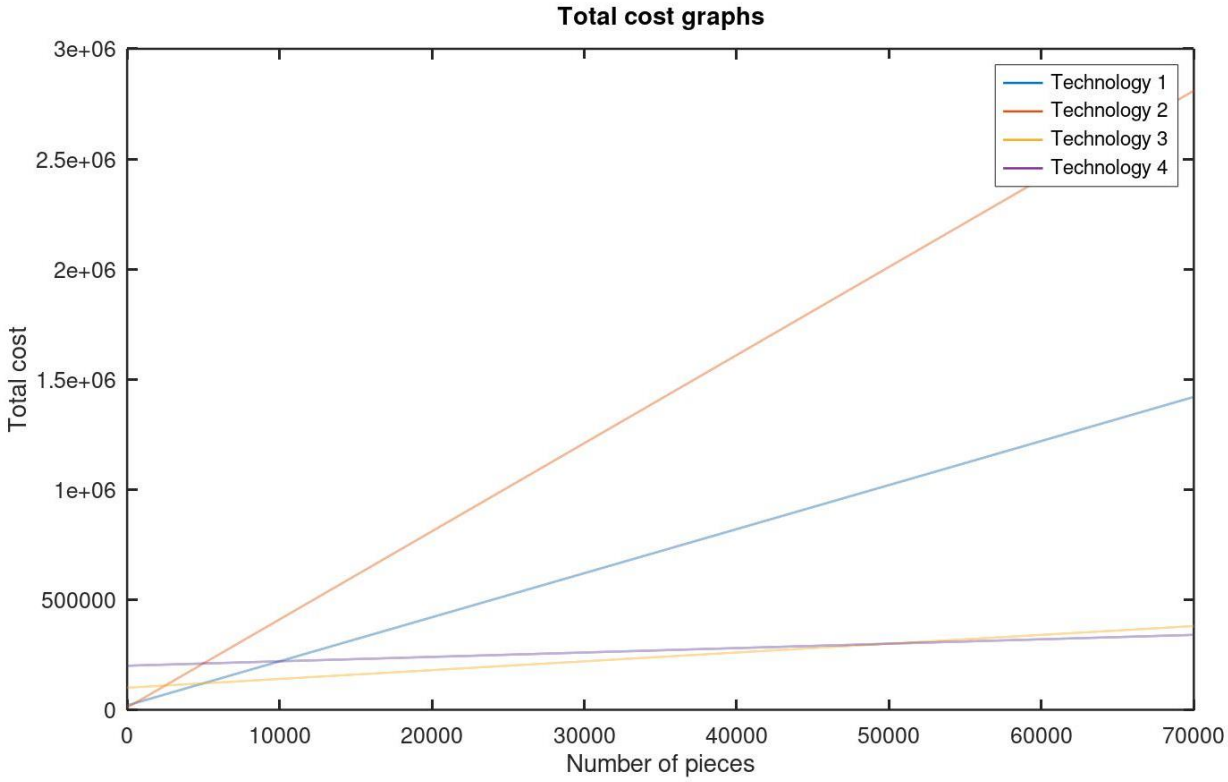
Τεχνολογία 2: Αρχικό κόστος σχεδίασης: 10.000€
Κόστος Ι.Σ.: 30€ ανά τεμάχιο
Κόστος πλακέτας και συναρμολόγησης: 10€ ανά τεμάχιο
Καμπύλης κόστους: K2(x) = 40x + 10.000
Καμπύλη κόστους ανά τεμάχιο: f2(x) = 10.000/x + 40

Τεχνολογία 3: Αρχικό κόστος σχεδίασης: 100.000€
Κόστος Ι.Σ.: 2€ ανά τεμάχιο
Κόστος πλακέτας και συναρμολόγησης: 2€ ανά τεμάχιο
Καμπύλης κόστους: K3(x) = 4x + 100.000
Καμπύλη κόστους ανά τεμάχιο: f3(x) = 100.000/x + 4

