Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

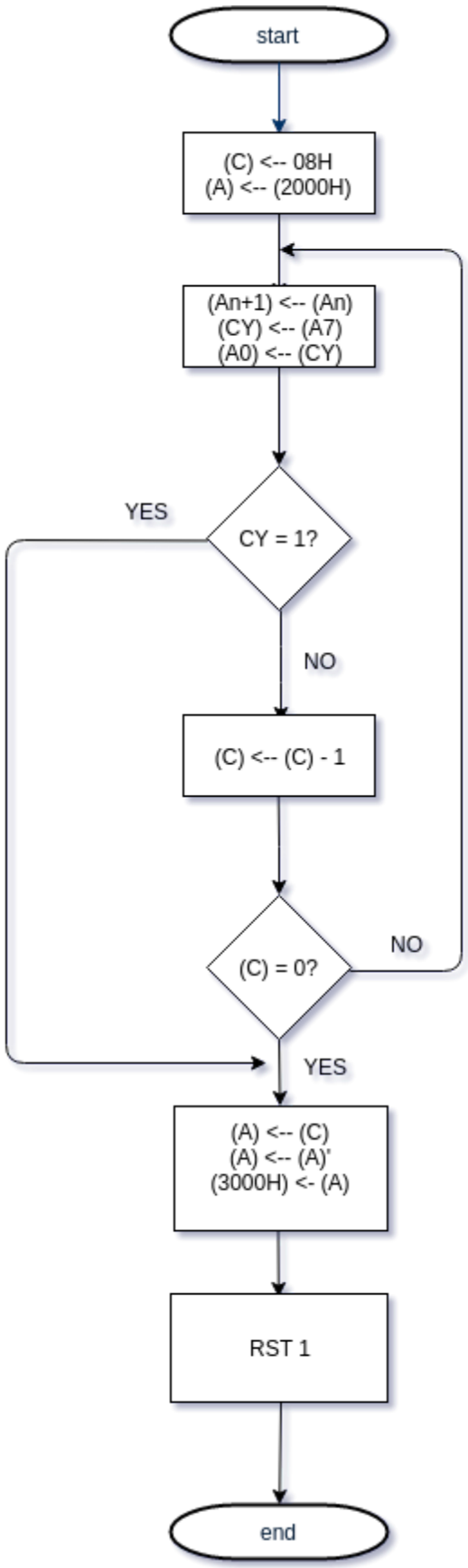
Συστήματα Μικρουπολογιστών

Σειρά Ασκήσεων 1	Θανάσουλας Γρηγόριος | ΑΜ: 03114131 Μόνου Σταματίνα | ΑΜ: 03114077

Άσκηση 1

Χρησιμοποιώντας τον πίνακα του παραρτήματος 2, προέκυψε ο παρακάτω πίνακας με την αντιστοιχία των δοθέντων εντολών, και ακολούθως δημιουργήσαμε το διάγραμμα ροής του προγράμματος.

Διεύθυνση	Περιεχόμενο	Ετικέτα	Εντολή
0800	**OE**		MVI C, 08H
0801	08		
0802	**3A**		LDA 2000H
0803	00		
0804	20		
0805	**17**	LOOP	RAL
0806	**DA**		*JC END1*
0807	0D		
0808	08		
0809	**0D**		DCR C
080A	**C2**		JNZ LOOP
080B	05		
080C	08		
080D	**79**	END1	MOV A, C
080F	**2F**		CMA
0810	**32**		STA 3000H
0811	00		
0812	30		
0813	**CF**		RST 1


start
(C) <-- 08H
(A) <-- (2000H)
(An+1) <-- (An)
(CY) <-- (A7)
(A0) <-- (CY)
CY = 1?
YES
NO
(C) <-- (C) - 1
(C) = 0?
NO
YES
(A) <-- (C)
(A) <-- (A)'
(3000H) <- (A)
RST 1
end

Με βάση το παραπάνω διάγραμμα ερμηνεύσαμε τη λειτουργία που επιτελεί το πρόγραμμα. Έτσι, λοιπόν:

