1η ΟΜΑΔΑ ΑΣΚΗΣΕΩΝΜΙΚΡΟ-ΥΠΟΛΟΓΙΣΤΩΝ
ΓΙΩΡΓΟΣ ΜΥΤΙΛΗΝΑΙΟΣ 03119841
ΚΩΣΤΗΣ ΙΩΑΝΝΟΥ el19840

1η Άσκηση

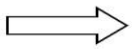
Η διαδικασία αποκωδικοποίηση γίνεται με χρήση του πίνακα 2 του παραρτήματος 2 των σημειώσεων.

Διεύθυνση	Περιεχόμενο	Ετικέτα	Εντολή
0800	06	START	MVI B,01H
0801	01		
0802	3A		LDA 2000H
0803	00		
0804	20		
0805	FE		CPI 00H
0806	00		
0807	CA		JZ L3
0808	13		
0809	08		
080A	1F	L1	RAR
080B	DA		JC L2
080C	12		
080D	08		
080E	04		INR B
080F	C2		JNZ L1
0810	0A		
0811	08		
0812	78	L2	MOV A,B
0813	2F	L3	CMA
0814	32		STA 3000H
0815	00		
0816	30		
0817	CF		RST 1

Όπως μπορούμε να καταλάβουμε το πρόγραμμα παίρνει μια δυαδική τιμή από την θέση 2000H(είσοδος) της μνήμης όπου βρίσκονται οι διακόπτες κι εμφανίζει στα LED(έξοδος 3000H) την θέση του άσου που βρίσκεται δεξιότερα με έναν δυαδικό αριθμό. Για παράδειγμα
Με είσοδο 1110 0010 παίρνουμε έξοδο 0000 0010.

Αφου τρέξουμε τον κώδικα διαπιστώνουμε ότι δεν βρίσκεται σε συνεχή λειτουργία ,δηλαδή μετά από ένα παράδειγμα τα LED που άναψαν παραμένουν ανάμενα χωρίς να επηρεάζονται από την είσοδο.

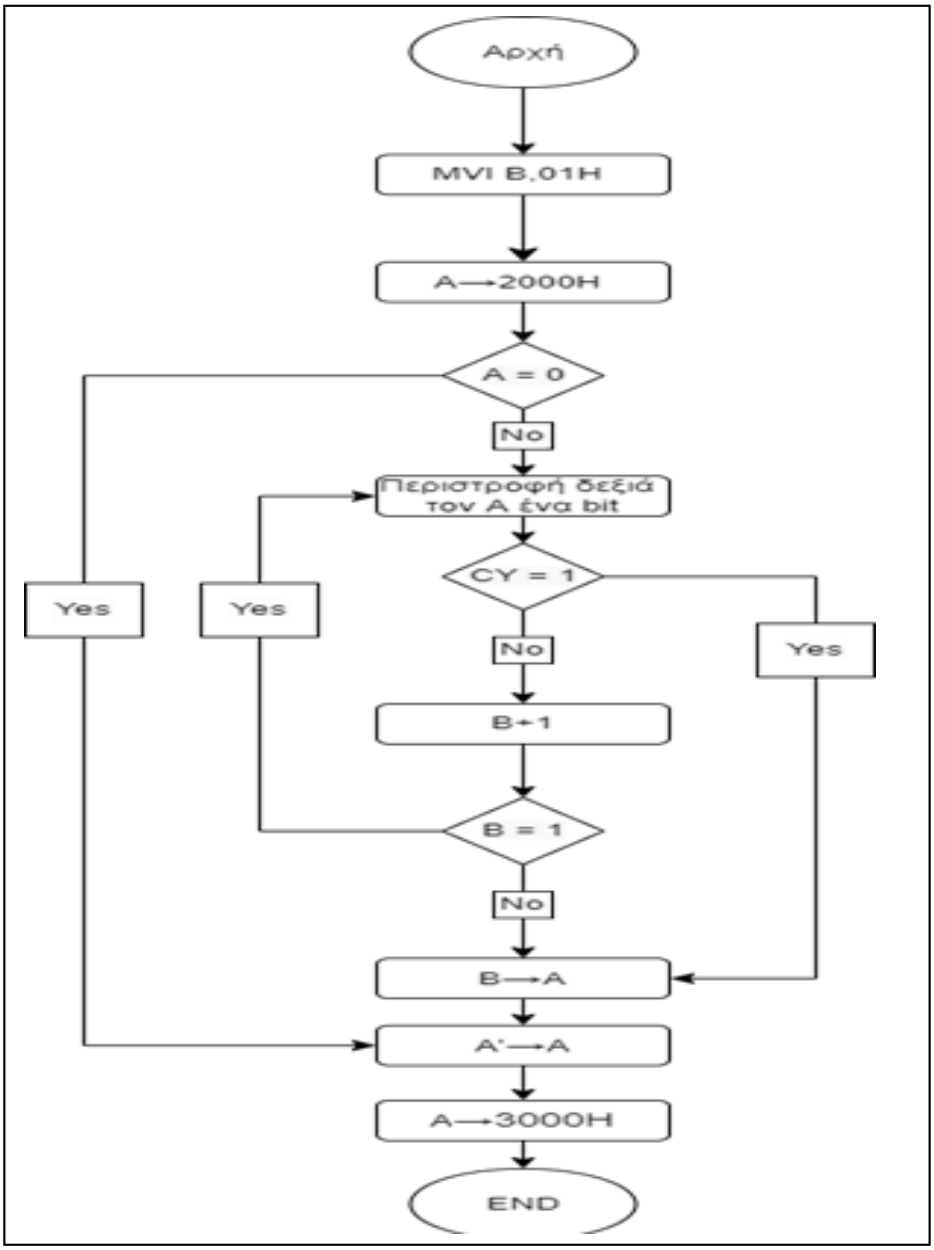
```
 06 01
3A 00 20
FE 00
CA 13 08
1F
DA 12 08
04
C2 0A 08
78
2F
32 00 30
```



```
START:
    MVI B,01H
    LDA 2000H
    CPI 00H
    JZ L3
L1:
    RAR
    JC L2
    INR B
    JNZ L1
L2:
    MOV A,B
L3:
    CMA
    STA 3000H
    RST 1
```

