Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2019-2020

Πεγείωτη Νάταλυ
03117707

1η ΟΜΑΔΑ ΑΣΚΗΣΕΩΝ
ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Συστήματα Μικροϋπολογιστών"

Ασκήσεις Προσωμοίωσης

1η ΑΣΚΗΣΗ

Δίνεται το πιο κάτω πρόγραμμα σε γλώσσα μηχανής:
06 01 **3A** 00 20 **FE** 00 **CA** 13 08 **1F DA** 12 08 **04 C2** 0A 08 **78 2F 32** 00 30 **CF**

Διεύθυνση	Ετικέτα	Περιεχόμενο (γλώσσα μηχανής)	Εντολή (γλώσσα assembly)	Περιγραφή εντολής
0800	START:	**06**	MVI B, 01H	Move the immediate 01H to register B
0801		01		
0802		**3A**	LDA 2000H	Load address 2000H to register A
0803		00		
0804		20		
0805		**FE**	CPI 00H	Compare immediate with 00
0806		00		
0807		**CA**	JZ 0813H	Jump to address 0813H (TARGET1) if result is 0
0808		13		
0809		08		
080A	TARGET3:	**1F**	RAR	Rotate (Only CARRY affected) Shift the accumulator A to the right, using CY (CY gets the least significant bit)
080B		**DA**	JC 0812H	Jump to address 0812H (TARGET2) if CY=1
080C		12		
080D		08		
080E		**04**	INR B	Increment register B (B = B + 1)
080F		**C2**	JNZ 080AH	Jump-if-not-Zero to address 080A (TARGET3)
0810		0A		
0811		08		
0812	TARGET2:	**78**	MOV A, B	Move the contents of register B to register A
0813	TARGET1:	**2F**	CMA	Complement A
0814		**32**	STA 3000H	Store Address 3000H
0815		00		
0816		30		
0817		**CF**	RST 1	Διακοπή 1 αφού η εντολή είναι RST n με n = 1 άρα θέση μνήμης 0008 (αφού στην RST n έχουμε n x 8)

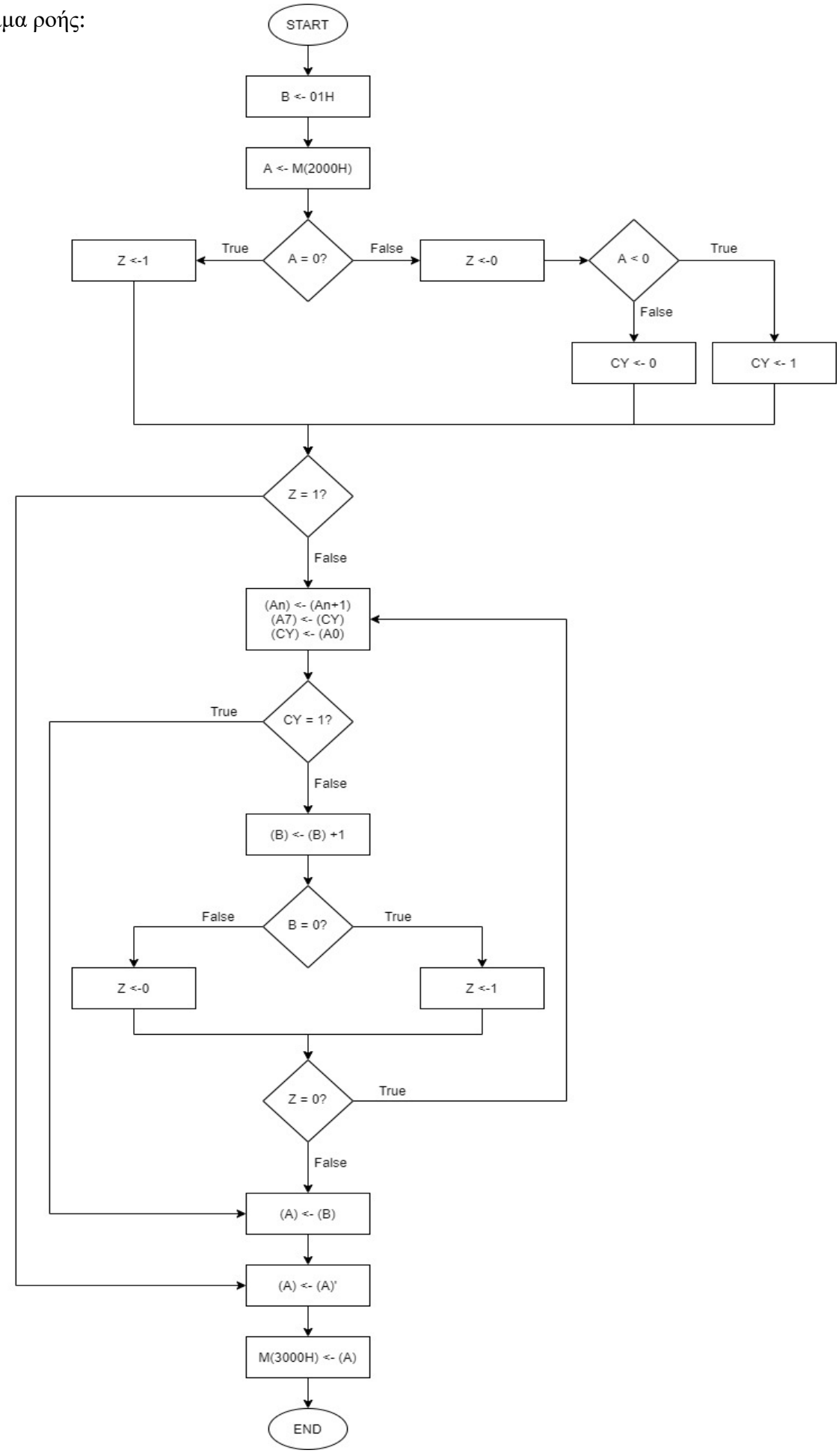
Το πρόγραμμα φαίνεται αποκωδικοποιημένο (σε γλώσσα assembly) πιο κάτω:

```
START:    MVI B, 01H
          LDA 2000H
          CPI 00H
          JZ TARGET1

TARGET3:  RAR
          JC TARGET2
          INR B
          JNZ TARGET3

TARGET2:  MOV A, B

TARGET1:  CMA
          STA 3000H
          RST 1
END
```

