Συστήματα Μικρουπολογιστών - 1ή Σειρά Ασκήσεων

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Ακαδημαϊκό έτος : 2018 – 2019 Εξάμηνο : 6ό
Μέλη ομάδας : Βόσινας Κωνσταντίνος ΑΜ : 03116435
Ανδριόπουλος Κωνσταντίνος ΑΜ : 03116023

Άσκηση 1η

Δίνεται το πρόγραμμα σε 8085 σε γλώσσα μηχανής με την ακόλουθη μορφή:

0E 08 3A 00 20 17 DA 0D 08 0D C2 05 08 79 2F 32 00 30 CF

Σύμφωνα με τους πίνακες εντολών αντικαθιστούμε τις διευθύνσεις με τις αντίστοιχες εντολές.

0E ➔ MVI k, byte //Μετακίνηση του αριθμού, εδώ το 08H, στον καταχωρητή k

3A ➔ LDA addr // Φόρτωση στον A το περιεχόμενο της διεύθυνσης addr, εδώ $(2000)_{16}$

17 ➔ RAL //Περιστροφή αριστερά των bits του A και εκχώρηση στο A0 το //περιεχόμενο του CY

DA➔ JC LABEL // Άλμα υπό συνθήκη στο LABEL, εδώ το $(080D)_{16}$, εάν CY==1

OD ➔ DCR k //Αφαίρεση 1 από το περιεχόμενο του καταχωρητή k

C2 ➔ JNZ LABEL //Άλμα υπό συνθήκη στο LABEL, εδώ $(0805)_{16,}$ εάν η σημαία Z!=0

79 ➔ MOV k1,k2 //Αντιγράφει το περιεχόμενο του k2 στον k1

27 ➔ CMA //Συμπλήρωμα ως προς 1 το περιεχόμενο του A

32 ➔ STA addr //Αποθήκευση στη θέση μνήμης addr το περιεχόμενο του A

CF ➔ RST 1 //Διακοπή του κώδικα και PC ← $(0800)_{16}$

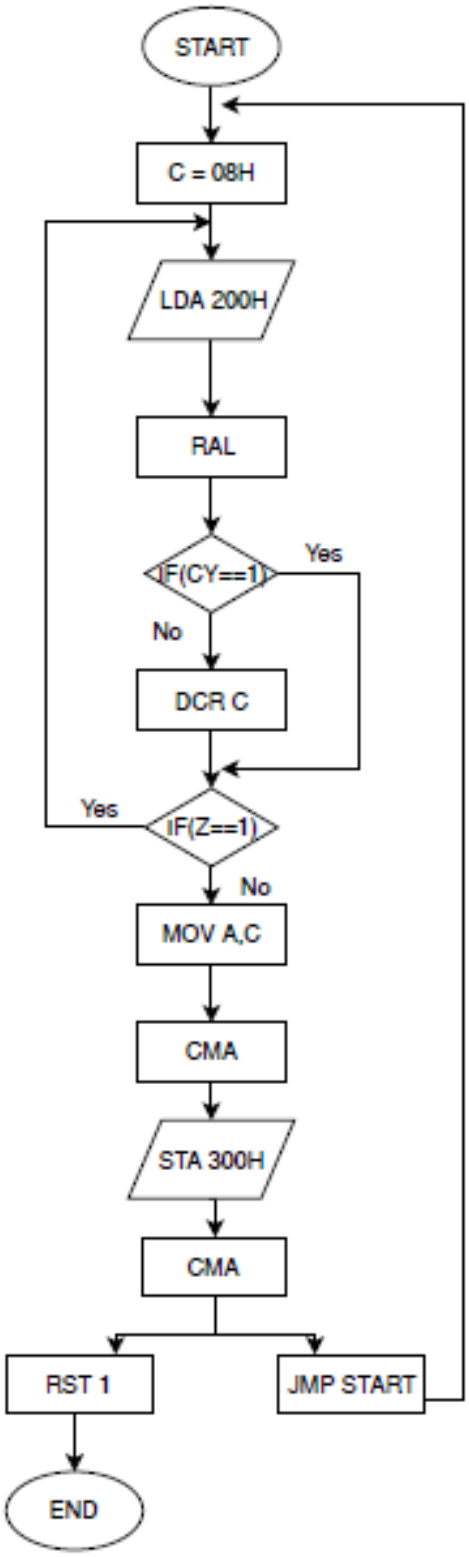
Επομένως ο κώδικας σε γλώσσα assembly 8085 είναι :

```
START:
MVI C,08H
LDA 2000H
L2:
RAL
JC L1
DCR C
JNZ L2
L1:
MOV A,C
CMA
STA 3000H
RST 1
END
```

Μετά την προσωμοίωση στον simulator βλέπουμε πως η λειτουργία του παραπάνω κώδικα είναι η καταμέτρηση του μεγαλύτερου, από αριστερά προς τα δεξιά διακόπτη είναι ανοιχτός. Συγκεκριμένα, εμφανίζεται σε δυαδική αναπαράστση στα led, ο αριθμός του διακόπτη.