1. Το πρόγραμμα αρχικοποιεί τον καταχωρτή C = 8.
2. Διαβάζει την είσοδο της θύρας στη διεύθυνση 2000H και την την εκχωρεί στον καταχωρητή A.
3. Περιστρέφει κυκλικά προς τα αριστερά τα bits του A. Θέτει ως κρατούμενο το MSB του A.
4. Αν το κρατούμενο, δηλ το A7 είναι 1 βγαίνει εκτός LOOP, και εκχωρεί στον A την τιμή του C. Η τιμή του C αντιπροσωπεύει τη θέση του bit του A όπου βρέθηκε ο πρώτος άσσος. Ο αριθμός 8 αντιστοιχεί στο MSB του αρχικού αριθμού στην είσοδο, ενώ το 1 στο LSB.
5. Αν το A7 δεν ήταν 1, άρα ήταν 0, και δεν έχουμε διανύσει 8 επαναλήψεις, τότε επαναλαμβάνεται η διαδικασία για τον αριθμό που έχει περιστραφεί προς τα αριστερά.
6. **Με τη διαδικασία αυτή βρίσκουμε τη θέση όπου εμφανίζεται ο πρώτος άσσος ξεκινώντας από αριστερά, και το C αποθηκεύεται στον A.**
7. *Ο A αντιστρέφεται ώστε να υπάρχει συμβατότητα με την ανεστραμμένη λογική των LEDs και δίνεται ως έξοδος στη θύρα 3000H.*

Προκειμένου να εκτελείτε συνεχόμενα το κομμάτι του κώδικα αυτού, προσθέτουμε την ετικέτα START στην αρχή του κώδικα και στο τέλος αντικαθιστούμε την εντολή RST με την JMP START.

Ο σχετικός κώδικας μας βρίσκεται στα αρχεία *askisi1_seira1.8085* και *askisi1_seira1_nonstop.8085.*

```
START:      MVI C,08H
            LDA 2000H
LOOP1:      RAL
            JC END1
            DCR C
            JNZ LOOP1
END1:       MOV A,C
            CMA
            STA 3000H
            JMP START
END
```

Άσκηση 2

```
IN 10H
LXI H,3000H     ;Φόρτωση της διεύθυνσης εισόδου στους L, H
```

```
        MVI M,FEH       ;Ενεργοποίηση του LSB LED
        MVI E,FEH       ;Αποθήκευση κατάστασης LED
        LXI B,03E8H     ;Ορισμός χρονοκαθυστέρησης 1000ms
READ1:          LDA 2000H       ;Διάβασμα εισόδου
        MOV D,A         ;Δημιοργία αντιγράφου του A στον D
        ANI 01H         ;Δημιουργία μάσκας για το τελευταίο ψηφίο
        CPI 01H         ;Έλεχγχος LSB ψηφίου -αν είναι 0 OFF κάνουμε pause
        JNZ READ1       ;Αν ΔΕΝ είναι 1 το τελικό ψηφίο, γύρνα πίσω
        MOV A,D
        ANI 80H         ;Δημιορυργία μάσκας για το MSB
        CPI 80H         ;Αν το LSB = 1 δεξιά, αλλιώς αριστερά
        JZ RIGHT
LEFT:   MOV A,E         ;Διάβασμα κατάστασης LEDS
        RLC             ;Περιστροφή πρός τα αριστερά
        STA 3000H
        MOV E,A
        CALL DELB
        JMP READ1
RIGHT: MOV A,E          ;Διάβασμα κατάστασης LEDS
        RRC             ;Περιστροφή πρός τα δεξιά
        STA 3000H
        MOV E,A
        CALL DELB
        JMP READ1
END
```