Η λειτουργία του προγράμματος περιγράφεται ακολούθως:

Ο καταχωρητής B αρχικοποιείται με την τιμή 01_{hex}. ***MVI B, 01H***
Το περιεχόμενο της διεύθυνσης μνήμης 2000_{hex} φορτώνεται στον καταχωρητή A. ***LDA 2000H***
Το περιεχόμενο του καταχωρητή A συγκρίνεται με την τιμή 0. ***CPI 00H***
Αν A<0, τότε CY=1
Αν A>0, τότε CY=0
Αν A=0, τότε Z=1
Αν Z=1, δηλαδή αν A=0, τότε πραγματοποιείται άλμα στη διεύθυνση μνήμης που δείχνει η ετικέτα TARGET1, δηλαδή στη διεύθυνση 0813_{hex} . ***JZ TARGET1***
Εάν δεν πραγματοποιηθεί το άλμα της προηγούμενης εντολής το πρόγραμμα προχωρά στην ακόλουθη εντολή, η οποία πραγματοποιεί δεξιά περιστροφή του καταχωρητή A κατά 1 bit, μέσω του κρατουμένου CY. ***RAR***
Εξετάζεται το περιεχόμενο του CY, και αν ισούται με 1 τότε πραγματοποιείται άλμα στη διεύθυνση μνήμης που δείχνει η ετικέτα TARGET2, δηλαδή στη διεύθυνση 0812_{hex}.
JC TARGET2
Εάν δεν πραγματοποιηθεί το άλμα της προηγούμενης εντολής, το περιεχόμενο του καταχωρητή B αυξάνεται κατά 1. Αν το περιεχόμενο του B ισούται με 0, τότε Z=1 αλλιώς Z=0. ***INR B***
Αν Z=0 (δηλαδή B≠0), τότε πραγματοποιείται άλμα στη διεύθυνση μνήμης που δείχνει η ετικέτα TARGET3, δηλαδή στη διεύθυνση $080A_{hex}$. ***JNZ TARGET3***
Εάν δεν πραγματοποιηθεί το άλμα της προηγούμενης εντολής, τότε το πρόγραμμα προχωρά στην ακόλουθη εντολή, η οποία μεταφέρει το περιεχόμενο του καταχωρητή B στον καταχωρητή A.
MOV A, B
Στον καταχωρητή A, αποθηκεύεται το συμπλήρωμα ως προς 1 του ιδίου. ***CMA***
Η τιμή του καταχωρητή A, αποθηκεύται στη διεύθυνση μνήμης 3000_{hex}. ***STA 3000H***
Πραγματοποιείται διακοπή 1, δηλαδή κλήση της υπορουτίνας που αρχίζει στη διεύθυνση 0008_{hex}, η οποία σώζει τα περιεχόμενα των καταχωρητών στη RAM και επιστρέφει στο monitor πρόγραμμα. ***RST 1***