Για να λειρουργεί ο κώδικας αενάως αντικαθιστούμε την εντολή `RST 1` με την `JMP START`.

Ακολουθεί το διάγραμμα ροής :



Άσκηση 2η

Το πρόγραμμα που επιτελεί τη λειτουργία της εκφώνησης εμφανίζεται παρακάτω:

```
START:

IN 10H
LXI B,01F4H ;B <- F4H, C<-01H, συνολικα το 500 σε δεκαδικο
            ;Χρηση στην κλήση DELB ώστε να έχουμε 1/2 sec


MVI D,01H   ;D δείχνει ποιό led ανάβει, αρχικά θέτω το 1ο

L1:

LDA 2000H   ;A<- εισοδος από διακόπτες
MOV E,A     ;Προσωρινά σώζω τον A
RAR         ;Ολισθηση δεξιά και CY <- LSB
JC L1       ;Loop εως LSB=0, αλλιως (LSB==1), δεν πραγματοποιειται κινηση
MOV A,E

CALL DELB   ;Καθυστέρηση συμφωνα με BC, 1/2 sec
RAL         ;CY <- MSB και περιστρο
JC LEFT     ;Αν MSB==1 αριστερή κίνηση, αλλιώς συνεχίζουμε δεξια

MOV A,D     ;Καταχωρειται στον A ο αριθμός του led που θα ανάψει
CMA         ;Συμπλήρωμα λόγω αντίστροφης λογικής LED
STA 3000H   ;Εμφάνιση του led που δείχνει ο A
CMA         ;Επαναφορά του A
RRC         ;Shift ΔΕΞΙΑ για να ανάψει το επομενο led
MOV D,A
JMP L1    ;Επιστροφη και ελεγχος αν αλλαξε το MSB

LEFT:
MOV A,D
CMA
STA 3000H
CMA
RLC         ;Αριστερή ολίσθηλη τώρα
MOV D,A
JMP L1    ;Επιστροφη και ελεγχος αν αλλαξε το MSB

END
```

Σημείωση : Παρ'όλο που η θεωρητική τιμή στους καταχωρητές B-C είναι σωστή, δηλαδή ισούται με $(500)_{10}$ κάποιες φορές στην προσωμοίωση η καθυστέρηση είναι μικρότερη του μισού second, οπότε αναγκαστήκαμε να αυξήσουμε την τιμή αυτή.

Άσκηση 3η

Ο τροποποιημένος κώδικας που επιτελεί τη ζητούμενη λειτουργία είναι ο ακόλουθος :

```
START:
LXI B,01F4H ;Αποθήκευση των 500 sec στους (BC)

LDA 2000H   ;Είσοδος δεδομένων στον Αποθήκευση
CPI 63H     ;Μεγαλύτερος του 99;
JNC GREATER ;Αν ναι μεταβαίνει στο GREATER, αλλιώς συνεχεια

MVI D,FFH   ;D <- Συμπληρωμα ως προς 2 του 1
DECA:
INR D       ;D++
SUI 0AH     ;A = A-10
JNC DECA    ;Αν δεν είναι <0 επαναλαμβάνεται

ADI 0AH     ;Αφου <0, αυξάνω τον A κατά 1
MOV E,A     ;Προσωρινή αποθήκευση, ο E έχει τις μονάδες
MOV D,A     ;Στον D βρίσκονται οι δεκάδες, μεταφέρονται στο A
RLC         ;4 αριστερές περιστροφές
RLC         ;ωστε τα 4 MSB του A να δείχνουν τις δεκάδες
RLC
RLC
ADD E       ;Προσθεση του E, ωστε τα 4 LSB να έχουν μονάδες
CMA
STA 3000H   ;Εμφάνιση στα LED
JMP START   ;Επανάληψη

GREATER:
MVI A,F0H  ;A <-(11110000), ωστε να αναψουν 4 LSB (αναστροφα LED)
STA 3000H
CALL DELB   ;Καθυστέρηση
CMA         ;A <- (00001111), ωστε να ανάψουν τα 4 MSB led
STA 3000H
CALL DELB
JMP START   ;Επιστροφή στην αρχή για νέα είσοδο
END
```

Άσκηση 4η

Τεχνολογία 1 : Διακριτά στοιχεία

Σε αυτή την τεχνολογία απαιτούνται 20.000 για την αρχική σχεδίαση, ενώ για την κατασκευή κάθε πλακέτας απαιντούνται 10 για τα I.C. και 10 για τη συναρμολόγηση.

