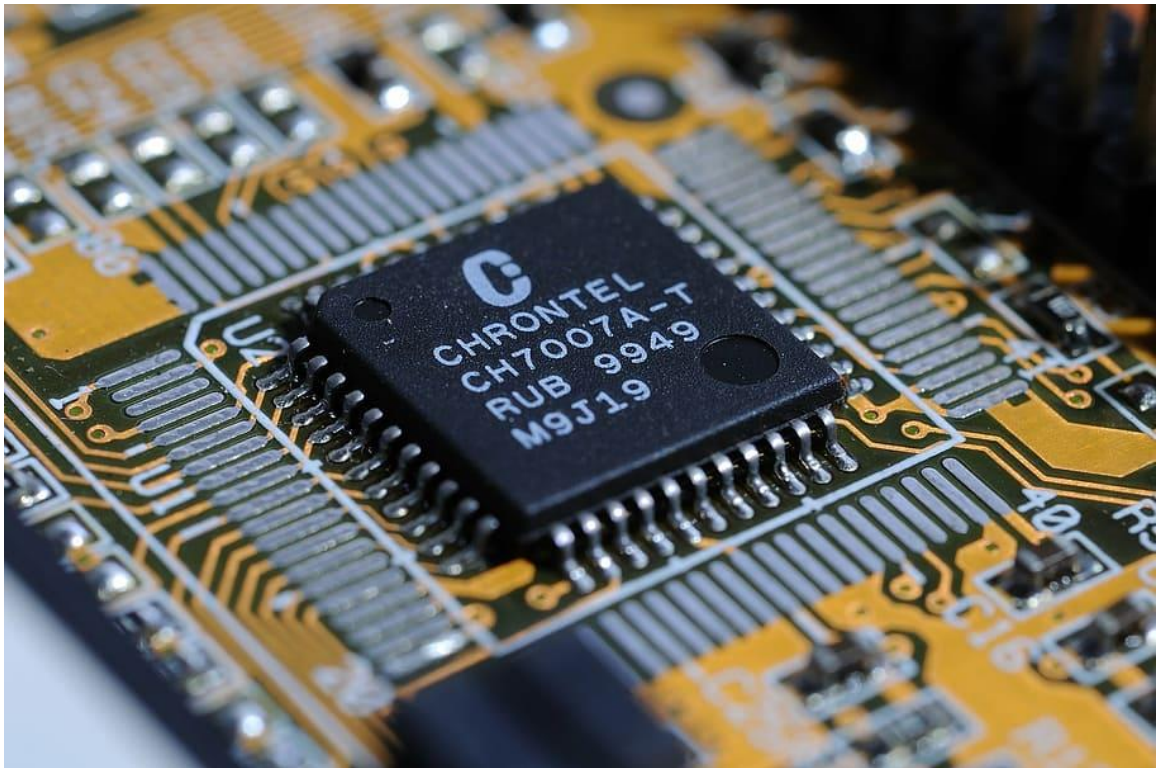


Συστήματα Μικροϋπολογιστών

1Η ΟΜΑΔΑ ΑΣΚΗΣΕΩΝ

Καμπουγέρης Χαράλαμπος: el20098
Κουστένης Χρίστος: el20227
Ακαδημαϊκό Έτος 2022-2023

1η Άσκηση (Προσομοίωσης)

Αξιοποιώντας τον πίνακα 2 σελ. 98-99 του αρχείου «Εισαγωγή στο TSIK» πετύχαμε την μετατροπή της γλώσσας μηχανής που μας δίνεται στις αντίστοιχες εντολές assembly του 8085. Ακολουθεί η αντιστοίχιση που πραγματοποιήθηκε:

Assembly	Machine Language
MVI C,08H	**0E** 08
LDA 2000H	**3A** 00 20
RAL	**17**
JC 080DH	**DA** 0D 08
DCR	**0D**
JNZ 0805H	**C2** 05 08
MOV A,C	**79**
CMA	**2F**
STA 3000H	**32** 00 30
RST 1	**CF**

Για να μπορέσει το πρόγραμμα να μεταγλωττιστέι και να τρέξει επιτυχώς στον προσομοιωτή πρέπει να αντικαταστήσουμε τις διευθύνσεις στις εντολές άλματος με ετικέτες. Επομένως το πρόγραμμα τρέχει επιτυχώς στην ακόλουθη τελική του μορφή όπου φαίνονται και οι αντίστοιχες θέσεις μνήμης που καταλαμβάνει κάθε εντολή.