Διάγραμμα ροής:



Για να επαναλαμβάνεται συνεχώς η λειτουργία του προγράμματος χωρίς τέλος, μπορούμε να αντικαταστήσουμε την τελευταία εντολή, δηλαδή την εντολή διακοπής προγράμματος RST1 με μια εντολή υποχρεωτικού άλματος JMP 0800H, στη διεύθυνση 0800H, όπου δηλώσαμε την αρχή του προγράμματος με την ετικέτα START.

2η ΑΣΚΗΣΗ

Κώδικας σε assembly

```
        IN 10H
        LXI B,01F4H   ; BC = 500
        MVI D,01H     ; Αρχικοποίηση του καταχωρητή D με 1
        MVI E,00H     ; Αρχικοποίηση καταχωρητή E (direction flag) με 0 (αριστερή περιστροφή)

START:CALL DELB       ; Προσθήκη delay = BC*1ms = 500*10^-3 = 0,5 s
        LDA 2000H     ; Φόρτωση του περιεχομένου της θέσης 2000H (dip switch) στον A
        ANI 02H       ; Απομόνωση του 2ου LSB του A και καταχώρηση του στον ίδιο
        CPI 01H       ; Σύγκριση του A με την τιμή 1, αν A μικρότερο τότε CY = 1, αλλιώς  CY = 0
        JC LSB        ; Αν CY = 1 (δηλαδή 2ο LSB = 0), τότε άλμα στην ετικέτα LSB
        MVI A,01H     ; Καταχώρηση της τιμής 1 στον A
        CMA           ; Συμπλήρωμα ως προς 1 του A και καταχώρηση του στον ίδιο
        STA 3000H     ; Ενημέρωση της κατάστασης  του LED στη θεση 3000H
        JMP START     ; Άλμα στην ετικέτα START

LSB:    LDA 2000H     ; Φόρτωση του περιεχομένου της θέσης 2000H (dip switch) στον A
        ANI 01H       ; Απομόνωση του LSB του A και καταχώρηση του στον ίδιο
        CPI 01H       ; Σύγκριση του A με την τιμή 1, αν A μικρότερο τότε CY = 1, αλλιώς  CY = 0
        JC LEFT       ; Αν CY = 1 (δηλαδή LSB = 0), τότε άλμα στην ετικέτα LEFT
        MOV A,D       ; Μεταφορά του περιεχομένου του D (προηγούμενη τιμή της 2000H) στον A
        CPI 80H       ; Σύγκριση του A με την τιμή 10000000, αν ισούνται τότε Z = 1, αλλιώς Z = 0
        JZ RIGHT      ; Αν Z = 1, τότε άλμα στην ετικέτα RIGHT
        MOV A,E       ; Μεταφορά του περιεχομένου του E (flag direction) στον A
        CPI 01H       ; Σύγκριση του A με την τιμή 1, αν ισούνται τότε Z = 1, αλλιώς Z = 0
        JZ RIGHT      ; Αν Z = 1, τότε άλμα στην ετικέτα RIGHT
LEFT:   MOV A,D       ; Μεταφορά του περιεχομένου του D στον A
        RLC           ; Αριστερή περιστροφή των ψηφίων του A
        CMA           ; Συμπλήρωμα ως προς 1 του A και καταχώρηση του στον ίδιο
        STA 3000H     ; Ενημέρωση της κατάστασης του LED στη θέση 3000H
        CMA
        MOV D,A       ; Μεταφορά του περιεχομένου του A (ενημερωμένη τιμή) στον D
        JMP START     ; Άλμα στην ετικέτα START

RIGHT:MVI E,01H       ; Φόρτωση στον E (direction flag) της τιμής 1 (δεξιά περιστροφή)
        MOV A,D       ; Μεταφορά του περιεχομένου του D στον A
        RRC           ; Δεξιά περιστροφή των ψηφίων του A
        CMA
        STA 3000H     ; Ενημέρωση της κατάστασης του LED στη θέση 3000H
        CMA
        MOV D,A       ; Μεταφορά του περιεχομένου του A (ενημερωμένη τιμή) στον D
        CPI 01H       ; Σύγκριση του A με την τιμή 1, αν ισούνται τότε Z = 1, αλλιώς Z = 0
        JNZ START     ; Αν Z = 0, τότε άλμα στην ετικέτα START
        MVI E,00H     ; Φόρτωση στον E (direction flag) της τιμής 0 (αριστερή περιστροφή)
        JMP START     ; Άλμα στην ετικέτα START

END
```