Άρα η συνάρτηση κόστους/τεμάχιο είναι : ***Cost = 20.000 +20x***, όπου x ο αριθμός των τεμαχίων.
Αντίστοιχα, η συνάρτηση κόστους ανά τεμάχιο είναι : $\mathbf{K_1 = 20.000/x + 20}$

Τεχνολογία 2 : FPGAs

Σε αυτή την τεχνολογία απαιτούνται 10.000 για την αρχική σχεδίαση, ενώ για την κατασκευή κάθε πλακέτας απαιντούνται 10 για τα FPGA και 30 για τη συναρμολόγηση.

Άρα η συνάρτηση κόστους είναι : ***Cost =10.000 +40x***, όπου x ο αριθμός των τεμαχίων.
Αντίστοιχα, η συνάρτηση κόστους ανά τεμάχιο είναι : $\mathbf{K_2 = 10.000/x + 40}$

Τεχνολογία 3 : SoC-1 με μικρή πλακέτα

Σε αυτή την τεχνολογία απαιτούνται 100.000 για την αρχική σχεδίαση, ενώ για την κατασκευή κάθε πλακέτας απαιντούνται 2 για την πλακέτα και 2 για τη συναρμολόγηση.

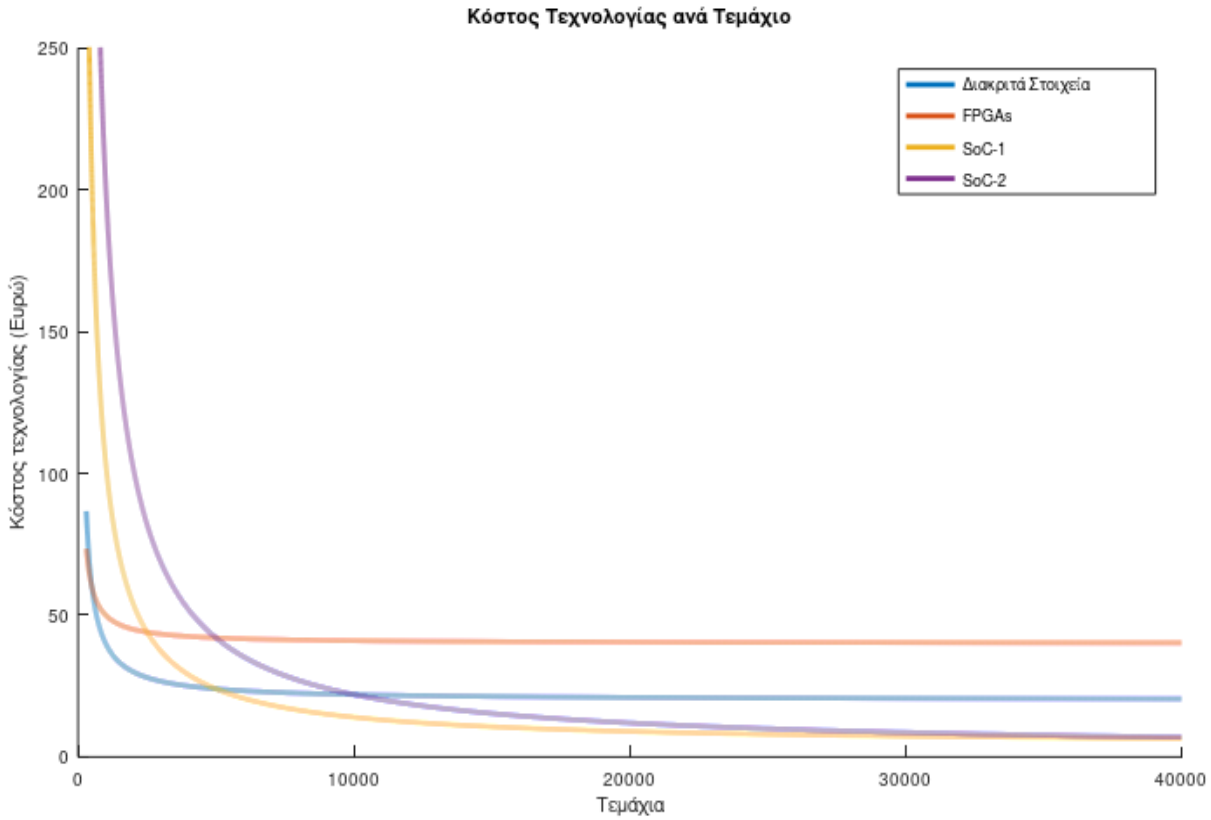
Άρα η συνάρτηση κόστους είναι : ***Cost =100.000 +4x***, όπου x ο αριθμός των τεμαχίων.
Αντίστοιχα, η συνάρτηση κόστους ανά τεμάχιο είναι : $\mathbf{K_3 = 100.000/x + 4}$

