Συστήματα Μικροϋπολογιστών

6ο Εξάμηνο

4η Ομάδα Ασκήσεων

Χαρδούβελης Γεώργιος-Ορέστης Κυριάκου Αθηνά

el15100 el17405

6ο Εξάμηνο 6ο Εξάμηνο

**Άσκηση 1**

Παρακάτω έχουμε το πρόγραμμα assembly που ελέγχει μέσω της διακοπής τύπου RST 6.5 τα φώτα ενός χώρου. Όταν προκαλείται διακοπή τύπου RST 6.5 (πάτημα πλήκτρου INTR) να αναβοσβήνουν (με περίοδο ~1/5 sec) όλα τα LED της πόρτας εξόδου. Αυτό να παραμένει για περίπου ένα (1) λεπτό της ώρας και μετά να σβήνει. Αν όμως ενδιάμεσα ξαναενεργοποιηθεί η διακοπή να ανανεώνεται ο χρόνος του ενός λεπτού. Ο χρόνος που παραμένει στην κατάσταση αυτή (άναμμα-σβήσιμο) να απεικονίζεται σε sec συνεχώς στα 2 δεξιότερα δεκαεξαδικά ψηφία των 7-segments displays.

IN 10H

MVI A,FFH ;at first all light are off

STA 3000H

MVI H,0AH ;in 0AFCH we store the supplementary program

MVI L,FCH ;a flag will be used: if flag==1, the interrupt button has been pressed

MVI A,00H

STA 0AF0H ;flag=0

INTERRUPT: ;makes falg=1 and returns to our program

MVI M,03H

INX H

MVI M,01H

MVI M,32H

INX H

MVI M,F0H

INX H

MVI M,0AH ;store the flag

INX H

MVI M,C9H ;RET code is C9H

INX H

MVI A,0DH

SIM

LOOP1:

EI ;enable interrupts

LDA 0AF0H

RAR ;if the flag is 1 continue

JNC LOOP1 ;else re-do the loop

LOOP2:

DI ;disable interrupts

MVI A,00H ;annul the flag

STA 0AF0H

MVI B,00H

MVI C,64H ;time stored for DELB - 100ms. We need 1 minute = 60\*1000ms=6000ms.

;6000/200=300 periods = 012H

MVI H,01H ;store in H-L

MVI L,2CH

MVI A,00H

STA 0AE0H ;initialize timer in 0AE0H - MSB

STA 0AE1H ;LSB

;store the message in 0AB0H

MVI A,10H ;set as 10H (blank) the 4 most-left digits

STA 0AB2H

STA 0AB3H

STA 0AB4H

STA 0AB5H

MVI A,05H ;initialiaze the counter in 0AE2

STA 0AE2H

MVI D,0AH ;initialize D-E for STDM

MVI E,B0H

MAINLOOP:

LDA 0AE1H ;loading the LSD of the timer to A

STA 0AB0H

LDA 0AE0H ;same for MSB

STA 0AB1H

;saving registers contents

PUSH PSW

PUSH B

PUSH D

PUSH H

CALL STDM

;retrieving register contenets

POP H

POP D

POP B

POP PSW

CALL DCD

MVI A,00H

STA 3000H ;lights on

CALL DELB

CALL DCD ;100ms have passed since the last DCD call

MVI A,FFH

STA 3000H ;lights off

EI ;enable interrupts

LDA 0AF0H

RAR ;check if there’s another interrupt

JC LOOP2

DI ;disable interrupts

LDA 0AE2H

DCR A

JNZ SMTHG ;if it’s zero set counter to 05h and increase timer by one

MVI A,05H

STA 0AE2H ;set counter back to 5

LDA 0AE1H

CPI 09H ;if it;s less than 9, go to SMTHG2

JC SMTHG2 ;else, LSB=0, MSB+=1

MVI A,00H

STA 0AE1H ;timer: LSB=0

LDA 0AE0H ;load to the MSB of the timer

INR A

STA 0AE0H ;store the increased MSB

JMP SMTHG3 ;jump there in order to avoid instructions that would happen

;if counter=0

SMTHG: ;saves A back to the counter address

STA 0AE2H

JMP SMTHG3

SMTHG2: ;if timer LSB < 9

LDA 0AE1H

INR A

STA 0AE1H

SMTHG3: ;