3η ΑΣΚΗΣΗ

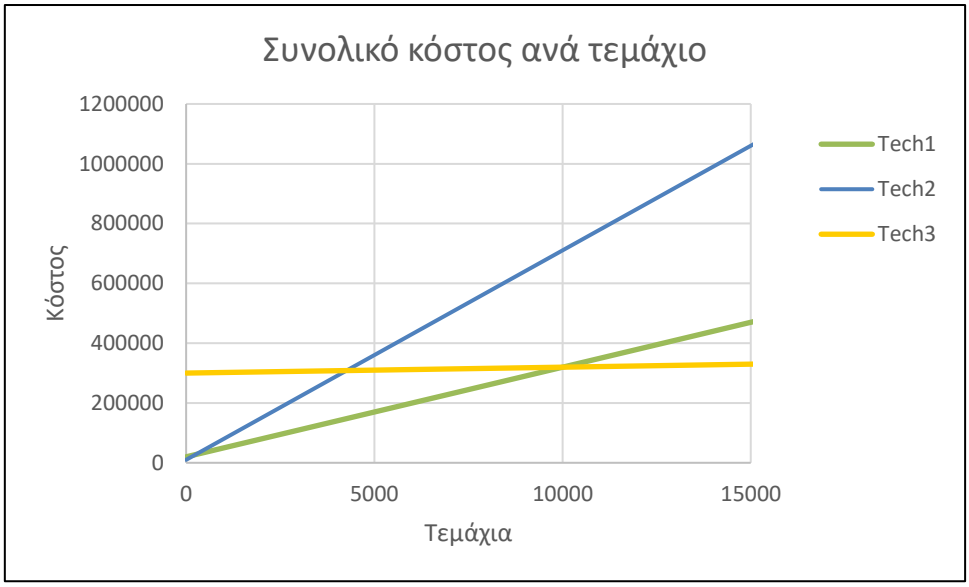
```
        IN 10H
START:MVI B,FFH   ; Αρχικοποίηση του B (μετρητής δεκάδων) με -1
        LDA 2000H ; Φόρτωση του περιεχομένου της θέσης 2000H στον A
MOD:  CPI 64H     ; Έλεγχος αν το A είναι μεγαλύτερο του 99, αν ναι CY = 0, αλλιώς CY = 1
        JC DECA   ; Αν CY = 1 (δηλαδή A <= 99), τότε άλμα στην ετικέτα DECA
        SUI 64H           ; Μείωση του A κατά 100
        JMP MOD   ; Άλμα στην ετικέτα MOD

DECA: INR B       ; Άυξηση του B (μετρητής δεκάδων) κατά 1
        SUI 0AH   ; Μείωση του A κατά 10
        JNC DECA  ; Αν A > 0, τότε άλμα στην ετικέτα DECA
        ADI 0AH   ; Αύξηση του A κατά 10 (το περιεχόμενο του A ισούται με τις μονάδες)
        MOV C,A   ; Μεταφορά του περιεχομένου του A (μονάδες) στον C
        MOV A,B   ; Μεταφορά του περιεχομένου του B (δεκάδες) στον A
        RLC       ; Αριστερή περιστροφή των ψηφίων του A
        RLC       ; η οποία επαναλαμβάνεται 4 φορές
        RLC       ; έτσι ώστε να μεταφερθεί ο αριθμός των δεκάδων
        RLC       ; στα 4 MSB ψηφία του καταχωρητή A
        ADD C     ; Προσθήκη του περιεχομένου του καταχωρητή C (μονάδες) στον A
        CMA       ; Συμπλήρωμα ως προς 1 του A και καταχώρησή του στον ίδιο
        STA 3000H ; Αποθήκευση του περιεχομένου του A στη θέση 3000H
        JMP START ; Άλμα στην ετικέτα START για επανάληψη της λειτουργίας

END
```