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΣΥΝΕΧΗ – ΛΕΙΤΟΥΡΓΙΑ

Για να κάνει συνεχή λειτουργία το πρόγραμμα θα πρέπει να βάλουμε μια εντολή jump στο τέλος ώστε κάθε φορά που τελειώνει να πηγαίνει πάλι στην αρχή κι να ελέγχει τους διακόπτες ώστε κάθε φορά που αλλάζουμε θέση σε διακόπτη να το βλέπει.




```
START:
   MVI  B,01H
   LDA  2000H
   CPI  00H
   JZ   L3
L1:
   RAR
   JC   L2
   INR  B
   JNZ  L1
L2:
   MOV  A,B
L3:
   CMA
   STA  3000H
   JMP  START
   RST  1
END
```

2η Άσκηση

Είσοδος 2000H και έξοδος 3000H,το πρόγραμμα μας θα κοιτάζει τα 2 πιο ασήμαντα ψηφιά της εισόδου (1ο και 2ο LSB) και με βάση αυτά θα εκτελείτε όπως αναφέρεται στην εκφώνηση.

(2 τελευταία Ψηφία)

01 :Αναμένο LED μετακινείτε μια θέση αριστερά μέχρι να βρεθεί στην αριστερότερη θέση τότε μετακινητέ μια θέση προς τα δεξιά μέχρι να βρεθεί στην δεξιότερη θέση τότε αλλάζει πορεία προς τα αριστερά και ούτω καθεξής.

00 :Αναμένο LED μετακινείτε μια θέση αριστερά συνεχώς μέχρι να αλλάξει η είσοδος.

10 ή 11:Το LED παραμένει αναμμένο στο σημείο που βρισκόταν τι στιγμή που μεταβήκαμε σε αυτήν την κατάσταση και όταν όταν πάει πάει σε μια από τις προηγούμενες καταστάσεις συνεχίζει να κινείτε με με την κατάλληλη κατεύθυνσή. Συνεπώς προς τα αριστερά με είσοδο 00 και συνεχίζει είτε με φορά προς τα δεξιά είτε με φορά προς τα αριστερά ανάλογα με την κατεύθυνση που είχε πριν σταματήσει για είσοδο 01