Τεχνολογία 4 : SoC-2 με πολύ μικρή πλακέτα

Σε αυτή την τεχνολογία απαιτούνται 200.000 για την αρχική σχεδίαση, ενώ για την κατασκευή κάθε πλακέτας απαιντούνται 1 για το I.C. και 1 για τη συναρμολόγηση.

Άρα η συνάρτηση κόστους είναι : ***Cost =200.000 +2x***, όπου x ο αριθμός των τεμαχίων.
Αντίστοιχα, η συνάρτηση κόστους ανά τεμάχιο είναι : $\mathbf{K_4 = 200.000/x + 2}$

Με βάση τα παραπάνω, σχεδιάζουμε τη γραφική παράσταση των καμπυλών κόστους ανά τεμάχιο για τις 4 τεχνολογίες. Η γραγική παράσταση εμφανίζεται παρακάτω :



Στη συνέχεια θα υπολογιστούν οι περιοχές τεμαχίων για τις οποίες είναι προτιμότερη η επιλογή τις κάθε τεχνολογίας. Γενικά παρατηρούμε πως όσο αυξάνονται τα τεμάχια, το αρχικό κόστος σχεδίασης «εξαφανίζεται», και το κόστος ανά τεμάχιο τείνει στο κόστος κατασκευής ενός τεμαχίου.

Τεχνολογίες 1 & 2 : Για μικρό αριθμό τεμαχίων, η τεχνολογία 1 είναι η οικονομικότερη, αφού έχει το μικρότερο κόστος σχεδίασης. Το σημείο που η τεχνολογία 2 γίνεται οικονομικότερη από την 1 είναι :

$$20000 + 20x_1 < 10000 + 40x_1 \Rightarrow \boldsymbol{x_1 > 500\ \tau\varepsilon\mu\acute{\alpha}\chi\iota\alpha}$$

Παρατηρόντας και το διάγραμμα παρατηρούμε πως μεταξύ των δύο, για **λιγότερα από 500 τεμάχια** συμφέρει η τεχνολογία **FPGAs**, ενώ για **πάνω από 500** τεμάχια συμφέρει η χρήση **διακριτών στοιχείων**.

Τεχνολογίες 1 & 3: Με την ίδια λογική, έχουμε το σημείο που η τεχνολογία 3 γίνεται οικονομικότερη από την 1 είναι:

$$100000 + 4x_2 < 20000 + 20x_2 \Rightarrow \boldsymbol{x_2 > 5000\ \tau\varepsilon\mu\acute{\alpha}\chi\iota\alpha}$$

Από και το διάγραμμα παρατηρούμε πως μεταξύ των δύο, για **λιγότερα από 5000 τεμάχια** συμφέρει η τεχνολογία **διακριτών στοιχείων**, ενώ για **πάνω από 5000** τεμάχια συμφέρει η χρήση **SoC-1**.

Τεχνολογίες 3 & 4: Με την ίδια λογική, έχουμε το σημείο που η τεχνολογία 4 γίνεται οικονομικότερη από την 3 είναι:

$$200000 + 2x_3 < 100000 + 4x_3 \Rightarrow \boldsymbol{x_3 > 50000\ \tau\varepsilon\mu\acute{\alpha}\chi\iota\alpha}$$

Από και το διάγραμμα παρατηρούμε πως μεταξύ των δύο, για **λιγότερα από 50000 τεμάχια** συμφέρει η τεχνολογία **SoC-1**, ενώ για **πάνω από 50000** τεμάχια συμφέρει η χρήση **SoC-2**.