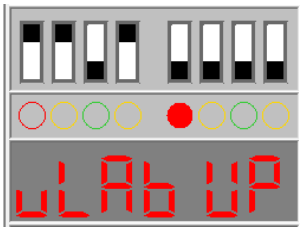
```
0800   0E     MVI C,08H
0801   08
0802   3A     LDA 2000H
0803   00
0804   20

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

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

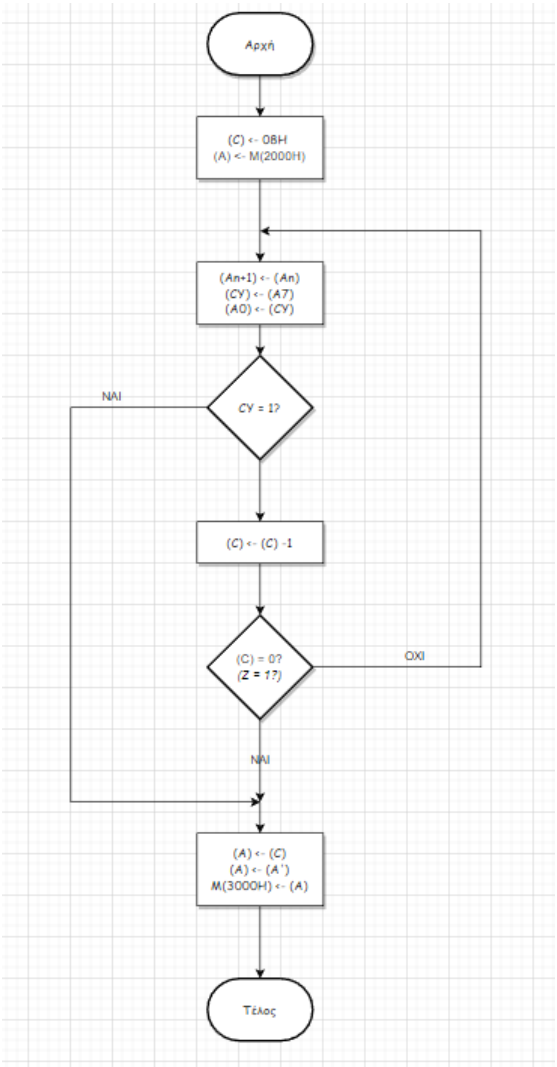
Διατρέχοντας μία προς μία τις εντολές το πρόγραμμα αυτό μεταφέρει στον καταχωρητή C τον αριθμό 8_{10} = $(00000100)_2$ με την εντολή MVI C,08H και τα δεδομένα της εισόδου που καθορίζονται από την κατάσταση των switches μεταφέρονται στον καταχωρητή A με την εντολή LDA 2000H. Στη συνέχεια, η εντολή RAL μετατοπίζει προς τα αριστερά το περιχόμενο του καταχωρητή A και η τιμή της σημαίας CY γίνεται (CY) <- A_7 . Έπειτα με την εντολή άλματος JC SAVE αν η σημαία του κρατούμενου έχει γίνει 1 απο την προηγούμενη εντολή τότε ο PC παίρνει την τιμή της μνήμης όπου βρίσκεται η ομώνυμη με το δεύτερο μέλος της εντολής ετικέτα. Στο ενδεχόμενο αυτό μεταφέρεται στον καταχωρητή C το περιεχόμενο του A με την εντολή MOV A,C και η εντολή CMA που ακολουθεί υπολογίζει το συμπλήρωμα του περιεχομένου του συσσωρευτή και το τοποθετεί σε αυτόν. Τέλος, για να εκφραστεί, μέσω των LEDs, το περιεχόμενο του A στην έξοδο το αποθηκεύουμε στην θέση μνήμης 3000H.

Μακροσκοπικά, η λειτουργία του προγράμματος αυτού είναι να λαμβάνει ως δεδομένο εισόδου τον μεγαλύτερο αριθμό, αριθμώντας από δεξιά προς τα αριστερά, του switch που είναι ON και να τον επιστρέφει στην έξοδο μέσω των LEDs σε δυαδική μορφή. Τα LEDs είναι επίσης αριθμημένα από δεξία προς τα αριστερά. Δηλαδή, το MSB είναι το αριστέροτερο LED. Για παράδειγμα, δίνεται η παρακάτω εικόνα που προκύπτει αφού μεταγλωττιστεί το πρόγραμμα και το τρέξουμε έχοντας κάνει FETCH ADDRESS στην θέση o800H της RAM και εκτελώντας με την RUN το πρόγραμμα.



Το μεγαλύτερο Switch σε θέση ON είναι το 8ο. Άρα τα LEDs απεικονίζουν τον αριθμό 8_{10} = 00001000_2 .

Ακολουθεί το διάγραμμα ροής του παραπάνω προγράμματος :



Το πρόγραμμα περιλαμβάνεται στο αρχείο *1st_ex.8085*

Για να επαναλαμβάνεται χωρίς τέλος το πρόγραμμα και τα LEDs να αλλάζουν δυναμικά φωτισμό με βάση τις αλλαγές που κάνουμε στα switches αφαιρούμε την εντολή RST 1 που προκαλεί επιστροφή στο monitor πρόγραμμα και την αντικαθιστούμε με την εντολή JMP START όπου το START είναι ένα label που τοποθετήσαμε στην αρχή του προγράμματος. Έτσι το τελικό πρόγραμμα γίνεται ως εξής:

```
START:

MVI C,08H ; Moves the number 8 into the C register
LDA 2000H ; Loads to the register A the content of the address 2000

RETRY:

RAL ;       Left rotation of the content of the A register
JC SAVE ;   If CY = 1 then go to SAVE label else continue to DCR
DCR C   ;   (C) <- (C)-1
JNZ RETRY ; If Z =! 0 then go to label RETRY

SAVE:

MOV A,C ;     (A) <- (C)
CMA ;         (A) <- (A')
STA 3000H  ; The content of A register goes to the 3000H memory slot
JMP START ;  Return to START

END
```

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

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

SAVE:
080D   79     MOV A,C
080E   2F     CMA
080F   32     STA 3000H
0810   00
0811   30
0812   C3     JMP START
0813   00
0814   08
```

Το παραπάνω πρόγραμμα περιλαμβάνεται στο αρχείο 1st_ex_looping_forever.8085

2η Άσκηση (Προσομοίωσης)

Στις παρακάτω εικόνες φαίνεται το πρόγραμμα που εκτελεί τη λειτουργία που περιγράφεται στην εκφώνηση. Επίσης, το πρόγραμμα συμπεριλαμβάνεται στο αρχείο *2nd_ex.8085*.

```
 IN 10H
 LXI B,01F4H  ; Insert a delay of 500ms
 MVI D,FEH    ; Startpoint of the lighted LED is the rightmost LED

START:

 LDA 2000H ; Put the input to accumulator
 RRC       ; Right rotation : (An)<-(An+1), (CY)<-(A7), (A0)<-(CY)

 JNC START ; If (CY)=0 go to START

 RLC    ;  Left rotation : (An+1)<-(An), (A0)<-(A7), (CY)<-(A7) to get to the initial state
 RLC    ;  Left rotation : (An+1)<-(An), (A0)<-(A7), (CY)<-(A7) to snatch the MSB through CY

 JC RIGHT_LEFT   ; Do the RIGHT to LEFT
 JMP LEFT_RIGHT  ; DO the LEFT to RIGHT

RIGHT_LEFT:

 MOV A,D   ; Transfer the previous state to the accumulator from D
 STA 3000H ; Store to the output address of LEDs
 RLC       ; Left rotation : (An+1)<-(An), (A0)<-(A7), (CY)<-(A7)
 CALL DELB ; Triggers the delay of 500ms
 MOV D,A   ; Saves the position of the ON LED
 JMP START ; Repeat the whole check of input

LEFT_RIGHT:

 MOV A,D    ; Transfer the previous state to the accumulator from D
 STA 3000H  ; Store to the output address of LEDs
 RRC        ; Right rotation : (An)<-(An+1), (CY)<-(A7), (A0)<-(CY)
 CALL DELB  ; Triggers the delay of 500ms
 MOV D,A    ; Saves the position of the ON LED
 JMP START  ; Repeat the whole check of input

 END
```

```
0800   DB     IN 10H
0801   10
0802   01     LXI B,01F4H
0803   F4
0804   01
0805   16     MVI D,FEH
0806   FE

START:
0807   3A     LDA 2000H
0808   00
0809   20
080A   0F     RRC
080B   D2     JNC START
080C   07
080D   08
080E   07     RLC
080F   07     RLC
0810   DA     JC RIGHT_LEFT
0811   16
0812   08
0813   C3     JMP LEFT_RIGHT
0814   22
0815   08

RIGHT_LEFT:
0816   7A     MOV A,D
0817   32     STA 3000H
0818   00
0819   30
081A   07     RLC
081B   CD     CALL DELB
081C   30
081D   04
081E   57     MOV D,A
081F   C3     JMP START
0820   07
0821   08

LEFT_RIGHT:
0822   7A     MOV A,D
0823   32     STA 3000H
0824   00
0825   30
0826   0F     RRC
0827   CD     CALL DELB
0828   30
0829   04
082A   57     MOV D,A