DCX H ;reduce H-L by one

MOV A,L

CPI 00H

JNZ MAINLOOP ; if L’s contents are 0, do next period

MOV A,H ;check H’s contents

MOV A,H

CPI 00H

JNZ MAINLOOP ; if H’s contents are 0, do next period

MVI A,00H

STA 0AF0H

JMP LOOP1 ;in case the interrupt button is pressed right after the last period and right

;Before coming back (thus we have to make flag=0)

END

**Άσκηση 2**

Στην άσκηση αυτή, θέλουμε ένα πρόγραμμα το οποίο όταν θα προκαλείται διακοπή RST 6.5 (INTR), να διαβάζει 2 διαδοχικά δεκαεξαδικά ψηφία ενός αριθμού που δίνονται από το πληκτρολόγιο. Ο αριθμός πρέπει να ανήκει στο εύρος 0-FFH.

Στη συνέχεια, να απεικονίζει τον αριθμό στα 2 αριστερότερα 7-segment display και να τον συγκρίνει με 3 κατώφλια Κ1<Κ2<Κ3, των οποίων οι τιμές βρίσκονται στους καταχωρητές C,D,E αντίστοιχα.

Τέλος, να ανάβει ένα από τα 4LSB ανάλογα με το διάστημα στο οποίο ανήκει η τιμή του αριθμού που εισήχθη.

Ο κώδικας το προγράμματος βρίσκεται στο αρχείο ex2.8085 και τα 3 κατώφλια ορίστηκαν ως:

K1=(100)10=64H στον καταχωρητή C

K2=(150)10=96H στον καταχωρητή D

K3=(200)10=C8H στον καταχωρητή E.

**Ο κώδικας του προγράμματος είναι:**

IN 10H

MVI H,0AH

MVI L,FCH ;(H)(L)<- 0AFCH

MVI M,3EH ;code for MVI A is 3EH

INX H

MVI M,01H ;Μoving 01H to A in order to store it to the flag

INX H

MVI M,32H ;code for STA is 32H

INX H

MVI M,F0H

INX H

MVI M,0AH ;The address must be saved with the hex bits in reverse order

INX H

MVI M,C9H ;code for RET is C9H

INX H

MVI A,00H ;initializing the flag with 0

STA 0AF0H

MVI A,0DH ;initializing the mask for interrupt RST 6.5

SIM ;sending it to mask interrupt register

L0:

EI ;enable interrupt

LDA 0AF0H ;load the flag address to A

RAR ;move the LSB to CY

JNC L0 ;if it is 0, loop to L0

L1:

DI ;if it is 1, disable interrupts and run the program

MVI A,00H ;make flag 0

STA 0AF0H ;starting address is 0AB0H

MVI D,0AH

MVI E,B0H ;storing starting address in D-E

MVI A,10H

STA 0AB0H

STA 0AB1H

STA 0AB2H

STA 0AB3H

CALL KIND ;call KIND to get the first digit

MOV B,A ;store it to B

STA 0AB5H

CALL KIND ;call KIND to get the second digit

STA 0AB4H

PUSH PSW ;saving all registers in the stack because

;STDM influences all registers

PUSH B

PUSH D

PUSH H

CALL STDM ;calling STDM

POP H ;retrieving all registers

POP D

POP B

POP PSW

CALL DCD ;calling DCD to depict the digits

MVI C,64H ;initialization of C,D,E

MVI D,96H

MVI E,C8H

RLC ;A has the second digit read

RLC ;left rotation of A

RLC

RLC

ORA B ;(A)<-(A)OR(B) to get B the first digit

RLC

RLC

RLC

RLC ;fixing the digit order, hex number in A register

;comparisons

MOV B,E ;K3 in B register

CMP B ;compare A with K3

JC L2 ;if A<K3, go to L2

JZ L2 ;if A=K3, go to L2

MVI A,F7H ;else turn ON the 4th LSB LED, (A)<-11110111

JMP L5

L2:

MOV B,D ;K2 in B register

CMP B ;compare A with K2.