Άσκηση 3

```
START:LDA 2000H     ;Διάβασμα Εισόδου
        CPI 64H         ;Είναι μεγαλύτερος του 99? (A-99)
        JNC ERRR        ;Αν ναι τέλος (A>99)
        MVI B,FFH       ;Αλλιώς A ? 99
DECA:  INR B
        SUI 0AH         ;Αλλεπάλληλες αφαιρέσεις του 10
        JNC DECA        ;Αν είναι θετικός συνέχισε
        ADI 0AH         ;Διόρθωση του αρνητικού υπολοίπου

        MOV D,A         ;Αποθήκευση των μονάδων στο D
        MOV A,B         ;Εκχώρυση των δεκάδων στο A
        RLC             ;Μετακίνηση 4 θέσεις αριστερά για να χωρέσουμε τα LSB
        RLC
        RLC
        RLC
        ORA D           ;Concantenate Δεκάδες--Μονάδες
        CMA             ;Αντιστροφή λογική, αρα το συμπλήρωμα
        STA 3000H
        JMP START
ERRR:  LXI B,07D0H      ;Ορισμός χρονοκαθυστέρησης 1000ms
        MVI A,0FH       ;Ενεργοποίηση των 4MSB
```

```
        STA 3000H
        CALL DELB
        MVI A,F0H       ;Ενεργοποίηση των 4MSB
        STA 3000H
        CALL DELB
        JMP START
END
```

Άσκηση 4

Υποερώτημα Α

Τα δεδομένα του κόστους φαίνονται στον παρακάτω πίνακα.

Επιλογή	Κόστος Υλικού / τμχ	Κόστος Κατασκευής / τμχ	Αρχικό Κόστος Σχεδίασης
1	10 €	10 €	20.000 €
2	30 €	10 €	10.000 €
3	2 €	2 €	100.000 €
4	1 €	1 €	200.000 €

Οι συναρτήσεις συνολικού κόστους **ανά τεμάχιο** είναι οι παρακάτω:

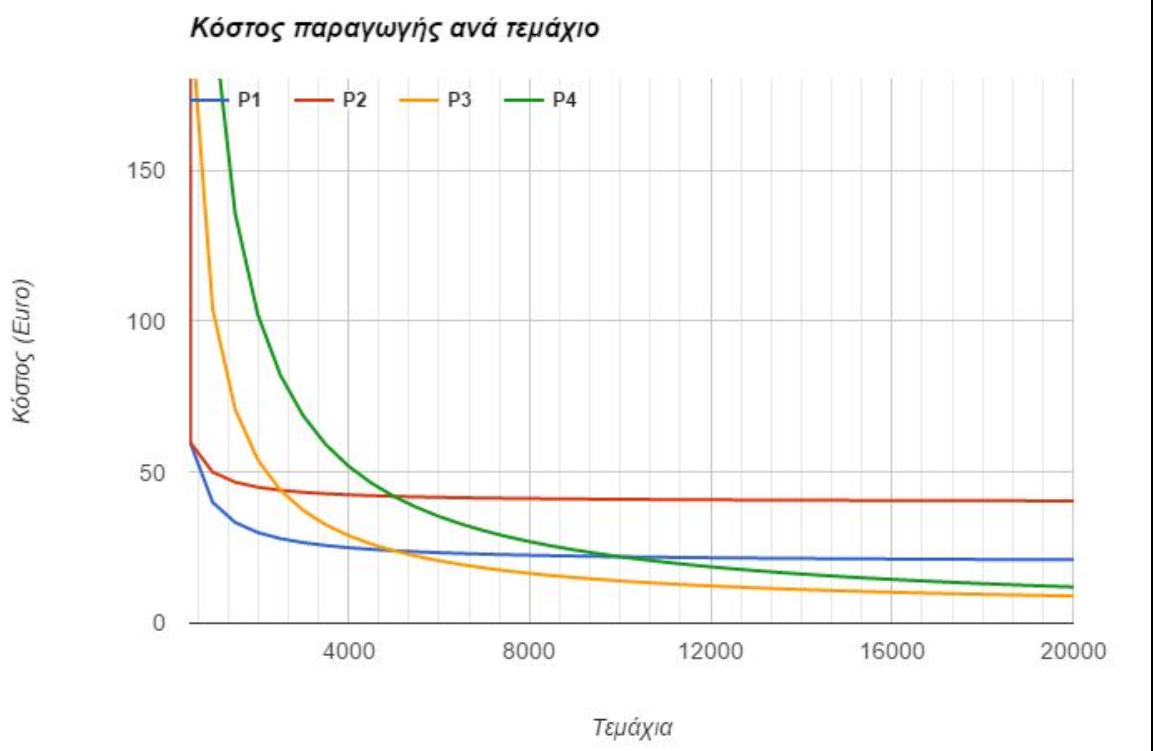
1) $P_1(z) = 20.000/z + 20$
2) $P_2(z) = 10.000/z + 40$
3) $P_3(z) = 100.000/z + 4$
4) $P_4(z) = 200.000/z + 2$