Συνεπώς από τα παραπάνω παίρνουμε τα διαστήματα στα οποία είναι συμφέρουσα η χρήση της κάθε τεχνολογίας ως εξής:

- 0 < τεμάχια < 500 : FPGAs
- 500 < τεμάχια < 5000 : Διακριτά στοιχεία
- 5000 < τεμάχια < 50000 : SoC-1
- 50000 < τεμάχια : SoC-2

Στη συνέχεια θα εξεταστεί για ποια τιμή κατασκευής των I.C. για το FPGAs θα εξαφανίσει την επιλογή της πρώτης τεχνολογίας. Η τεχνολογία διακριτών στοιχείων είναι συμφέρουσα στο διάστημα μεταξύ 500 και 5000 τεμαχίων. Επομένως εάν σε αυτό το διάστημα γίνει πιο συμφέρουσα η επιλογή των FPGAs, δε θα υπάρχει λόγος επιλογής της πρώτης τεχνολογίας.

Επομένως, θέλουμε για 5000 τεμάχια να ισχύει:

$$20000 + 20 * 5000 \geq 10000 + (10 + x)5000 \Rightarrow \boldsymbol{x \leq 21,99\ \varepsilon\upsilon\rho\acute{\omega}}$$

Άρα, για κόστος κατασκευής τψν I.C.'s των FPGA ίσο με περίπου 22 ευρώ, εξαφανίζεται η επιλογή των διακριτών στοιχείων.

Άσκηση 5η

(i)

```
module A_3_20a(A,B,C,D,F);

output F;
input A,B,C,D;
wire w1,w2,w3,w4,w5;

not G1(w1, C);

and G2(w2, C, D);
or G3(w3, w2, B);
and G4(w4, w3, A);
and G5 (w5, w1, B);

or G6(F, w4, w50);

endmodule
```

```
module A_3_21b(A,B,C,D,F);

output F;
input A,B,C,D;
wire w1,w2,w3,w4,w5,Anot, Bnot, Cnot;

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


nand
    G4 (w1, A, Bnot),
    G5 (w2, Anot, B),
    G6 (w3, Cnot, D),
    G7 (w4, w1, w2),
    G8 (w5, w3, w4);

not G9(F, w5)
```

```
module A_3_24(A,B,C,D,E,F);

output F;
input A,B,C,D,E;
wire w1,w2,Enot;

not G1(Enot, E);

nor
    G2(w1, A, B),
    G3(w2, C, D),
    G4(F, w1, w2, Enot);

Endmodule
```

```
module A_3_25(A,B,C,D,F);

output F;
input A,B,C,D;
wire w1,w2,w3,w4,Anot,Bnot,Dnot;
not
    G1(Anot, A),
    G2(Bnot, B),
    G3(Dnot, D);
nor
    G4(w1, Anot, B),
    G5(w2, A, Bnot),
    G6(w3, C, Dnot),
    G7(w4, w1, w2),
    G8(F, w4,w3);

endmodule
```

(ii)

```
module A_3_20b(A,B,C,D,F);
output F;
input A,B,C,D;

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

endmodule
```

```
module A_3_21a(A,B,C,D,F);

output F;
input A,B,C,D;

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

endmodule
```

```
module A_3_24(A,B,C,D,E,F);

output F;
input A,B,C,D,E;

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

endmodule
```

```
module A_3_25(A,B,C,D,F);

output F;
input A,B,C,D,E;

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

endmodule
```

Άσκηση 6η

(i)

```
module A_4_36(D,x,y,V);

output x,y,V;
input [0:3] D;
wire w1,w2;

not G1(w1,D[2]);
and G2(w2, w1, D[1]);

or
    G3(y, D[3], W2),
    G4(x, D[2], D[3]),
    G5(V, x, D[1], D[0]);

endmodule
```

(ii)

```
module A_4_45(D,x,y,V);

output [0:1] out, V;
input [0:3] D;
reg [0:1] out, V;

always @(D)
    begin
        if(D[0])
            begin
                out[0]=1;
                out[1]=1;
            end

        else if(D[1])
            begin
                out[0]=1;
                out[1]=0;
            end

        else if(D[2])
            begin
                out[0]=0;
                out[1]=1;
            end

        else if(D[3])
            begin
                out[0]=0;
                out[1]=0;
            end

        if(D) V=1;
        else  V=0;
    end

endmodule
```