JC L3 ;if A<K2, go to L3

JZ L3 ;if A=K2, go to L3

MVI A,FBH ;else turn ON the 3rd LSB LED, (A)<-11111011

JMP L5

L3:

MOV B,C ;K1 in B register

CMP B ;compare A with K1

JC L4 ;if A<K1, go to L4

JZ L4 ;if A=K1, go to L4

MVI A,FDH ;else turn ON the 2nd LSB LED, (A)<-11111101

JMP L5

L4:

MVI A,FEH ;if A<K1 or A=K1, (A)<-11111110 to turn ON the 1st LSB LED

L5:

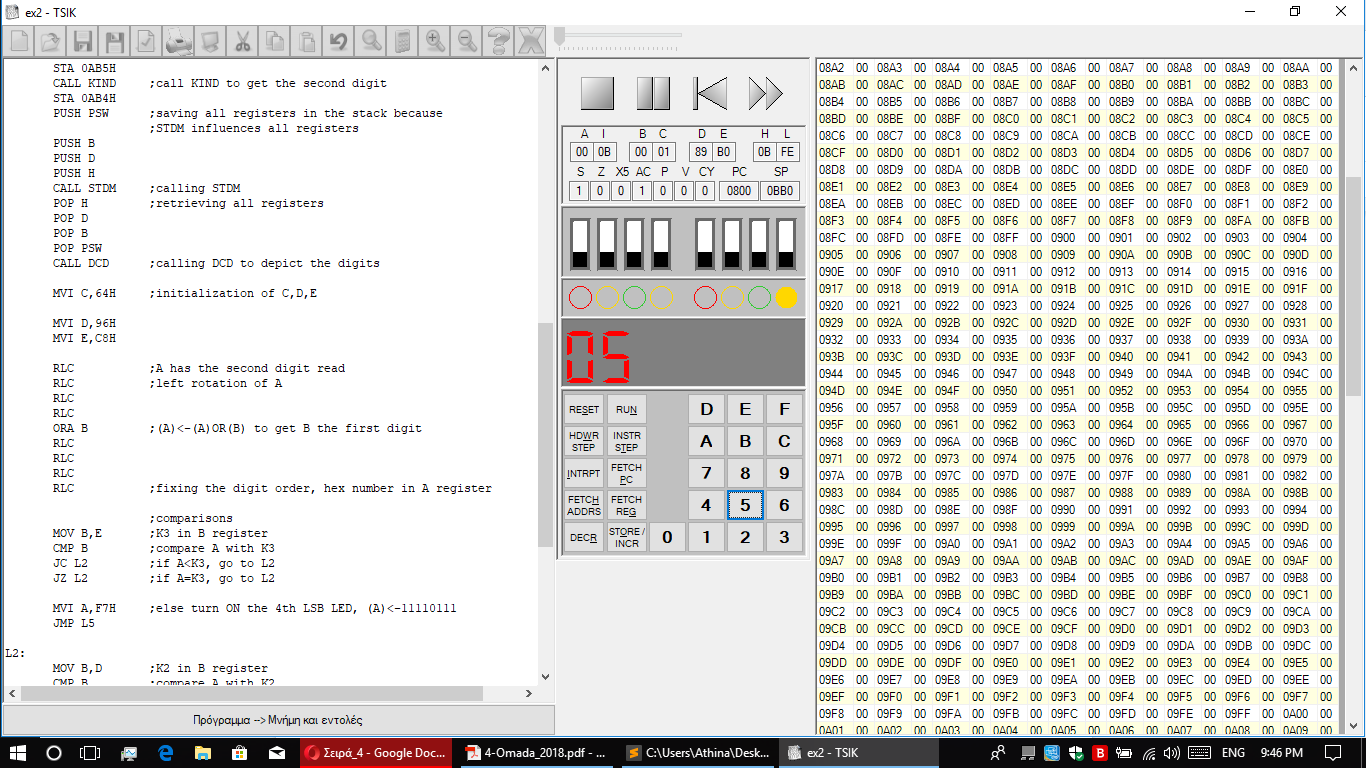
STA 3000H ;output in LEDs

JMP L0 ;go back to L0, waiting for the next interrupt

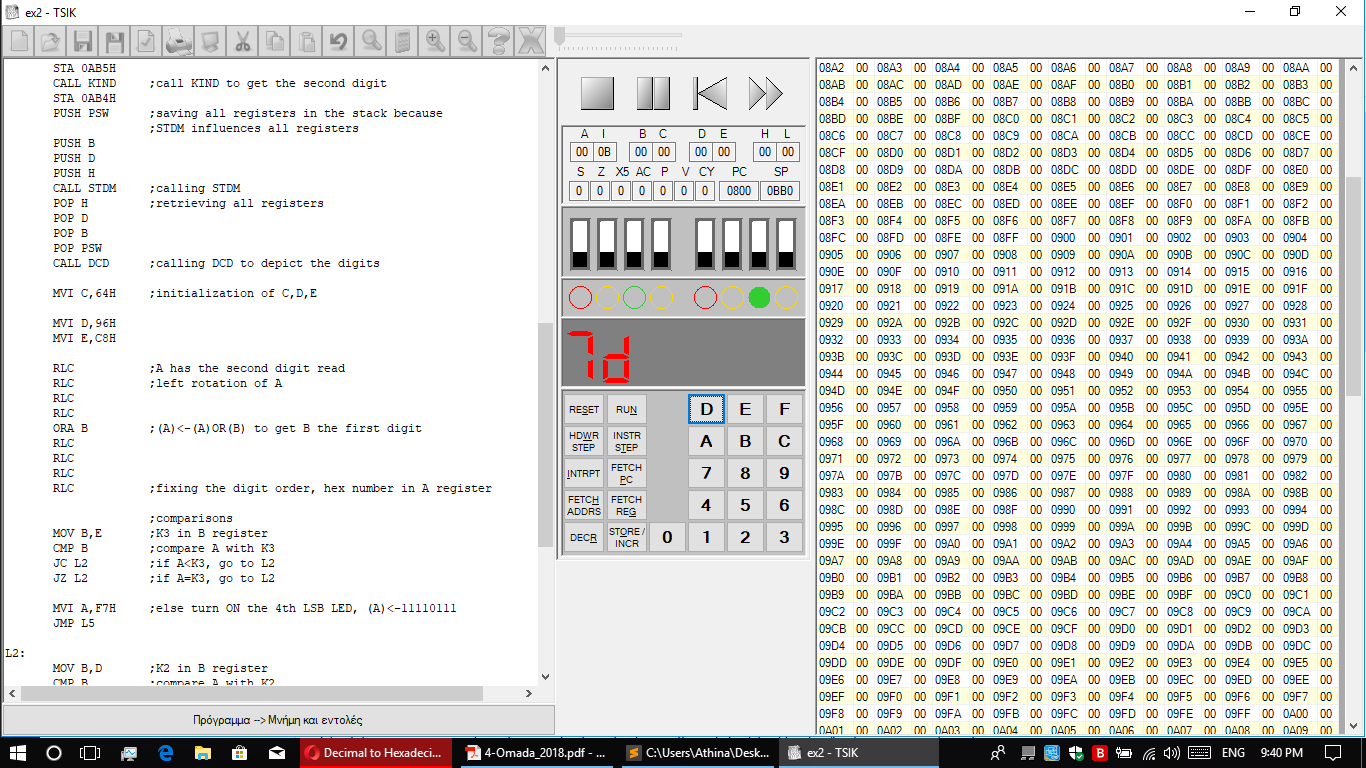