Θεωρητικές Ασκήσεις

4η ΑΣΚΗΣΗ

Κόστος τεχνολογιών ανά τεμάχιο:

1. Tech1 = 20.000 + 15*Τεμάχιο + 15*Τεμάχιο
2. Tech2 = 10.000 + 60*Τεμάχιο + 10*Τεμάχιο
3. Tech2 = 300.000 + 1*Τεμάχιο + 1*Τεμάχιο



Το σημείο τομής της καμπύλης της Tech1 (1η τεχνολογία) και της καμπύλης της Tech2 (2η τεχνολογία) υπολογίζεται ως εξής:

$$Tech1 = Tech2$$
$$20000 + 15 \times Τεμάχιο + 15 \times Τεμάχιο = 10000 + 60 \times Τεμάχιο + 10 \times Τεμάχιο$$
$$10000 = 40 \times Τεμάχιο$$
$$Τεμάχιο = 250$$

- Tech1 > Tech2 για αριθμό τεμαχίων < 250
- Tech1 = Tech2 για αριθμό τεμαχίων ίσο με 250
- Tech1 < Tech2 για αριθμό τεμαχίων > 250

Το σημείο τομής της καμπύλης της Tech1 (1η τεχνολογία) και της καμπύλης της Tech3 (3η τεχνολογία) υπολογίζεται ως εξής:

$$Tech1 = Tech3$$
$$20000 + 15 \times Τεμάχιο + 15 \times Τεμάχιο = 300000 + 1 \times Τεμάχιο + 1 \times Τεμάχιο$$
$$280000 = 28 \times Τεμάχιο$$
$$Τεμάχιο = 10000$$

- Tech1 < Tech3 για αριθμό τεμαχίων < 10000
- Tech1 = Tech3 για αριθμό τεμαχίων ίσο με 10000
- Tech1 > Tech3 για αριθμό τεμαχίων > 10000

Το σημείο τομής της καμπύλης της Tech2 (2η τεχνολογία) και της καμπύλης της Tech3 (3η τεχνολογία) υπολογίζεται ως εξής:

$$Tech2 = Tech3$$
$$10000 + 60 \times Τεμάχιο + 10 \times Τεμάχιο = 300000 + 1 \times Τεμάχιο + 1 \times Τεμάχιο$$
$$290000 = 68 \times Τεμάχιο$$
$$Τεμάχιο = 4264.7 \approx 4265$$

- Tech2 < Tech3 για αριθμό τεμαχίων < 4264
- Tech2 = Tech3 για αριθμό τεμαχίων ίσο με 4264.7 ≈ 4265
- Tech2 > Tech3 για αριθμό τεμαχίων > 4265

Από τα πιο πάνω συμπεραίνουμε ότι:

- Για αριθμό τεμαχίων < 250 είναι συμφερότερη η 2η τεχνολογία
- Για αριθμό τεμαχίων > 250 και < 10000 είναι συμφερότερη η 1η τεχνολογία
- Για αριθμό τεμαχίων > 10000 είναι συμφερότερη η 3η τεχνολογία

Για να εξαφανιστεί η 1η τεχνολογία Tech1 πρέπει το συνολικό κόστος της 2ης τεχνολογίας Tech2 ανά τεμάχιο να μικρότερο ή ίσο με το κόστος της 1ης τεχνολογίας , για οποιαδήποτε ποσότητα τεμαχίων μέχρι 10000. Αυτό συμβαίνει γιατί για περισσότερα από 10000 τεμάχια είναι συμφερότερη η 3η τεχνολογία.
Για τον υπολογισμό του κόστους k των I.C. της 2ης τεχνολογίας χρησιμοποιούμε την πιο κάτω ισότητα:

$$20000 + 15 \times 10000 + 15 \times 10000 = 10000 + k \times 10000 + 10 \times 10000$$
$$210000 = k \times 10000$$
$$k = 21$$

Άρα το κόστος των I.C. της 2ης τεχνολογίας πρέπει να είναι 21€ ανά τεμάχιο.