Τεχνολογία 4: Αρχικό κόστος σχεδίασης: 200.000€
Κόστος Ι.Σ.: 1€ ανά τεμάχιο
Κόστος πλακέτας και συναρμολόγησης: 1€ ανά τεμάχιο
Καμπύλης κόστους: K4(x) = 2x + 200.000
Καμπύλη κόστους ανά τεμάχιο: f4(x) = 200.000/x + 2

Επομένως, μπορούμε να δημιουργήσουμε τα διαγράμματα κόστους ανά τεμάχιο και συνολικού κόστους για τις 4 αυτές τεχνολογίες, όπως παρουσιάζονται και παρακάτω:


Cost per piece graphs on logarithmic scale
Technology 1
Technology 2
Technology 3
Technology 4
Cost per piece
Number of pieces
0
2
4
6
8
10
12
14
0
10000
20000
30000
40000
50000
60000
70000


Total cost graphs
Technology 1
Technology 2
Technology 3
Technology 4
Total cost
Number of pieces
0
500000
1e+06
1.5e+06
2e+06
2.5e+06
3e+06
0
10000
20000
30000
40000
50000
60000
70000

Οι 4 περιοχές αριθμού τεμαχίων που είναι συμφερότερες για την κάθε τεχνολογία προκύπτουν από τα σημεία τομής των 4 καμπυλών:

Σύμφωνα με τα διαγράμματα, για ένα μικρό αριθμό τεμαχίων η Τεχνολογία 2 είναι η συμφερότερη αρχικά, μέχρι να την προσπεράσει η Τεχνολογία 1:

$K2(x) = K1(x) \Rightarrow 40x + 10.000 = 20x + 20.000 \Rightarrow x = 500$ τεμάχια.

Άρα περιοχή 1 = [0, 500) : συμφέρει η Τεχνολογία 2.

Στη συνέχεια, για ένα μεγαλύτερο αριθμό τεμαχίων, φαίνεται να συμφέρει η Τεχνολογία 1, μέχρι το σημείο όπου η Τεχνολογία 3 γίνεται συμφερότερη:

$K1(x) = K3(x) \Rightarrow 20x + 20.000 = 4x + 100.000 \Rightarrow x = 5.000$ τεμάχια.

Άρα περιοχή 2 = [500, 5.000): συμφέρει η Τεχνολογία 1.

Έπειτα, έρχεται η σειρά της Τεχνολογίας 3, μέχρι, πολύ αργότερα, να την συναντήσει η Τεχνολογία 4:

$K3(x) = K4(x) \Rightarrow 4x + 100.000 = 2x + 200.000 \Rightarrow x = 50.000$ τεμάχια.

Άρα περιοχή 3 = [5.000, 50.000): συμφέρει η Τεχνολογία 3.

Και επομένως τελικά έχουμε την Τεχνολογία 4 να είναι η συμφερότερη από 50.000 τεμάχια και πάνω:

Άρα περιοχή 4 = [50.000, $+\infty$): συμφέρει η Τεχνολογία 4.

Ψάχνουμε την τιμή κόστους I.C. ανά τεμάχιο (I.C.P.) για την Τεχνολογία 2, που θα την καταστήσει συμφερότερη της Τεχνολογίας 1, ώστε να εξαφανιστεί πλήρως η επιλογή της 1:

$K1(x) > K2'(x) \Rightarrow 20x + 20.000 > (10+I.C.P.)x + 10.000 \Rightarrow (10 - I.C.P.)x > -10.000 \Rightarrow$
$(I.C.P - 10)x < 10.000$

Επειδή το x αποτελεί αριθμό γνήσια θετικό, πρέπει ο συντελεστής του να είναι αρνητικός, ώστε να ισχύει πάντα η παραπάνω ισότητα. Άρα, έχουμε:

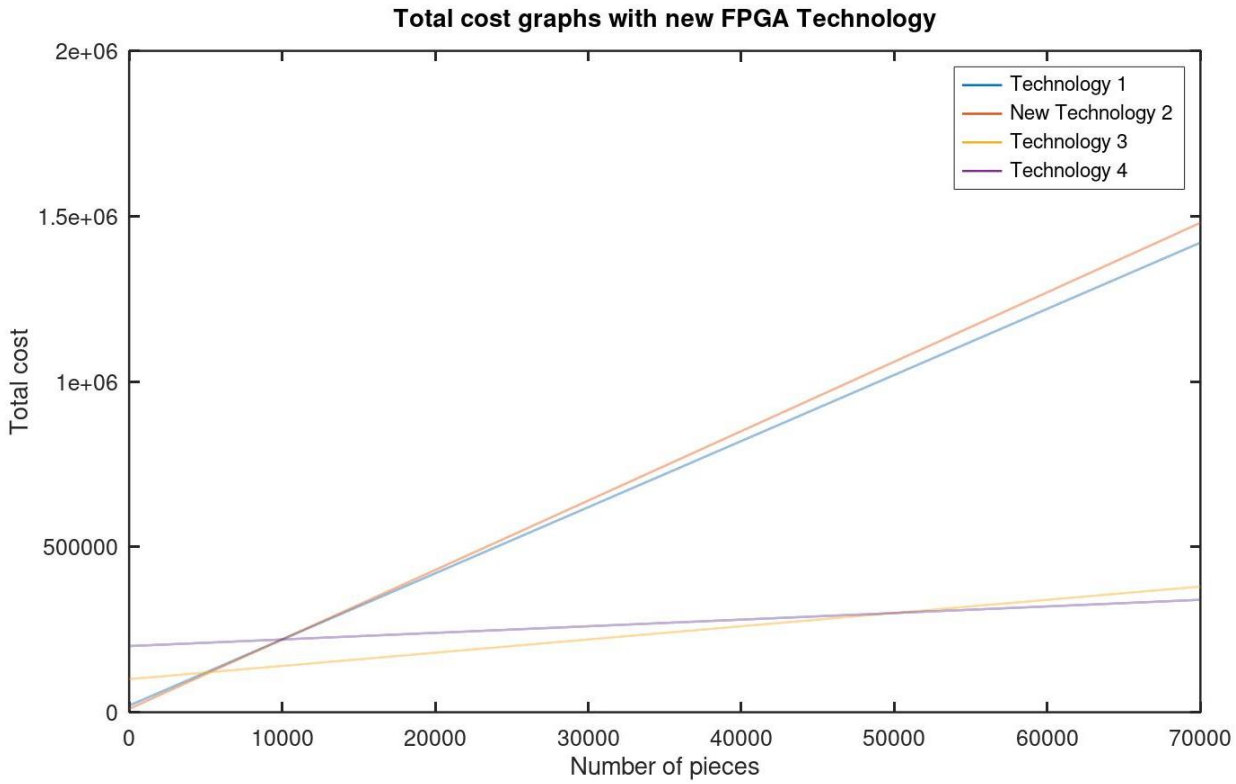
$I.C.P. - 10 < 0 \Rightarrow I.C.P. < 10$

Δηλαδή, χρειάζεται η τιμή κόστους ανά τεμάχιο των I.C. στην Τεχνολογία 2 των FPGAs να είναι μικρότερη των 10€, ώστε να εξαφανιστεί πλήρως η επιλογή της Τεχνολογίας 1.

Ωστόσο, στη συγκεκριμένη περίπτωση μας ενδιαφέρει μόνο η περιοχή [500, 5.000) στην οποία η Τεχνολογία 1 παίρνει το προβάδισμα από την Τεχνολογία 2 και στη συνέχεια η Τεχνολογία 3 γίνεται πιο συμφέρουσα. Άρα έχουμε την εξίσωση (I.C.P - 10)x < 10.000 και την 500 <= x < 5.000, οπότε μελετώντας τις οριακές περιπτώσεις με x = 500 και x = 5.000 παίρνουμε ότι:

I.C.P. < 30 και I.C.P. < 12, τις οποίες συναληθεύουμε και οδηγούμαστε στην I.C.P. < 12.