(Ο κώδικας υπάρχει και σε μορφή file.8085)

```
IN 10H        ;Επιτρέπει την πρόσβαση σε όλη την RAM
LXI B,01F4H   ;Χρησιμοποιείτε για την καθυστέρηση
              ;BC=data(01F4)
MVI E,FEH     ;Θέση που αρχίζει το led ,πρώτο led
                ;E=data(FEH)
START:
LDA 2000H      ;A =data(2000) το A παίρνει την είσοδο
RRC            ;Μετακίνηση Δεξια
JC START_LEFT
RRC            ;Μετακίνηση Δεξιά
JC START       ;Αν cy=1 Jump
JMP CIRCLE

START_LEFT:
LDA 2000H      ;A=data(2000) το A παίρνει την είσοδο
RRC            ;Μετακίνηση Δεξιά
JNC CIRCLE     ;Αν cy=0 Jump
RRC            ;Μετακίνηση Δεξιά
JC START_LEFT  ;Αν cy=1 Jump
JMP LEFT

START_RIGHT:
LDA 2000H       ;o A παίρνει την είσοδο
RRC             ;Μετακίνηση Δεξια
JNC CIRCLE
RRC             ;Μετακίνηση Δεξιά
JC START_RIGHT  ;Αν έχει κρατούμενο jump
JMP RIGH
```

```
LEFT:
CALL DELB  ;Καθυστέρηση  0.5 sec
MOV A,E    ; A=data(E) το Α παίρνει την προηγούμενη του θέση
STA 3000H ; data(3000) = A , LED ON στην έξοδο στην θέση που
είναι ο Α
RLC        ;Μετακίνηση Αριστερά
JNC RIGHT  ;Αν δεν έχει κρατούμενο jump
MOV E,A    ;E= data(A) κρατάμε την τρέχουσα θέση του Α
JMP START_LEFT

RIGHT:
CALL DELB  ;Καθυστέρηση 0.5sec
MOV A,E    ;A=data(E) ο Α παίρνει την προηγουμένη του θέση
STA 3000H  ;LED ON στην έξοδο στην θέση που ειναι ο Α
RRC        ;Μετακίνηση Δεξιά
JNC LEFT   ;Αν cy=0 jump
MOV E,A    ;E=data(A) αποθήκευσε την τρέχουσα θέση του Α
JMP START_RIGHT

CIRCLE:
CALL DELB  ;Καθυστέρηση 0.5 sec
MOV A,E ;A=data(E) ο Α παίρνει την προηγούμενη του θέση
STA 3000H ;data(3000)=A LED ON στην έξοδο στην θέση του Α
RLC  ;Μετακίνηση Αριστερά
MOV E,A  ;E=data(A) αποθηκεύουμε την τρέχουσα θέση του Α
JMP START

END
```

3η Άσκηση (Ο κώδικας υπάρχει καισε μορφή file.8085)

```
START:
MVI D,FFH   ;D = 1111 1111 , -1 με συμπλήρωμα ως προς 2
LDA 2000H   ;A = data(2000H) περνάμε την είσοδο στο Α
CPI 63H     ;cy=1 if A<(99)dec else cy=0
JNC HUN     ;If A>(99)dec jump

DECA:
INR D    ;D-- , Κρατάμε τις δεκάδες
SUI 0AH  ;A = A -(10)dec
JNC DECA ;if A<0 συνέχισε τις αφαιρέσεις
ADI 0AH  ;A=A +10 ,αλλιώς διόρθωσε το αρνητικό υπόλοιπο
MOV C,A  ; C =A κρατάμε την τιμή του Α με τα LSB
MOV A,D  ; A =D = Δεκάδες
RLC      ;Μετακινούμαστε 4 φορές αριστερά ώστε να έχουμε
RLC      ;κρατήσουμε στα 4MSB την δεκάδα και μηδενίζουμε
RLC      ;τα 4LSB
RLC
ADD C      ;A = A + C ,περνάμε στα 4lSB τις μονάδες κρατήσαμε
CMA        ; Αντιστρέφουμε την λογική ώστε 1->ON 0->off
```

```
STA 3000H ;έξοδος data(3000H)= A = 4MSB 4LSB =
Δεκάδες Μονάδες
JMP START

HUN:
CPI C7H   ;if A <(199)dec cy=1 else cy=0

JNC ALARM ;if A>199 jump to ALARM
SUI 64H   ;A = A -(100)dec
JMP DECA  ;jumb to DECA


ALARM:
LXI B,0063H ;B = (100)dec

MVI A,F0H   ;A = (1111 0000)bin
STA 3000H   ;MSB off LSB ON

CALL DELB   ;delay
MVI A,FFH   ; A =(1111 1111)bin
STA 3000H   ;MSB off LSB off

CALL DELB   ;delay
            ;Επαναλαμβάνουμε  το ίδιο μοτίβο για κάποιες φορές
CALL DELB
MVI A,F0H
STA 3000H

CALL DELB
MVI A,FFH
STA 3000H

CALL DELB
MVI A,F0H
STA 3000H

CALL DELB
MVI A,FFH
STA 3000H


CALL DELB
MVI A,F0H
STA 3000H

CALL DELB
MVI A,FFH
STA 3000H

CALL DELB
MVI A,F0H
STA 3000H
CALL DEL

JMP START ;Βγαίνουμε  απο το ALARM jump στο START

END:
HLT
END
```

