**Εργαστήριο Μικροϋπολογιστών**

**1η σειρά ασκήσεων**

Ομάδα: Δ12

**Μέλη**:

Βακαλόπουλος Θεόδωρος , ΑΜ: 03114013

Μαυρομμάτης Ιάσων , ΑΜ: 03114771

Νικητοπούλου Δήμητρα , ΑΜ: 03114954

Παρακάτω παρατίθενται οι κώδικες των ασκήσεων που μας ζητήθηκαν, με επαρκή σχολιασμό για να γίνεται εμφανής ο τρόπος σκέψης και η υλοποίηση που επιλέξαμε. Όπου θεωρήσαμε ότι χρειάζονται περαιτέρω εξηγήσεις προστέθηκαν στο συγκεκριμένο ερώτημα.

**Άσκηση 1:**

LXI B,03E8H ;1000=03E8 MS

START1:

MVI E,00H ;ΑΡΧΙΚΟΠΟΙΩ ΜΕΤΡΗΤΗ

MVI A,FFH

STA 3000H ;ΕΜΦΑΝΙΖΩ ΤΟ 0

CALL DELB

;ΔΕΝ ΚΡΑΤΩ ΠΟΥΘΕΝΑ ΤΗΝ ΚΑΤΕΥΘΥΝΣΗ ΤΗΣ ΚΙΝΗΣΗΣ, ΔΙΑΔΟΧΙΚΑ

;ΠΑΩ ΠΑΝΩ ΚΑΙ ΚΑΤΩ ΣΥΝΕΧΩΣ

UP:

CALL READING ;ΔΙΑΒΑΖΩ ΔΙΑΚΟΠΤΕΣ

INR E ;ΜΕΤΡΗΣΗ ΑΝΩ

MOV A,E

CMP D ;ΑΝ ΕΧΩ ΦΤΑΣΕΙ ΣΤΗΝ ΤΙΜΗ ΠΟΥ ΘΕΛΩ ΠΑΩ ΣΕ ΜΕΤΡΗΣΗ ΚΑΤΩ

JNC DOWN1 ;ΑΛΛΙΩΣ ΕΜΦΑΝΙΖΩ ΤΗΝ ΤΙΜΗ Κ ΚΑΤΟΠΙΝ ΣΥΝΕΧΙΖΩ ΑΝΩ

CMA

STA 3000H

CALL DELB

JMP UP

DOWN1:

MOV A,E ;ΕΜΦΑΝΙΖΩ ΤΗΝ ΤΙΜΗ ΠΟΥ ΔΕΝ ΕΙΧΕ ΠΡΟΛΑΒΕΙ ΝΑ ΕΜΦΑΝΙΣΤΕΙ ΚΑΤΑ

CMA ;ΤΗ ΜΕΤΡΗΣΗ ΑΝΩ

STA 3000H

CALL DELB

DOWN2:

CALL READING

DCR E ;ΜΕΤΡΗΣΗ ΚΑΤΩ

MOV A,E

CPI 00H ;ΟΤΑΝ ΦΤΑΣΩ ΣΤΟ 0 ΞΕΚΙΝΩ ΜΕΤΡΗΣΗ ΑΝΩ ΠΑΛΙ

JZ START1

CMA ;ΑΛΛΙΩΣ ΕΜΦΑΝΙΖΩ

STA 3000H

CALL DELB

JMP DOWN2

;ΕΛΕΓΧΟΣ ΓΙΑ ΙΚΑΝΟΠΟΙΗΣΗ ΣΥΝΘΗΚΗΣ ΚΑΙ

;ΕΠΙΣΤΡΟΦΗ ΜΕΣΩ ΤΟΥ ΚΑΤΑΧΩΡΗΤΗ D ΤΟΥ ΑΡΙΘΜΟΥ ΠΟΥ ΔΙΝΕΤΑΙ ΣΤΑ ΜSB

READING:

START2:

LDA 2000H ;ΑΝ LSB=0 ΔΕΝ ΚΑΝΩ ΤΙΠΟΤΑ

RRC

JNC START2

RRC

RRC

RRC

ANI 0FH

MOV D,A ; D ΚΡΑΤΑ ΤΗΝ ΤΙΜΗ ΠΟΥ ΔΙΝΕΤΑΙ ΑΠΟ ΤΟΥΣ ΔΙΑΚΟΠΤΕΣ

CPI 00H ; ΑΝ D=0 ΞΕΚΙΝΑ ΑΠΟ ΤΗΝ ΑΡΧΗ

JZ START1

RET

END

**Άσκηση 2**:

Στη συγκεκριμένη άσκηση, προσπαθώντας να πετύχουμε την απενεργοποίηση των διακοπών όταν δεν ικανοποιείται η δεδομένη συνθήκη για το lsb, οδηγηθήκαμε σε αδιέξοδο καθώς μετρούσε το σύστημα πάντα μια έξτρα διακοπή κατά τη μετακίνηση του διακόπτη. Αυτό που κάναμε, λοιπόν, ήταν να επιτρέπουμε συνεχώς διακοπές κι απλά να δεχόμαστε μόνο αυτές που ικανοποιούν τη συνθήκη, αμελώντας τις υπόλοιπες.

IN 10H

MVI A,0DH

SIM

EI

LXI B,0600H ;In register pair B-C store the delay

LXI H,0941H

MVI M,10H ;Store 10H in memory addresses 0941-0945H

INX H ;so that the 5 left 7-segment displays are always Off

MVI M,10H ;and only the right one displays the number of interrupts

INX H

MVI M,10H

INX H

MVI M,10H

INX H

MVI M,10H

INX H ;Because every press of the interrupt button creates two

MVI M,01H ;interrupts we ignore half of them based on the flag in

MVI A,0DH ;memory address 0946H

SIM

MVI D,00H ;Counter initialisation

MVI E,00H ;number of interrupts

LXI H,0940H ;Store in register pair H-L the address 0940H

START:

MOV A,D

ANI 0FH ;Isolate the 4 LSBs

RRC ;Rotate them to become the 4 MSBs

RRC

RRC

RRC

CMA

STA 3000H ;Display the counter

INR D ;Increment counter for next iteration

DI

CALL DELB ;Call routine for a delay of 0.1s

EI

JMP START ;Loop forever

INTR\_ROUTINE:

DI

LDA 2000H

RRC

JNC RETURN ;If LSB switch is OFF ignore the interrupt

LXI H,0946H

MOV A,M

XRI 01H ;Change flag for next interrupt

MOV M,A

CPI 00H

JNZ RETURN ;Ignore half interrupts

INR E ;Increment the counter of interrupts

MOV A,E

ANI 0FH ;Isolate 4 LSBs to display the result modulo 16

MOV E,A

LXI H,0940H

MOV M,E

PUSH D

LXI D,0940H ;Store in register pair D-E the address of the message

CALL STDM ;Move the message in the right address for display

CALL DCD ;Display the counter in right 7-segment display

POP D

RETURN:

EI

RET

END

**Άσκηση 3:**

Σε αυτή την άσκηση διαβάζουμε δύο δεκαεξαδικά ψηφία και στη συνέχεια αφού δημιουργήσουμε ένα δεκαεξαδικό αριθμό από αυτά καλούμαστε να τον μετατρέψουμε σε δεκαδικό αριθμό 3 ψηφίων. Αυτό ουσιαστικά το επιτυγχάνουμε με αλλεπάλληλες αφαιρέσεις αρχικά του 100 από τον αριθμό μας αυξάνοντας το μετρητή των εκατοντάδων και στη συνέχεια του 10 αυξάνοντας αντίστοιχα το μετρητή των δεκάδων μέχρι να μείνει ένα υπόλοιπο μικρότερο του 10 που θα αντιστοιχεί στις μονάδες. Αφού κάνουμε τη διάσπαση απεικονίζουμε στον ενδείκτη 7 τμημάτων τα τρία δεκαδικά ψηφία που σχηματίζουν το ζητούμενο δεκαδικό αριθμό.