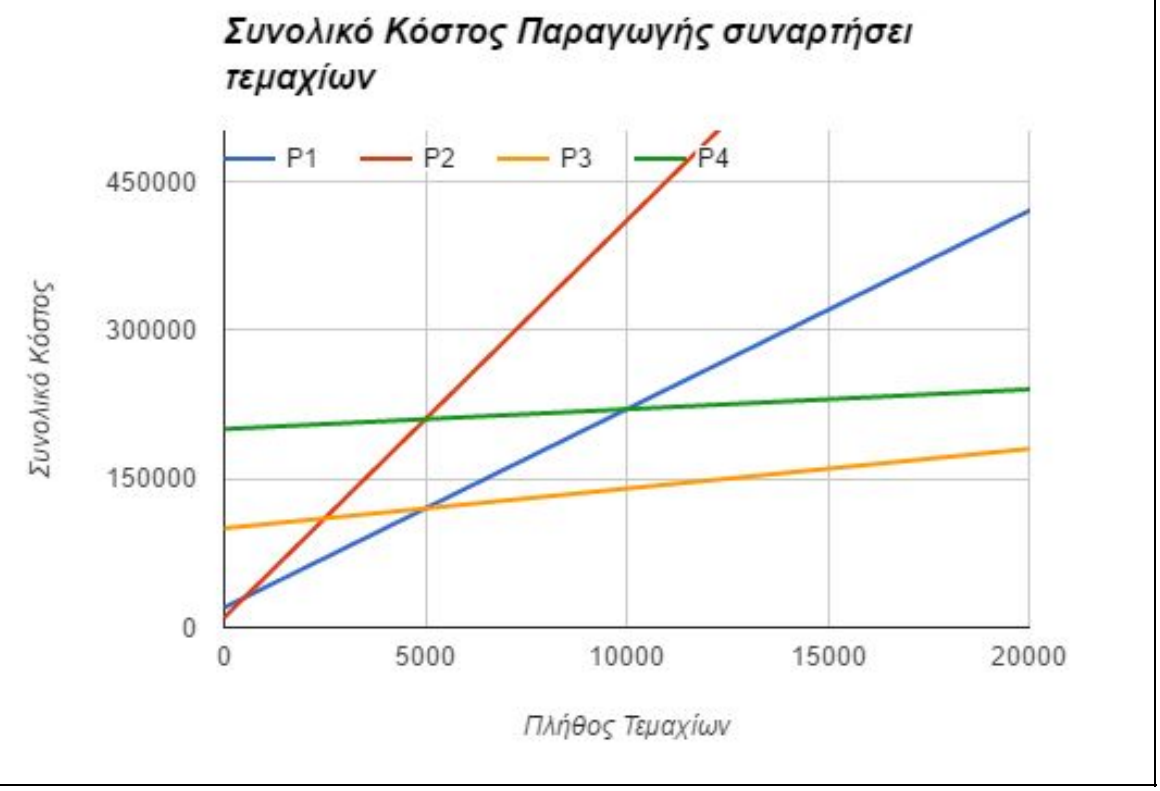
Παρακάτω, έχοντας λάβει υπόψιν τις γραφικές απεικονίσεις, υπολογίζονται οι οριακές τιμές μέχρι τις οποίες συμφέρει η κάθε λύση:

- **Λύση 2:**
  Για να συμφέρει η λύση Β, πρέπει να έχει το μικρότερο κόστος. Προσδιορίζουμε με την εξίσωση το πλήθος τεμαχίων διότι στο γράφημα δεν είναι ορατό.
  P2(z)=P1(z) => z= 500, άρα συμφέρει από 1 έως 500 τεμάχια.
- **Λύση 1:**
  P1(z) = P3(z) => z = 5.000, άρα συμφέρει από 500 έως 5000 τεμάχια
- **Λύση 3:**
  P3(z)=P4(z) => z= 50.000. Άρα συμφέρει από 5000 έως 50000 τεμάχια.
- **Λύση 4:**
  Από 50000 τεμάχια και πάνω.