Επομένως, για τιμή κόστους ανά τεμάχιο των I.C. μικρότερη των 12€ (π.χ. 11€), η επιλογή της Τεχνολογίας 1 εξαφανίζεται από την περιοχή [500, 5.000) και επομένως και συνολικά, αφού αργότερα έχουμε άλλες πιο συμφέρουσες τεχνολογίες. Αυτό είναι φανερό και στο παρακάτω διάγραμμα, όπου στην περιοχή [500, 5.000) βλέπουμε ότι η Νέα Τεχνολογία 2 με τα I.C. των 11€ είναι συμφέρουσα σε σχέση με την Τεχνολογία 1, ή οποία πλέον δεν αποτελεί τη συμφέρουσα επιλογή για καμία περιοχή.



5η Άσκηση:

i) F1 = A(BC + D) + B'C'D'

```
module circuit_A (A, B, C, D, F1);
    input A, B, C, D;
    output F1;
    wire notB, notC, w, x, y, z;
    not(notC, C);
    not(notB, B);
    and(x, notC, notB, D);
    and(w, C, B);
    or(z, w, D);
    and(y, z, A);
    or(F1, x, y);
endmodule
```

F2(A, B, C, D) = Σ (0, 2, 3, 5, 7, 9, 10, 11, 13, 14)

```
module circuit_B (A, B, C, D, F2);
    input A, B, C, D;
    output F2;
    wire notB, notC, notD, notA, a, b, c, d, e, f, g, h, i, k;
    not(notC, C);
    not(notB, B);
    not(notA, A);
    not(notD, D);
    and(a, notA, notB, notC, notD);     //0
    and(b, notA, notB, C, notD);        //2
    and(c, notA, notB, C, D);           //3
    and(d, notA, B, notC, D);           //5
    and(e, notA, B, C, D);              //7
    and(f, A, notB, notC, D);           //9
    and(g, A, notB, C, notD);           //10
    and(h, A, notB, C, D);              //11
    and(i, A, B, notC, D);              //13
    and(k, A, B, C, notD);              //14
    or(F2, a, b, c, d, e, f, g, h, i, k);
endmodule
```

F3 = ABC + (A + BC)D + (B + C)DE

```
module circuit_C (A, B, C, D, E, F3);
    input A, B, C, D ,E;
    output F3;
    wire a, b, c, d, e, f, g;
    and(a, A, B, C);
    and(b, B, C);
    or(c, A, b);
    and(d, D, c);
    or(e, B, C);
    and(f, D, E);
    and(g, f ,e);
    or(F3, a, d, g);
endmodule
```

F4 = A(B + CD + E) + BCDE

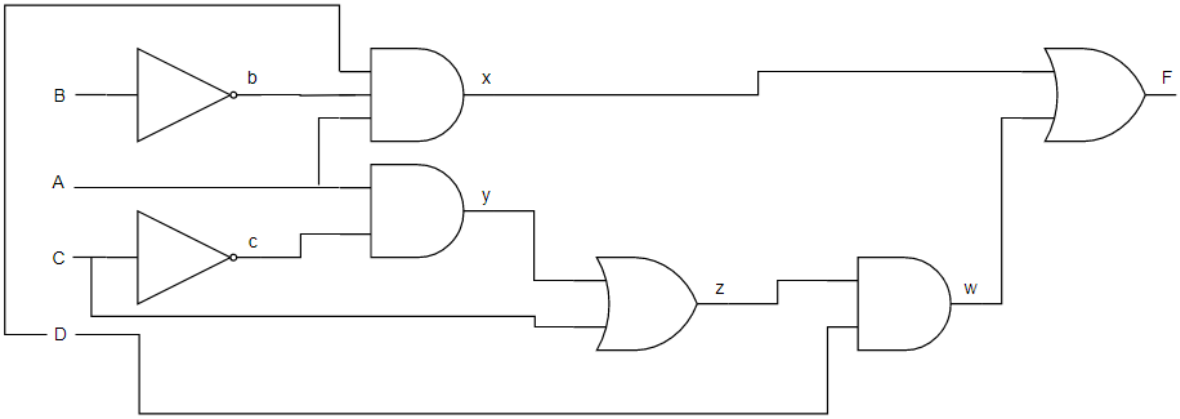
```
module circuit_D (A, B, C, D, E, F4);
    input A, B, C, D, E;
    output F4;
    wire a, b, c, d;
    and(a, C, D);
    or(b, B, a, E);
    and(c, A, b);
    and(d, B, C, D, E);
    or(F4, c, d);
endmodule
```