Άσκηση 4 (Φορητή ηλεκτρονική συσκευή)
Έχουμε 4 τεχνολογίες (όπου Χ=Πλήθος των τεμαχίων)

1) Χρήση διακριτών στοιχείων και I.C. όπως μικροελεγκτών, περιφερειακών, μνημών κλπ. Τοποθετημένα σε μια σε μια σχετικά μεγάλη πλακέτα.
   - Κόστος σχεδίασης :15.000€
   - IC ανα Τεμάχιο:10€
   - Συναρμολόγηση ανα Τεμάχιο:10€

   ➔ K1=15.000+(10+10)*X =15.000+20*X

2) Χρήση FPGAs και μικρού αριθμού περιφερειακών τοποθετημένα σε μια σε μια πλακέτα.
   - Κόστος σχεδίασης :7.000€
   - IC ανα Τεμάχιο:50€
   - Συναρμολόγηση ανα Τεμάχιο:10€

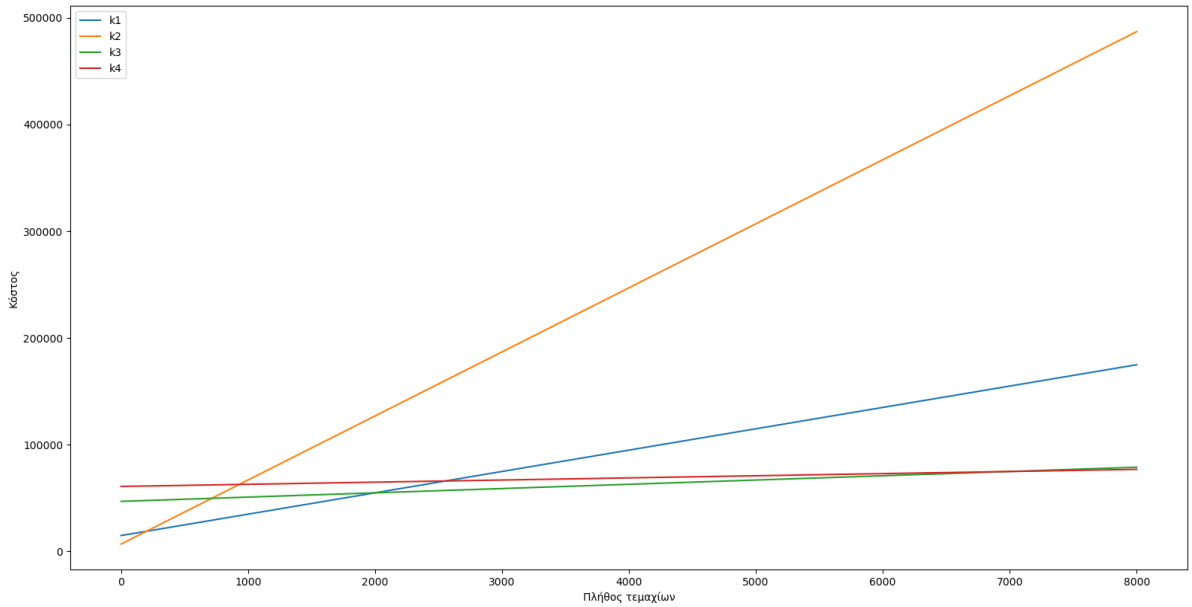
   ➔ K2=7.000+(50+10)*X =7.000+60*X

3) Σχεδίαση ειδικού SoC-1 με μια μικρή πλακέτα.
   - Κόστος σχεδίασης :47.000€
   - IC ανα Τεμάχιο:2€
   - Συναρμολόγηση ανα Τεμάχιο:2€

   ➔ K3=47.000+(2+2)*X =47.000+4*X

4) Σχεδίαση ειδικού SoC-2 με μια πολύ πιο μικρή πλακέτα
   - Κόστος σχεδίασης :61.000€
   - IC ανα Τεμάχιο:1€
   - Συναρμολόγηση ανα Τεμάχιο:1€

   ➔ K4=61.000+(1+1)*X =61.000+2*X



Θα βρούμε ανάλογα με το πλήθος των τεμαχίων που παράγουμε ποια τεχνολογία μας συμφέρει για να έχουμε ελάχιστο κόστος.