Συνοπτικά:

- **Από 1 έως 500 τεμάχια** συμφέρει **η λύση 2.**
- **Από 500 έως 5000 τεμάχια** συμφέρει **η λύση 1.**
- **Από 5000 έως 50000 τεμάχια** συμφέρει **η λύση 3.**
- **Από 50000 και πάνω** συμφέρει **η λύση 4.**





Υποερώτημα Β

- Εάν επιφέρουμε **αλλαγή στην αρχική τιμή σχεδίασης της λύσης 3,** τότε ουσιαστικά για να εξαφανιστεί η τεχνολογία 1, πρέπει από την παραγωγή 500 έως 5000 τεμαχίων τουλάχιστον, πλέον, να συμφέρει η τεχνολογία 3 (μετατόπιση ευθείας συνολικού κόστος προς τα κάτω, λόγο μείωσης αρχικού κόστους). Έχουμε κόστος P1(500) = 60 ευρώ / τεμάχιο, και άρα πρέπει $P3(500) \leq 60 \Rightarrow x/500 + 4 \leq 60 \Rightarrow x \leq 28000$ ευρώ.Άρα πρέπει να είναι το αρχικό κόστος μικρότερο από **28000 ευρώ**.
- Εάν επιφέρουμε **αλλαγή στην αρχική τιμή σχεδίασης της λύσης 2, ,** τότε ουσιαστικά για να εξαφανιστεί η τεχνολογία 1, πρέπει έως και την παραγωγή τεμαχίων τουλάχιστον, πλέον, να συμφέρει η τεχνολογία 2 (αλλαγή κλίσης ευθείας συνολικού κόστος προς τα κάτω, λόγο μείωσης κόστους τεμαχίου). Έχουμε κόστος P1(5000) = 24 ευρώ / τεμάχιο, και άρα πρέπει $P2(5000) \leq 24 \Rightarrow 10000/5000 + 10 + x \leq 24 \Rightarrow x \leq 12$ Άρα πρέπει να είναι το κόστος να είναι μικρότερο από **12 ευρώ**.

Άσκηση 5

Πρόβλημα 3.31

```
module circuit_3_20 (A, B, C, C_inv, D, F);
        input A, B, C, D;
        output F;
        wire w1, w2, w3, w4, w5;
        and G1 (w1, C, D),
             G2 (w2, b, C_inv),
             G3 (w4, w3, A);
        or G4(w3, w1, B),
           G5(F, w4, w2);
endmodule

module circuit_3_23b (F, A, B, C, D, A_inv, B_inv, C_inv);
        input A, B, C, D, A_inv, B_inv, C_inv;
        output F;
        wire w1, w2, w3, w4, w5, w6, w7;

        nand G1 (w1, A, B_inv);
        nand G2 (w1, A_inv, B);
        not (w1_inv, w1);
        not (w2_inv, w2);
        or G4 (w3, w1_inv, w2_inv);
        or G3 (w4, C, D_inv);
        not (w5, C_inv);
        not (w6, D);
```

```
        nand G5 (w7, w5, w6)
        not(F, w7)
endmodule

module circuit_3_24 (F, A, B, C, D, E_inv);
        input A, B, C, D, E_inv;
        output F;
        wire w1, w2, w1_inv, w2_inv, E;
        nor (w1, A, B);
        nor (w2, C, D);
        not( w1_inv, w1);
        not( w2_inv, w2);
        not (E, E_inv);
        and (F, w1_inv, w2_inv, E);
end module

module circuit_3_25 (F, A, B, C, D, A_inv, B_inv, D_inv);
        input A, B, C, D, A_inv, B_inv, D_inv;
        output F;
        wire w1, w2, w3, w4, w5, w6, w7, w8, w9, w10;
        not (w5, A_inv);
        not (w6, B);
        not (w7, A);
        not (w8, B_inv);
        not (w10, w3);
        and (w1, w5, w6);
        and (w2, w7, w8);
        and (F, w9, w10);
        nor (w3, C, D_inv);
        nor (w4, w1, w2);
endmodule
```