Ασκήσεις στη γλώσσα περιγραφής υλικού Verilog

5η ΑΣΚΗΣΗ

F1 = A(CD+B) + BC'D'

i.
```
module F1 (E, A, B, C, D);
output E;
input A, B, C, D;
wire Cnot, Dnot, out3, out4, out5, out6;

not
G1(Cnot, C),
G2(Dnot, D);

and
G3(out3, C, D),
G4(out4, out6, A),
G5(out5, B, Cnot, Dnot);

or
G6(out6, out3, B),
G7(E, out4, out5);
endmodule
```

ii.
```
module F1 (A, B, C, D, E);
output E;
input A, B, C, D;

assign E = A&(C&D | B) | (B&~C&~D);

endmodule
```

F2(A, B, C, D) = Σ(0, 2, 3, 5, 7, 8 , 10, 11, 14, 15)
= A'B'C'D' + A'B'CD' + A'B'CD + A'BC'D + A'BCD + AB'C'D' +AB'CD' + AB'CD + ABCD' + ABCD

i.
```
module F2(E, A, B, C, D);
output E;
input A, B, C, D;
wire Anot, Bnot, Cnot, Dnot, out5, out6, out7, out8, out9, out10, out11, out12, out13, out 14;

not
G1(Anot, A),
G2(Bnot, B),
G3(Cnot, C),
G4(Dnot, D);

and
G5(out5, Anot, Bnot, Cnot, Dnot),
G6(out6, Anot, Bnot, C, Dnot),
G7(out7, Anot, Bnot, C, D),
G8(out8, Anot, B, Cnot, D),
G9(out9, Anot, B, C, D),
G10(out10, A, Bnot, Cnot, Dnot),
G11(out11, A, Bnot, C, Dnot),
G12(out12, A, Bnot, C, D),
G13(out13, A, B, C, Dnot),
G14(out14, A, B, C, D);

or
G15(E, out5, out6, out7, out8, out9, out10, out11, out12, ou13, out14);

endmodule
```

ii.
```
module F2(E, A, B, C, D);
output E;
input A, B, C, D;

assign E = (~A&~B$~C&~D) | (~A&~B&C&~D) | (~A&~B&C&D) |
(~A&B&~C&D) | (~A&B&C&C) | (A&~B&~C&~D) | (A&~B&C&~D) |
(A&~B&C&D) | (A&B&C&~D) | (A&B&C&D)

endmodule
```

F3 = ABC + (A+B)CD + (B + CD)E

i.
```
module F3(F, A, B, C, D, E);
output F;
input A, B, C, D, E;
wire out1, out2, out3, out4, out5, out6;

and
G1(out1, A, B, C),
G2(out2, C, D),
G3(out3, out2, out5),
G4(out4, out6, E);

or
G5(out5, A, B),
G6(out6, out2, B),
G7(F, out1, out3, out4);

endmodule
```

ii.
```
module F3(F, A, B, C, D, E);
output F;
input A, B, C, D, E;

assign F = (A&B&C) | ((A|B)&C&D) | ((B|(C&D))&E)

endmodule
```