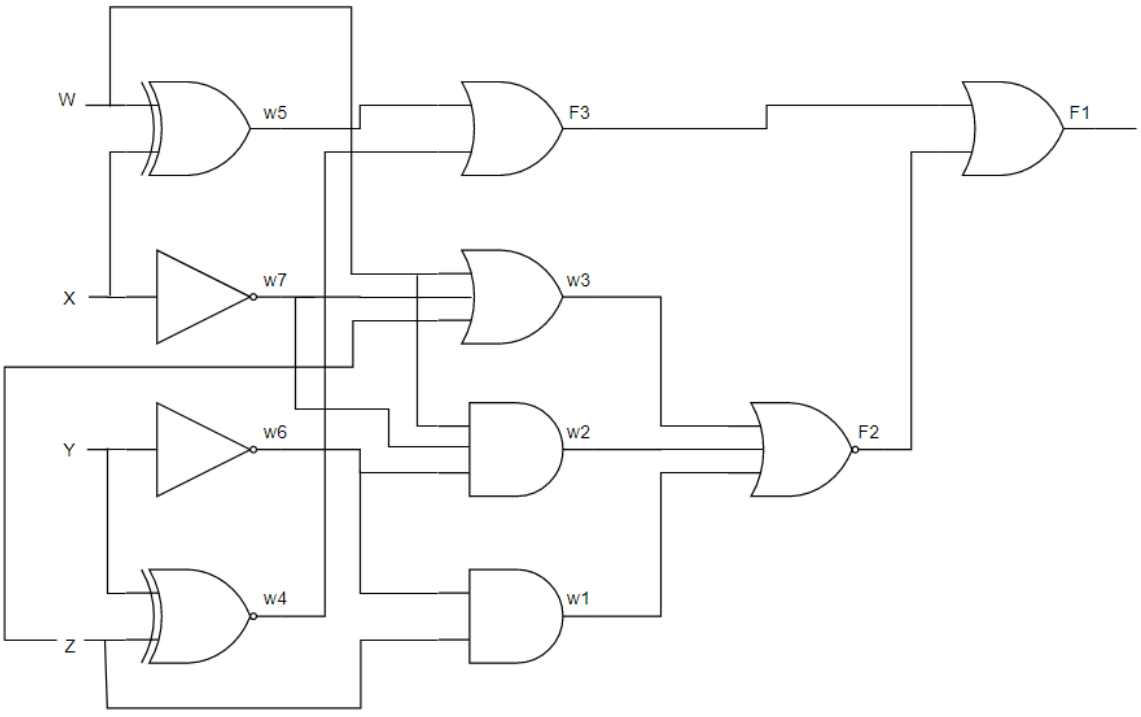
ii)