Πρόβλημα 3.32

```
module circuit_3_20b( A, B, C, D, B_inv, C_inv, F);
        output F;
        input A, B, C, D, B_inv, C_inv;
        assign F = (~(~(((~(~(D & C))) | (~B_inv) ) & A))) | (~(~(B & C_inv )) )
endmodule

module circuit_3_21a( A, B, C, A_inv, B_inv, D_inv, F);
        output F;
        input A, B, C, A_inv, B_inv, D_inv
        assign F = ((A & B_inv) | (A_inv & B)) & (C | D_inv);
endmodule

module circuit_3_24 (A, B, C, D, E_inv, F);
        output F;
        input A, B, C, D, E_inv;
        assign F = (~(~(A|B))) | (~(~(C | D))) | (~E_inv);
endmodule
```

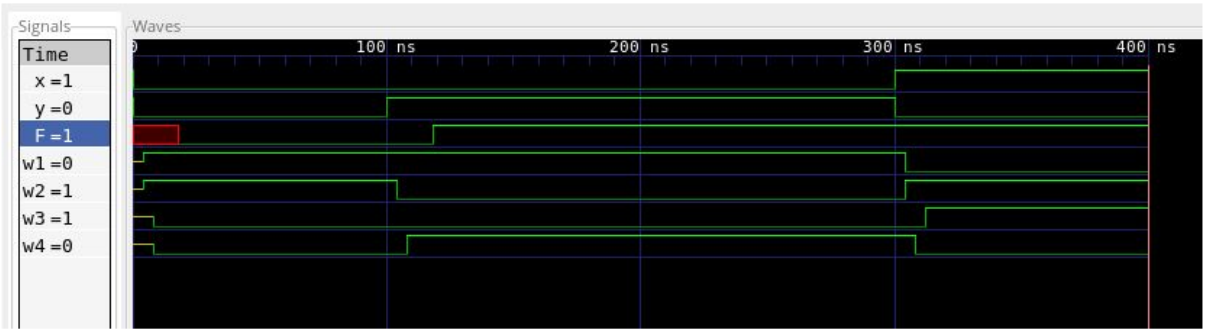
```
module circuit_3_25 (A, B, C, A_inv, B_inv, D_inc, F);
	output F;
	input A, B, C, A_inv, B_inv, D_inc;
	assign F = (~(~(((~A_inv) & (~B)) | ((~A) & (~B_inv)))) & (~(~(C | D_inv)))
endmodule
```

Άσκηση 6

Πρόβλημα 3.33

```
module circuit_3_30 (F, x, y);
	output F;
	input x, y;
	wire w1, w2, w3, w4;
	not #(4) (w1, x);
	not #(4) (w2, y);
	and #(8) (w3, x, w2);
	and #(8) (w4, w1, y);
	or #(10) (F, w3, w4);
endmodule
```

Η προσομοίωση με τη χρήση icarus verilog και gtk wave (ο κώδικας βρίσκεται στο αρχείο circuit.v), δίνει το παρακάτω γράφημα:

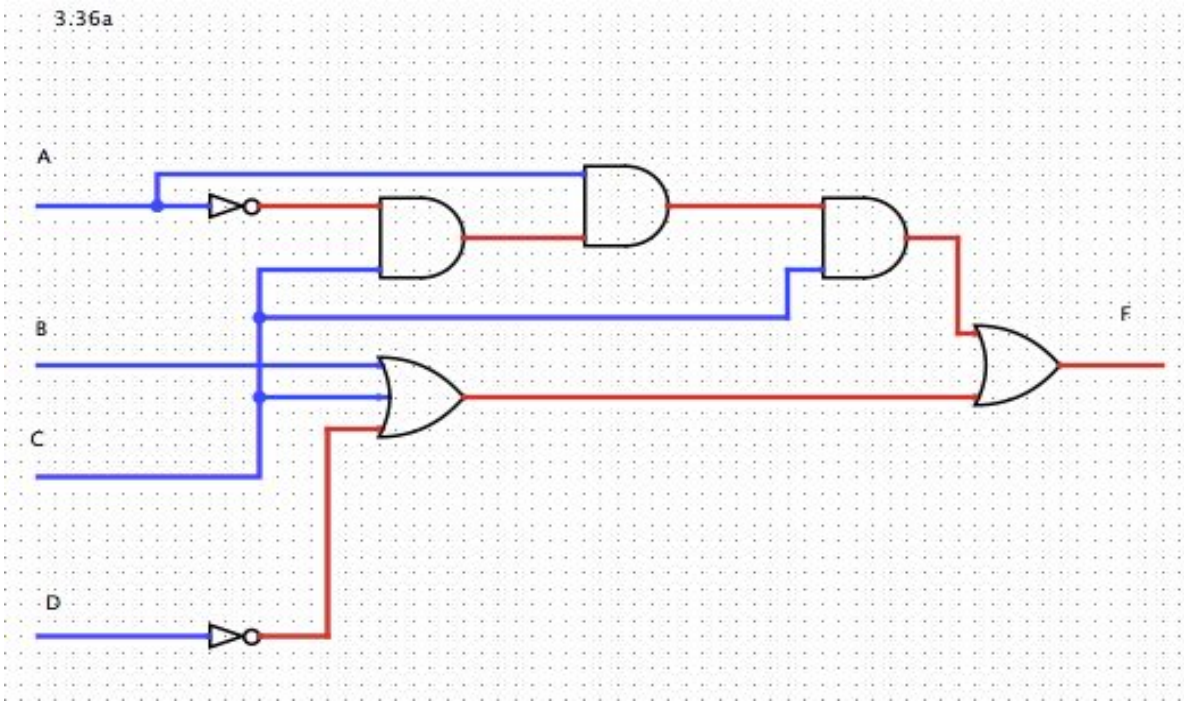


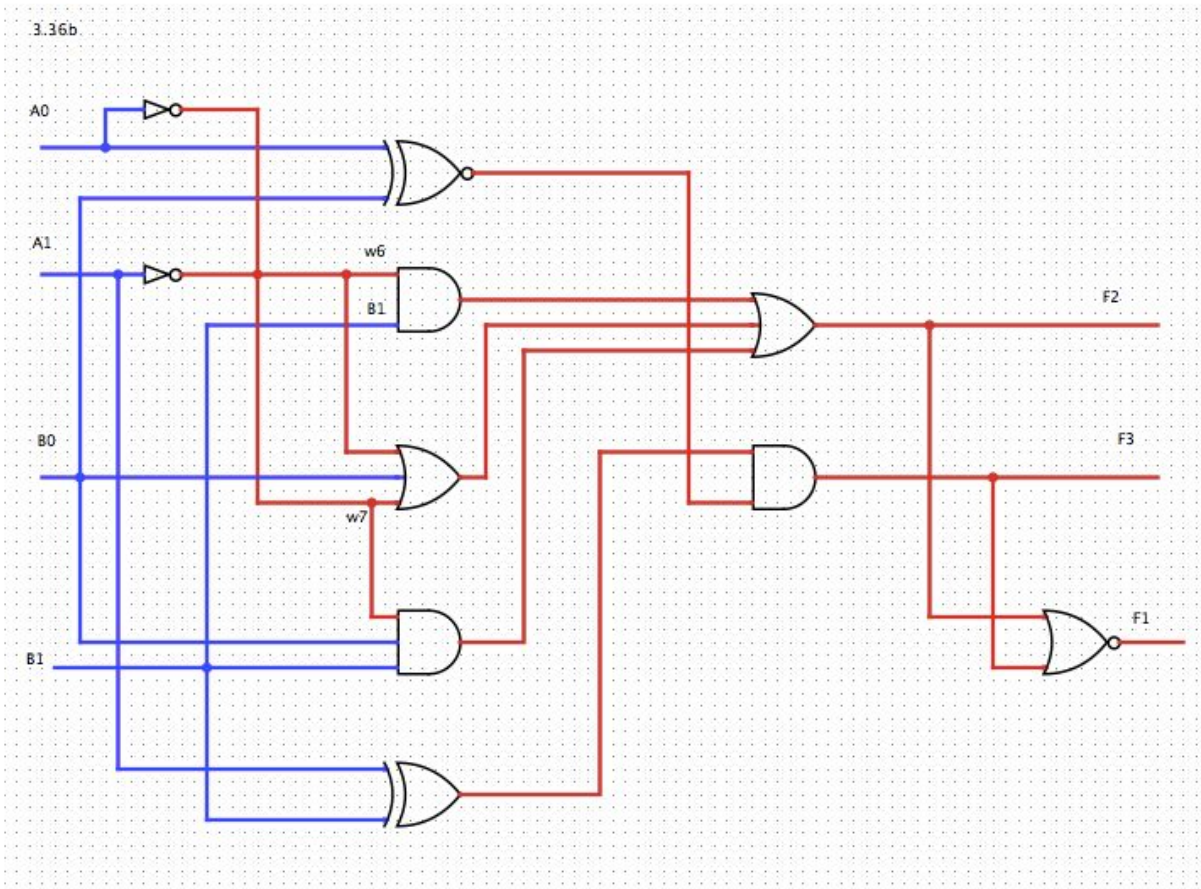
Πρόβλημα 3.34

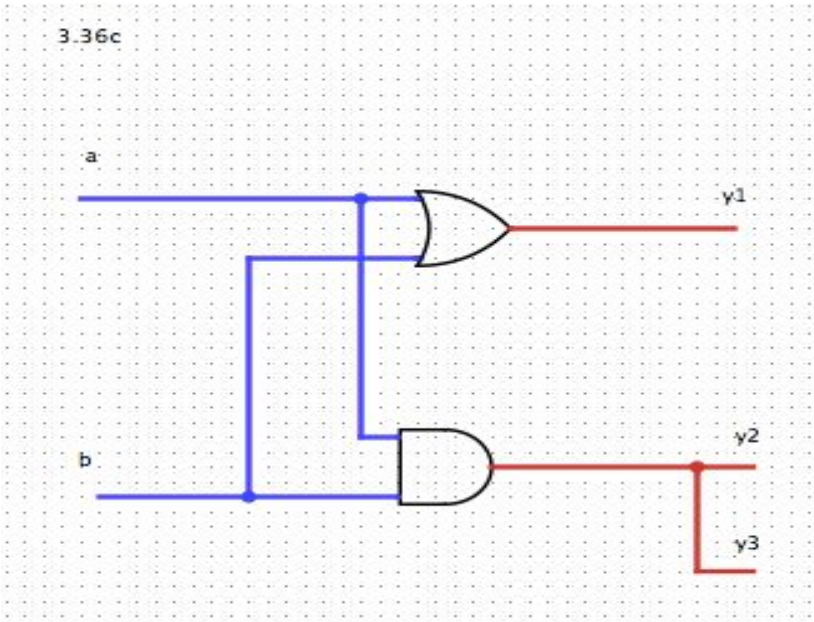
```
module circuit_3_34 (Out1, Out2, Out3, A, B, C, D);
	output Out1, Out2, Out3;
	input A, B, C, D;
	assign Out1 = (A | ~B) & ~C & (C|D);
	assign Out2 = ((~C & D) | (B & C & D) | (C & ~D)) & (~ A | B);
	assign Out3 = ((A & B | C) & D) | ~B & C
endmodule
```

Πρόβλημα 3.36

Ακολουθούν τα διαγράμματα του προβλήματος 3.36:


3.36a
A
B
C
D
F


3.36b
A0
A1
B0
B1
w6
w7
F2
F3
F1



Άσκηση 7

Πρόβλημα 4.36

```
module_circuit_4_23 (D, x, y, v)
        output x, y, v;
        input [0:3] D;
        not (w1, D[2]);
        and (w3, w1, D[1]);
        or (y, D[3], w3);
        or (w1, D[3], D[2]);
        or (v, w2, D[1], D[0]);
endmodule;
```

Πρόβλημα 4.45

```
module_circuit_4_23 (D, x, y, v)
        output x, y, v;
        input [0:3] D;
        reg x,y, v;
        always @(D[0], D[1], D[2], D[3])
        begin
                if (D[3])
                begin
                        x=1;
                        y=1;
                end
                else if(D[2])
                begin
                        x=1;
```

```
                y=0;
            end
            else if(D[1])
            begin
                x=0;
                y=1;
            end
            else
            begin
                x=0;
                y=0;
            end
        if (D[0] || D[1] || D[2] || D[3]) v=1;
        else v=0;
        end
endmodule
```