IN 10H

LOOP1:

CALL KIND ;Read first hex digit

RLC ;Rotate left 4 times

RLC

RLC

RLC

MOV B,A

CALL KIND ;Read second hex digit

ADD B ;Create input hex number

;MOV B,A ;Store it in register B

LXI D,0000H ;Initialise hundreds, tens = 0

LOOP2: ;Convert hex number to decimal

SUI 64H ;Subtract 1 hundred

JC CASE1

INR D ;hundreds++

JMP LOOP2

CASE1:

ADI 64H ;Restore one extra subtraction

LOOP3:

SUI 0AH ;Subtract 1 ten

JC CASE2

INR E ;tens++

JMP LOOP3

CASE2:

ADI 0AH ;Restore one extra subtraction

LXI H,0900H ;Prepare the message of the 6 7-segment displays

MVI M,10H ;The hex code 10H represents space in 7-segment display

INX H

MVI M,10H

INX H

MVI M,10H

INX H

MOV M,A ;Store units

INX H

MOV M,E ;Store tens

INX H

MOV M,D ;Store hundreds

LXI D,0900H ;Store the address of the message in register pair D-E

CALL STDM ;Move the message in the proper address for display

CALL DCD ;Display the result

JMP LOOP1 ;Loop forever

END

**Άσκηση 4:**

Όπως και στην άσκηση 2 έτσι κι εδώ πρέπει να αγνοούμε τις διακοπές όταν βρισκόμαστε σε μία συγκεκριμένη κατάσταση, δηλαδή όταν το βαγονέτο είναι ακινητοποιημένο. Για να επιτύχουμε αυτή τη λειτουργία αρχικά επιτρέπουμε όλες τις διακοπές και μέσα στη ρουτίνα εξυπηρέτησης ελέγχουμε αν πρέπει να εκτελεστεί ή όχι ανάλογα με τη θέση των διακοπτών. Με παρόμοια λογική αντιμετωπίζουμε και το πρόβλημα των διπλών διακοπών που προκαλούνται με το πάτημα του πλήκτρου Intr. Συγκεκριμένα χρησιμοποιούμε μια θέση μνήμης ως σημαία για να προσδιορίσουμε αν πρόκειται για άρτια ή περιττή διακοπή κάτι που ελέγχουμε στην αρχή της ρουτίνας εξυπηρέτησης για να αποφασίσουμε αν θα εκτελεστεί όλος ο κώδικας της ρουτίνας ή όχι.

IN 10H

MVI A,0DH

SIM

EI

MVI D,00H ;The LSB of register D determines the move of wagon left or right

MVI E,01H ;Initialise wagon position in LSB

MOV A,E

CMA

OUT 30H ;Display wagon position

LXI B,0FFFH ;In register pair B-C store the delay of a single move

LXI H,0900H ;Because every press of the interrupt button creates two

MVI M,01H ;interrupts we ignore half of them based on the flag in

;memory address 0900H

START:

IN 20H ;Read input switch

RRC

JNC START ;If LSB is OFF wait

MOV A,D

RRC

JNC MOVE\_LEFT ;Check if wagon is moving left or right

MOVE\_RIGHT:

MOV A,E

RRC ;Move wagon one position right

CPI 80H ;Check if previous position was the right border

JZ RIGHT\_BORDER

MOV E,A ;If no, current position is acceptable

JMP CONTINUE

RIGHT\_BORDER: ;Else move wagon two positions left

RLC

RLC

MOV E,A ;Save current position

MVI D,00H ;Change wagon direction

JMP CONTINUE

MOVE\_LEFT:

MOV A,E

RLC ;Move wagon one position left

CPI 01H ;Check if previous position was the left border

JZ LEFT\_BORDER

MOV E,A ;If no, current position is acceptable

JMP CONTINUE

LEFT\_BORDER: ;Else move wagon two positions right

RRC

RRC

MOV E,A ;Save current position

MVI D,01H ;Change wagon direction

CONTINUE:

MOV A,E

CMA

OUT 30H ;Display wagon current position

DI

CALL DELB ;Create a delay of 0.5s

EI

JMP START ;Loop forever

INTR\_ROUTINE:

DI

IN 20H

RRC

JNC RETURN ;If LSB switch is OFF ignore the interrupt

LXI H,0900H

MOV A,M

XRI 01H ;Change flag for next interrupt

MOV M,A

CPI 00H

JNZ RETURN ;Ignore half inrerrupts

MOV A,D ;If LSB switch is ON change wagon direction

XRI 01H

MOV D,A

RETURN:

EI

RET

END