END

**Ενδεικτικά αποτελέσματα των προσομοιώσεων:**

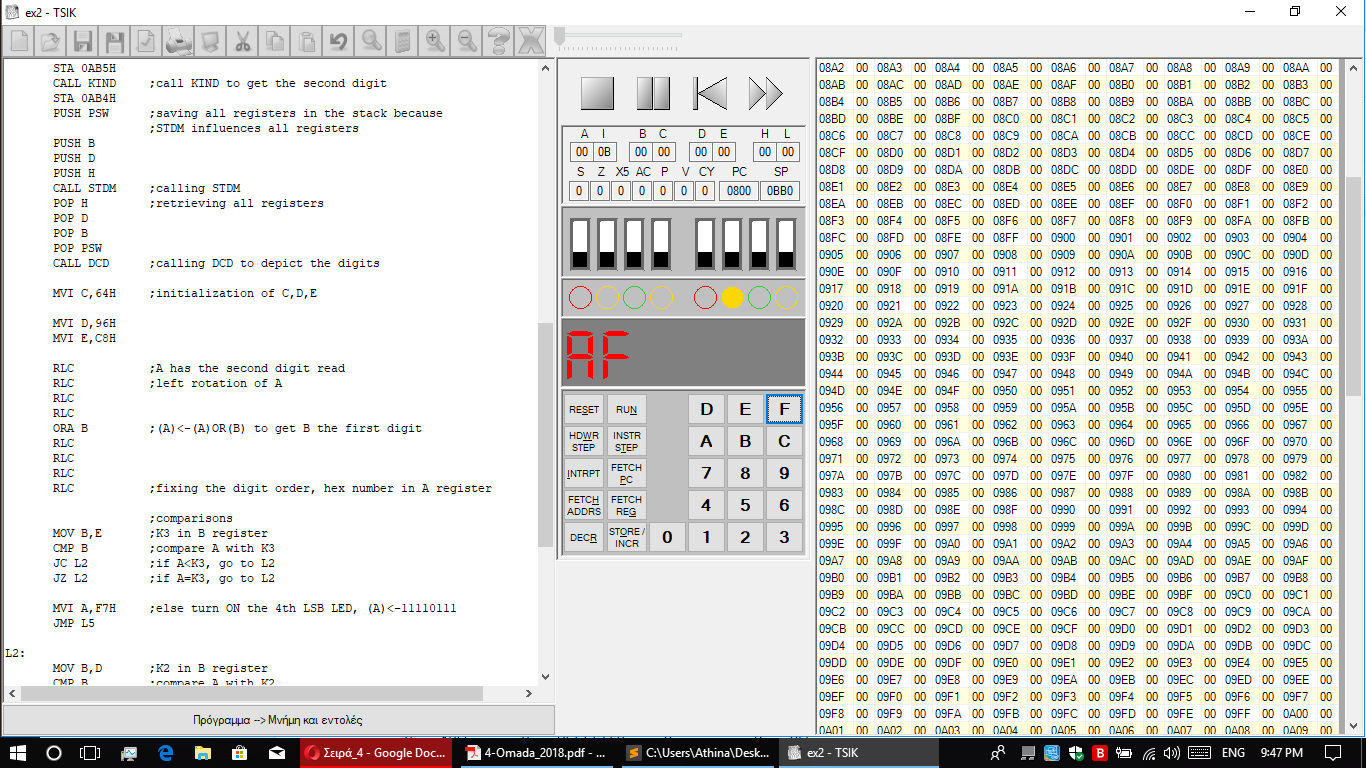
Για είσοδο **(5)10=05H**, μόλις πατηθεί το κουμπί INTRPT για να πραγματοποιηθεί διακοπή, αναμένουμε να ανάψει το 1ο LSB LED όπως και συμβαίνει.