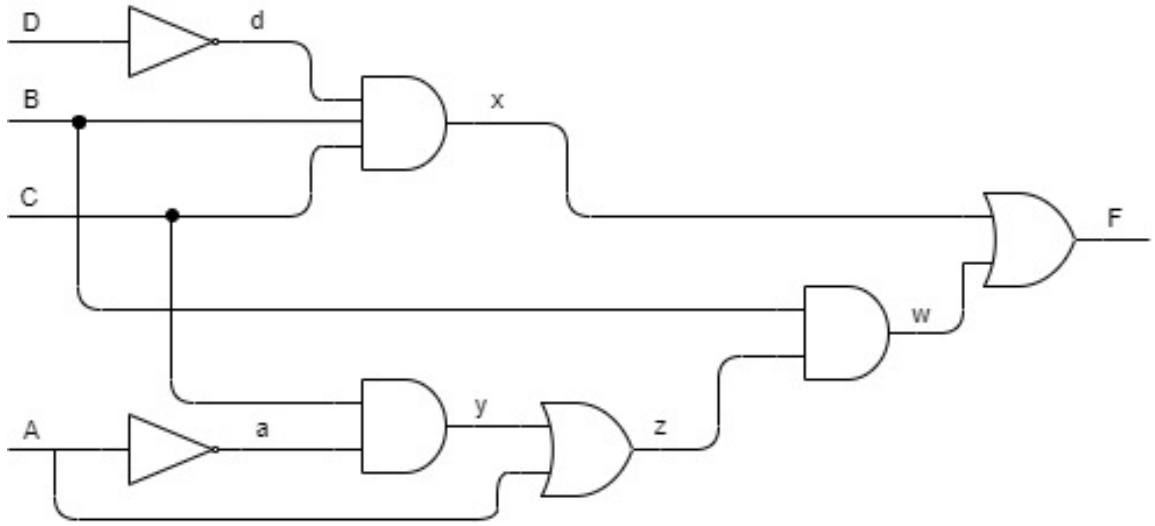
F4 = A(BC + D + E) + CDE

i.
```
module F4(F, A, B, C, D, E);
output F;
input A, B, C, D, E;
wire out1, out2, out3, out4;

and
G1(out1, B, C),
G2(out2, C, D, E),
G3(out3, out4, A);

or
G4(out4, out1, D, E),
G5(F, out2, out3);

endmodule
```

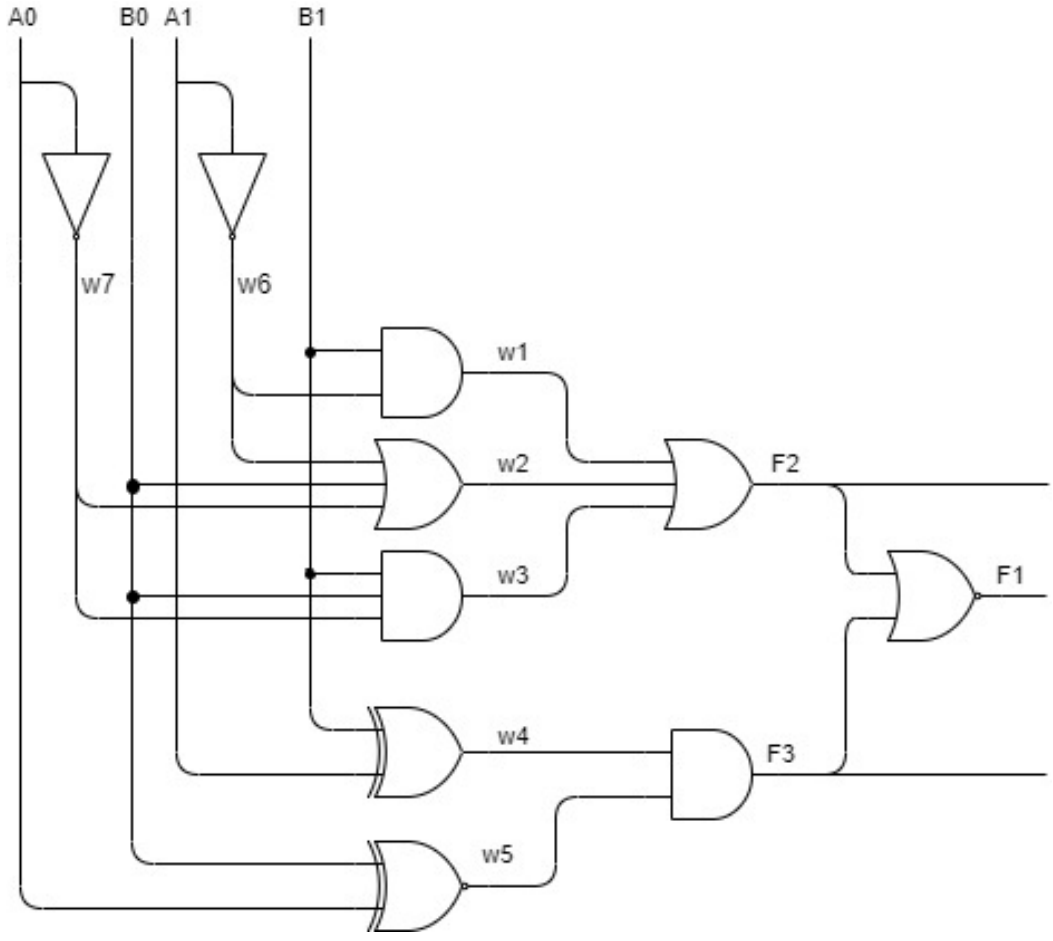
ii.
```
module F4(F, A, B, C, D, E);
output F;
input A, B, C, D, E;

assign F = A&(B&C | D | E) | (C&D&E);

endmodule;
```