082B   C3     JMP START
082C   07
082D   08
```

3η Άσκηση (Προσομοίωσης)

Στις παρακάτω εικόνες φαίνεται το πρόγραμμα που εκτελεί τη λειτουργία που περιγράφεται στην εκφώνηση. Επίσης, το πρόγραμμα συμπεριλαμβάνεται στο αρχείο *3rd_ex.8085*.

```
        LXI B,01F4H         ;insert 500ms delay to B
START:
        LDA 2000H           ;load switches to A
        CPI C8H             ;compare A to 200
        JNC TOOBIG          ;if A > 199 goto TOOBIG
        CPI 64H             ;compare A to 100
        JNC BIG             ;if A > 99 goto BIG
        MVI D,FFH           ;initial value of counter
DECA:
        INR D               ;D = D + 1
        SUI 0AH             ;A = A - 10
        JNC DECA            ;if A > 0 continue
        ADI 0AH             ;if A < 0 correct negative
        MOV E,A             ;load units (A) to E
        MOV A,D             ;load decimal units (D) to A
        RLC                 ;move decimal units 4 bits to left
        RLC
        RLC
        RLC
        ADD E               ;add units to decimal
        CMA                 ;reverse A for LEDs
        STA 3000H           ;load value A to LEDs
        JMP START
BIG:
        MVI A,F0H           ;load value of 4 LSB LEDs to A
        STA 3000H           ;load A to LEDs
        CALL DELB           ;delay 500ms
        MVI A,FFH           ;load value of blank LEDs to A
        STA 3000H           ;load A to LEDs
        CALL DELB           ;delay 500ms
        JMP START           ;start over
TOOBIG:
        MVI A,0FH           ;load value of 4 MSB LEDs to A
        STA 3000H           ;load A to LEDs
        CALL DELB           ;delay 500ms
        MVI A,FFH           ;load value of blank LEDs to A
        STA 3000H           ;load A to LEDs
        CALL DELB           ;delay 500ms
        JMP START           ;start over

        END
```

```
0800   01     LXI B,01F4H
0801   F4
0802   01

START:
0803   3A     LDA 2000H
0804   00
0805   20
0806   FE     CPI C8H
0807   C8
0808   D2     JNC TOOBIG
0809   3B
080A   08
080B   FE     CPI 64H
080C   64
080D   D2     JNC BIG
080E   28
080F   08
0810   16     MVI D,FFH
0811   FF

DECA:
0812   14     INR D
0813   D6     SUI 0AH
0814   0A
0815   D2     JNC DECA
0816   12
0817   08
0818   C6     ADI 0AH
0819   0A
081A   5F     MOV E,A
081B   7A     MOV A,D
081C   07     RLC
081D   07     RLC
081E   07     RLC
081F   07     RLC
0820   83     ADD E
0821   2F     CMA
0822   32     STA 3000H
0823   00
0824   30
0825   C3     JMP START
0826   03
0827   08

BIG:
0828   3E     MVI A,F0H
0829   F0
082A   32     STA 3000H
082B   00
082C   30
082D   CD     CALL DELB
082E   30
082F   04
0830   3E     MVI A,FFH
0831   FF
0832   32     STA 3000H
0833   00
0834   30
0835   CD     CALL DELB
0836   30
0837   04
0838   C3     JMP START
0839   03
083A   08

TOOBIG:
083B   3E     MVI A,0FH
083C   0F
083D   32     STA 3000H
083E   00
083F   30
0840   CD     CALL DELB
0841   30
0842   04
0843   3E     MVI A,FFH
0844   FF
```