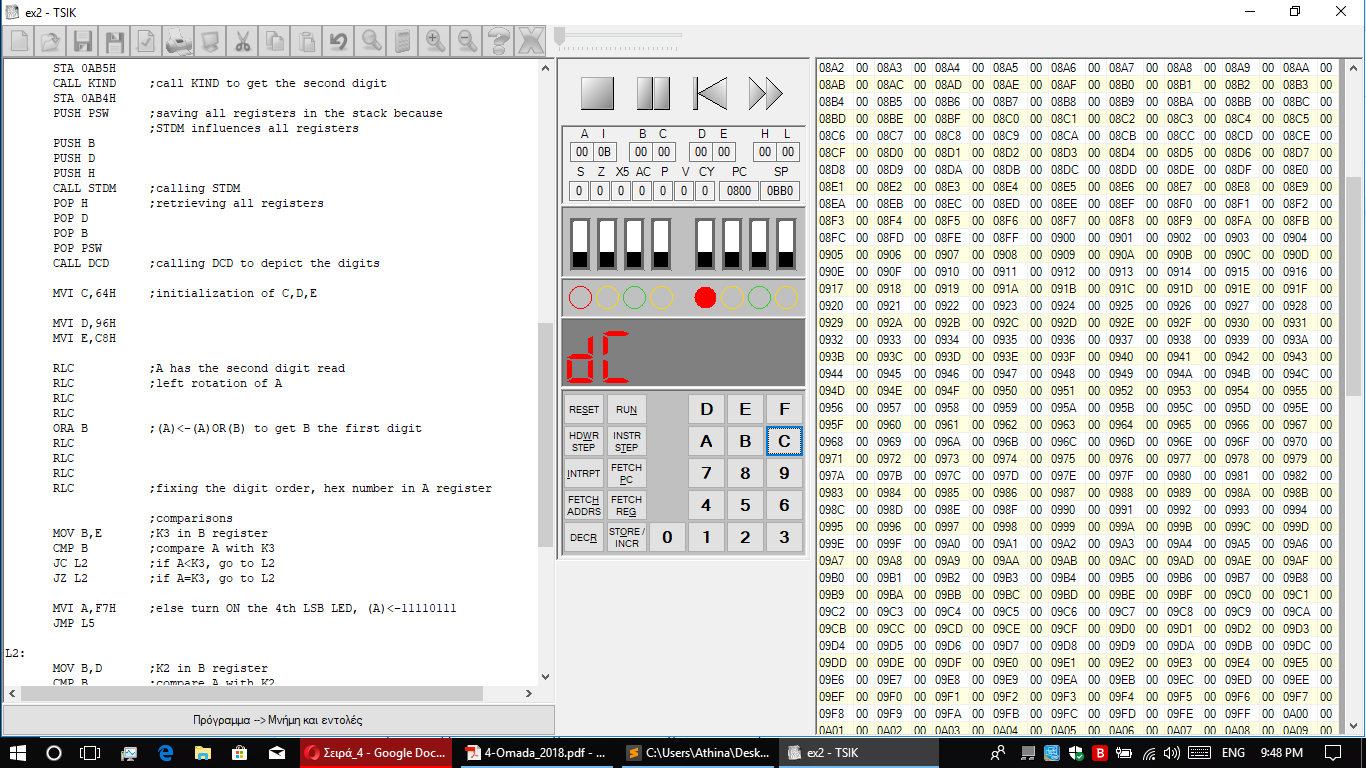
Για είσοδο **(125)10=7DH**, μόλις πατηθεί το κουμπί INTRPT για να πραγματοποιηθεί διακοπή, αναμένουμε να ανάψει το 2ο LSB LED όπως και συμβαίνει.



Για είσοδο **(175)10=AFH**, μόλις πατηθεί το κουμπί INTRPT για να πραγματοποιηθεί διακοπή, αναμένουμε να ανάψει το 3ο LSB LED όπως και συμβαίνει.

****

Για είσοδο **(220)10=DCH**, μόλις πατηθεί το κουμπί INTRPT για να πραγματοποιηθεί διακοπή, αναμένουμε να ανάψει το 4ο LSB LED όπως και συμβαίνει.

****

**Άσκηση 3**

Εφόσον η εκτέλεση της εντολής έχει ήδη αρχίσει, η διακοπή θα αναγνωριστεί μόλις η εκτέλεση τελειώσει. Οι ενέργειες που θα πραγματοποιηθούν είναι οι εξής:

1) Θα εκτελεστεί η παρούσα λειτουργία, δηλαδή θα γίνει JMP στην θέση 1200H.