6η ΑΣΚΗΣΗ

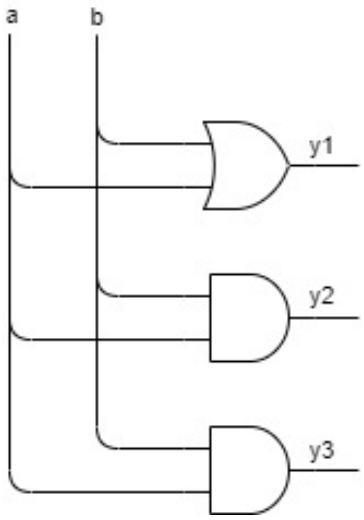
i. (a)



(b)



(c)



ii.
```
module 4_bit_adder(S, C4, A, B, C0);
output [3:0]S;
output C4;
input [3:0] A, B;
input C0;
wire C1, C2, C3;

full_adder FA0 (S[0], C1, A[0], B[0], C0),
           FA1 (S[1], C2, A[1], B[1], C1),
           FA2 (S[2], C3, A[2], B[2], C2),
           FA3 (S[3], C4, A[3], B[3], C3);

endmodule


module full_adder(S, C, x, y, z);
output S, C;
input x, y, z;
wire S1, C1, C2;

half_adder HA1 (S1, C1, x, y),
           HA2 (S, C2, S1, z);

or G1 (C, C2, C1);
endmodule

module half_adder(S, C, x, y);
output S, C;
input x, y;
xor G2(S, x, y);
and G3(C, x, y);
endmodule
```

iii.
```
module 4_bit_adder(S, C4, A, B, C0);
output [3:0]S;
output C4;
input [3:0] A, B;
input C0;

always@(S, A, B)      {C4, S} = C0 ? A – B: A + B;

endmodule
```

7η ΑΣΚΗΣΗ

i.

```
module Mealy_Model(y, x, clock, reset);
output reg y;
input x, clock, reset;
reg [1:0] state, next;
parameter sa = 2'b00;
parameter sb = 2'b01;
parameter sc = 2'b10;
parameter sd = 2'b11;

always @ (posedge clock)
if (reset == 1'b0) state <= sa;
else state <= next;

always @ (state, x)
begin
        next = sa;
        y = 1'b0;
        case (state)
        sa: if (x == 1'b0) begin next = sb; y = 1'b1; end
             else begin next = sc; y = 1'b0; end
        sb: if (x == 1'b0) begin next = sc; y = 1'b0; end
            else begin next = sd; y = 1'b1; end
        sc: if (x == 1'b0) begin next = sb; y = 1'b0; end
            else begin next = sd; y = 1'b1; end
        sd: if (x == 1'b0) begin next = sc; y = 1'b1; end
        else begin next = sa; y = 1'b0; end
        default: begin next = sa; y = 1'b0; end
        endcase

end
endmodule
```

ii.

```
module Moore_Model(y, x, clock, reset);
output reg y;
input x, clock, reset;
reg [1:0] state, next;
parameter sa = 2'b00;
parameter sb = 2'b01;
parameter sc = 2'b10;
parameter sd = 2'b11;

always @ (posedge clock)
if (reset == 1'b0) state <= sa;
else state <= next;

always @ (state, x)
begin
        next = sa;
        y = 1'b0;
        case (state)
        sa: if (x == 1'b0) next = sb;
                else next = sc;
        sb: begin
                y = 1'b1;
                if (x == 1'b0) next = sc;
                else next = sd; end
        sc: begin
                y = 1'b1;
                if (x == 1'b0) next = sb;
                else next = sd; end
        sd: begin
                y = 1b'0;
                if (xin == 1'b0) next = sc;
                else next = sa; end
        default: next = sa;
        endcase

end
endmodule
```

iii.
```
module up_down_counter (count, D, clock, clear);
output reg [3:0] count;
input D, clock, clear;

always @ (posedge clock, negedge clear)
if (clear == 0) count <= 4'b0000;
else case(mode)
        1'b1:  count <= count + 4'b0001;
        1'b0:  count <= count - 4'b0001;
        default: count <= count;
endcase
endmodule
```