4η Άσκηση (Θεωρητική)

Με βάση τα δεδομένα, εξάγουμε τις σχέσεις κόστους ανά τεμάχιο και έχουμε τις παρακάτω συναρτήσεις κόστους (Ki (x)) ανά τεμάχιο x:

$$K_{1(x)} = \frac{20000 + 20x}{x}$$

$$K_{2(x)} = \frac{10000 + 40x}{x}$$

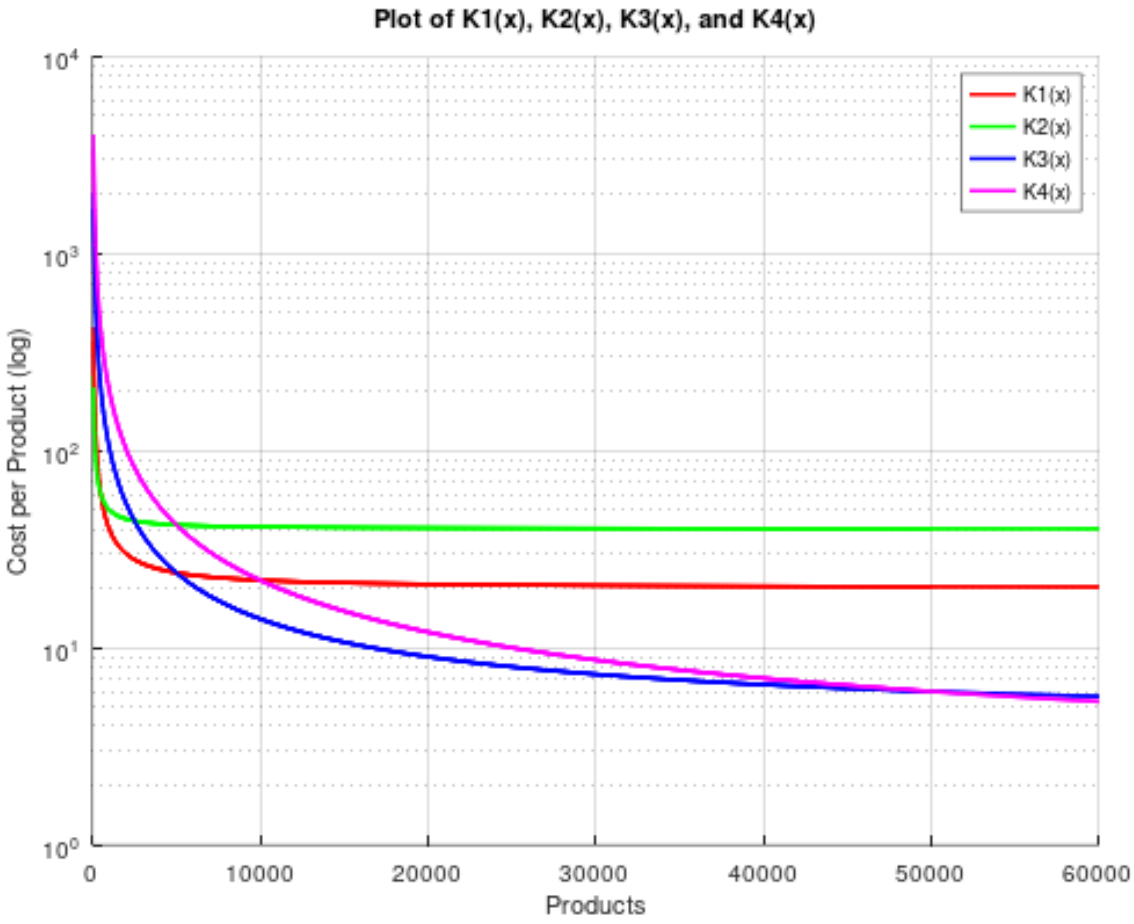
$$K_{3(x)} = \frac{100000 + 4x}{x}$$

$$K_{4(x)} = \frac{200000 + 2x}{x}$$

Όπου :

- K1 η 1η τεχνολογία I.C.
- K2 η 2η τεχνολογία FPGA
- K3 η 3η τεχνολογία Soc-1
- K4 η 4η τεχνολογία Soc-2

Οι γραφικές παραστάσεις και ο κώδικας για την υλοποίηση τους (Octave) φαίνεται παρακάτω:




```
% Define the functions
K1 = @(x) (20000 + 20.*x) ./ x;
K2 = @(x) (10000 + 40.*x) ./ x;
K3 = @(x) (100000 + 4.*x) ./ x;
K4 = @(x) (200000 + 2.*x) ./ x;

% Create a vector of x-values
x = linspace(0, 60000, 1000);

% Evaluate the functions over the range of x-values
y1 = K1(x);
y2 = K2(x);
y3 = K3(x);
y4 = K4(x);

% Plot the functions on the same graph
hold on;
semilogy(x, y1, 'r', 'LineWidth', 1.2);
semilogy(x, y2, 'g', 'LineWidth', 1.2);
semilogy(x, y3, 'b', 'LineWidth', 1.2);