2) Θα αναγνωριστεί η διακοπή RST 6.5

3) Θα απενεργοποιηθούν οι διακοπές

4) Θα αποθηκευτεί στις 2 προηγούμενες θέσεις μνήμης, από αυτή του τρέχοντος Stack Pointer, ο Program Counter. ((SP-1))=(PCH), ((SP-2))=(PCL) δηλαδή στη θέση 2FFFFH θα αποθηκευτεί ο αριθμός 10H και στη θέση 2FFFEH θα αποθηκευτεί ο αριθμός 0H.

5) Ο Stack Pointer θα δείχνει πλέον στην θέση που είναι αποθηκευμένο το LSB του PC που έγινε push στην στοίβα δηλαδή SP=2FFFEH.

6) O PC θα δείχνει πλέον στην θέση μνήμης 0034H, γιατί εκεί είναι η προεπιλεγμένη θέση αποθήκευσης της ρουτίνας εξυπηρέτησης της διακοπής RST 6.5 .

**Άσκηση 4**

Σε αυτή την άσκηση, μία συσκευή εισόδου είναι συνδεδεμένη με ένα μΥ-Σ 8085 στο port 20H. Αποστέλλει 16 δεδομένα των 8bit, αποστέλλοντας κάθε φορά 4bit, πρώτα τα LSB και έπειτα τα MSB. Η συσκευή, πριν αποστείλλει 4bit, προκαλεί διακοπή RST5.5.

Ζητείται ο υπολογισμός του μέσου όρου των δεδομένων που εισάγονται στο μΥ-Σ από τη συσκευή με συγγραφή προγράμματος Assembly καθώς και η ρουτίνα εξυπηρέτησης της διακοπής.

**Κύριο πρόγραμμα:**

MVI A,0EH ;(A)=0EH

SIM ;προγραμματισμός της μάσκας διακοπών επανεκκίνησης, επιτρέπουμε τη

διακοπή RST5.5

LXI H,0 ;το τελικό αποτέλεσμα θα είναι στους καταχωρήτες (Η)-(L) που

αρχικοποιούνται στο 0

MVI C,20H ;τα δεδομένα από τη συσκευή θα εισαχθούν με 32=20Η αποστολές

MVI B,00H ;flag 0 αν διαβάζω LSB αλλιώς FFH, διαβάζω πρώτα LSB

EI ;ενεργοποίηση του συστήματος διακοπών

ADDR:

MOV A,C ;βρόγχος για την ανάγνωση όλων των δεδομένων

CPI 0

JNZ ADDR

DI ;απενεργοποίηση του συστήματος διακοπών αν διαβαστούν όλα τα

δεδομένα

DAD H ;διαίρεση του περιεχομένου των καταχωρητών (H)-(L) με το 16

DAD H ;ισοδυναμεί με ολίσθηση αριστερά του H-L κατά 4 θέσεις

DAD H ;ακέραιο μέρος στον H και κλασματικό στον L

DAD H ;δηλαδή με προσθήκη του Η-L 4 φορές στον εαυτό του

HLT

**Ρουτίνα εξυπηρέτησης διακοπής RST5.5:**

002C:

JMP RST5.5

RST5.5:

PUSH PSW ;σώζεται η τρέχουσα κατάσταση του μΥ-Σ (Α και σημαίες)

IN 20H ;δεδομένα εισόδου από το port 20H, αποθήκευση στον Α

MVI D,0

ΑΝI 0FH ;αποθήκευση στον Α μόνο των 4 LSB που εισάγονται από τη συσκευή

CPI B,00H

JZ FINAL ;αν έχουν εισαχθεί τα LSB, δε χρειάζεται να κάνω ολίσθηση

RLC ;αν έχουν εισαχθεί τα MSB, χρειάζεται να κάνω 4xRLC

RLC

RLC

RLC

FINAL:

MOV E,A ;περιεχόμενο ανάγνωσης στον Ε

DAD D ;(H)(L)<-(H)(L)+(D)(E)

DCR C ;προστέθηκε κατάλληλα το δεδομένο που διαβάστηκε

MOV A,B

CMA

MOV B,A ;κάνω bitwise συμπλήρωμα της σημαίας

POP PSW ;επαναφορά της κατάστασης του μΥ-Σ

EI

RET ;επιστροφή στο κύριο πρόγραμμα