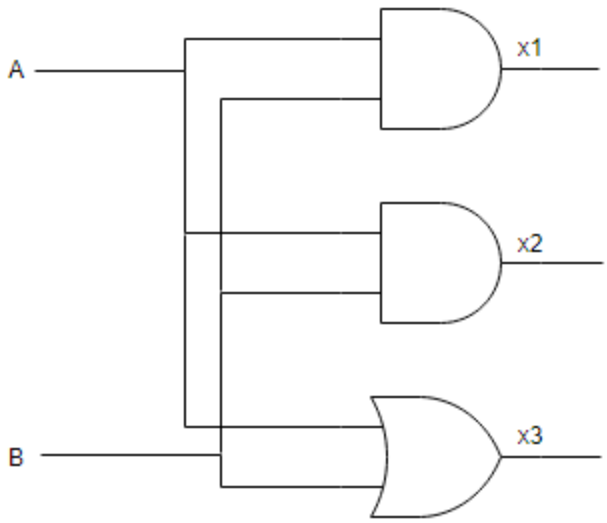
```
module dataFlow(A, B, C, D, E, F1, F2, F3, F4);
    input A, B, C, D, E;
    output F1, F2, F3, F4;
    assign
        F1 = (A & ((B & C) | D)) | (~A & ~B & D) ,

        F2 = (~A & ~B & ~C & ~D) | (~A & ~B & C & ~D) |
             (~A & ~B & C & D) | (~A & B & ~C & D) |
             (~A & B & C & D) | (A & ~B & ~C & D) |
             (A & ~B & C & ~D) | (A & ~B & C & D) |
             (A & B & ~C & D) | (A & B & C & ~D),

        F3 = (A & B & C) | ((A | (B & C)) & D) | ((B | C) & D & E),

        F4 = (A & (B | (C & D) | E)) | (B & C & D & E);
endmodule
```

6η Άσκηση:

i)






A
B
x1
x2
x3

ii)

```
module half_adder(S, C, x, y);
    input x, y;
    output S, C;
    xor(S, x, y);
    and(C, x, y);
endmodule

module full_adder(S, C, x, y, z);
    input x, y, z;
    output  S, C;
    wire S1, C1, C2;
    half_adder HA1(S1, C1, x, y);
    half_adder HA2(S, C2, S1, z);
    or G1(C, C2, C1);
endmodule

module _4_bit_add_sub(S, C, V, A, B, C0);
    input C0;
    input [3:0] A, B;
    output C, V;
    output [3:0] S;
    wire C1, C2, C3;
    wire w0, w1, w2, w3;
    xor G1(w0, B[0], C0);
    xor G2(w1, B[1], C0);
    xor G3(w2, B[2], C0);
    xor G4(w3, B[3], C0);
    full_adder FA0 (S[0], C1, w0, A[0], C0);
    full_adder FA1 (S[1], C2, w1, A[1], C1);
    full_adder FA2 (S[2], C3, w2, A[2], C2);
    full_adder FA3 (S[3], C, w3, A[3], C3);
    xor G5(V, C, C3);
endmodule
```

iii)

```
module add_sub (S, C, A, B, C0);
    output [3: 0] S;
    output C;
    input [3: 0] A, B;
    input C0;
    assign {C, S} = C0 ? A - B : A + B ;
endmodule
```

7η Άσκηση:

i)

```
module Mealy_Model(y, x, clock, reset);
    output [1:0] y;
    input x, clock, reset;
    reg [1:0] state;
    parameter a = 2'b00, b = 2'b01, c = 2'b10, d = 2'b11;
    always @(posedge clock, negedge reset)
    if (reset == 0) state <= a;
    else case(state)
    a: if(x) state <= a; else state <= d;
    b: if(x) state <= a; else state <= c;
    c: if(x) state <= b; else state <= d;
    d: if(x) state <= d; else state <= c;
    endcase
    assign y = (state[1] & state[0] & x) | (~state[0] & ~x) |
               (~state[1] & ~x);
endmodule
```

ii)

```
module Moore_Model(y, x, clock, reset);
    output [1:0] y;
    input x, clock, reset;
    reg [1:0] state;
    parameter a = 2'b00, b = 2'b01, c = 2'b10, d = 2'b11;
    always @(posedge clock, negedge reset)
    if (reset == 0) state <= a;
    else case(state)
    a: if(x) state <= a; else state <= d;
    b: if(x) state <= a; else state <= c;
    c: if(x) state <= d; else state <= b;
    d: if(x) state <= d; else state <= c;
    endcase
    assign y = (state[1] & ~state[0]) | (~state[1] & state[0]);
endmodule
```