semilogy(x, y4, 'm', 'LineWidth', 1.2);
hold off;

legend('K1(x)', 'K2(x)', 'K3(x)', 'K4(x)');
xlabel('x');
ylabel('y');
title('Plot of K1(x), K2(x), K3(x), and K4(x)');
axis([0, 60000, 1, 1e6]);
grid on;

xlabel('Products');
ylabel('Cost per Product (log)');
```

Στη συνέχεια εξισώνουμε κάθε συνάρτηση κόστους σε ζεύγη και λαμβάνουμε τα εξής σημεία τομής των καμπυλών:

Ζεύγη Συναρτήσεων Ki	Σημείο Τομής (Άξονα x)
K1-K2	500
K1-K3	5000
K1-K4	10000
K2-K3	2500
K2-K4	5000
K3-K4	50000

Έτσι με τη βοήθεια του γραφήματος και των παραπάνω, βρίσκουμε ποια τεχνολογία έχει τα χαμηλότερα κόστη για τα διάφορα εύρη τιμών του x.

x	Συμφέρουσα Τεχνολογία
(0,500)	2 (FPGA)
(500,5000)	1 (I.C.)
(5000,50000)	3 (SoC-1)
(50000,+∞)	4 (SoC-2)

Προκειμένου να εξαφανιστεί η επιλογή της πρώτης τεχνολογίας θα πρέπει, για το νέο κόστος (έστω λ) ανά I.C. της τεχνολογίας των FPGAs, να ισχύει:

$$K_2'(x) = \frac{10000 + (10 + \lambda)x}{x} < \frac{20000 + 20x}{x} = K_1(x) \Longrightarrow \lambda < \frac{1000}{x} + 10$$

Επομένως πρέπει $\lambda - 10 < 0 \Longrightarrow \lambda < 10$