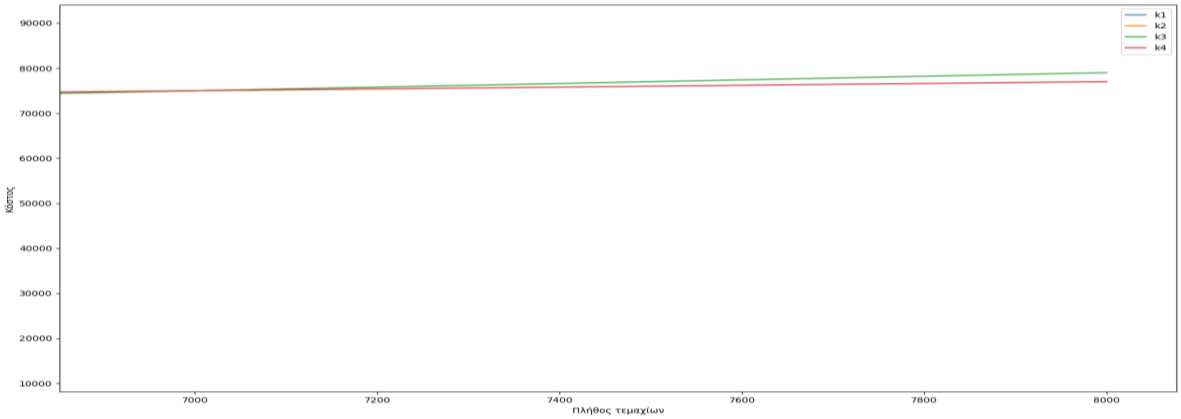
$0 \leq X < 200$: Μας συμφέρει η τεχνολογία 2 (K2)

$200 \leq X < 2000$: Μας συμφέρει η τεχνολογία 1 (K1)

$2000 \leq X < 7000$: Μας συμφέρει η τεχνολογία 3(K3)

$7000 \leq X$: Μας συμφέρει η τεχνολογία 4 (K4)

Η K3 αυξάνει πιο γρήγορα από την K4 αλλά και οι δύο αυξάνουν με πολύ μικρό ρυθμό και καθώς η K4 αρχίζει από 61.000 ενώ η K3 αρχίζει από 47.000 θα αργήσει να περάσει εμφανώς την K4 όπως φαίνεται στο διάγραμμα.



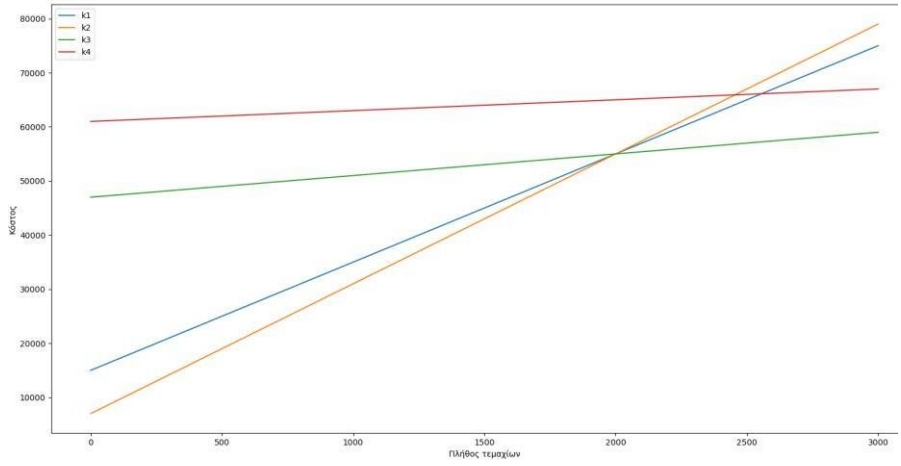
Έστω IC ανα Τεμάχιο στην τεχνολογία δύο κοστίζει α € , πρέπει να βρούμε για ποια α το κόστος της τεχνολογίας δύο είναι πάντα μικρότερο από το κόστος της τεχνολογίας ένα.

$k2 \leq k1 \ για \ άθε \ X(ήθος \ τεαχίω) < -> 7000 + (\alpha + 10) * X \leq 15000 + 20 * X$

$\alpha \leq \frac{8000}{X} + 10$: όμως όπως είδαμε K1 είναι μικρότερο του K2 για $200 \leq X < 2000$

Όπου το δεξί μέλος της εξίσωσης πάει από 50 εώς 14 .Άρα $\boldsymbol{\alpha \leq 14€}$

Για α =14€ (από 50€) έχουμε:



Πλέον δηλαδή

$0 \leq X < 2000$: Μας συμφέρει η τεχνολογία 2 (K2)

$2000 \leq X < 7000$: Μας συμφέρει η τεχνολογία 3(K3)

$7000 \leq X$: Μας συμφέρει η τεχνολογία(K4)

Άρα η τεχνολογία K1 δεν μας συμφέρει ποτέ μας είναι αδιάφορη.