1η ΕΡΓΑΣΤΗΡΙΑΚΗ ΆΣΚΗΣΗ
ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"

Ομάδα: 16
Ημερομηνία επίδειξης: 11/10/2017
Μέλη:
Κερασιώτης Ιωάννης, Α.Μ.:03114951
Πευκιανάκης Κωνσταντίνος, Α.Μ.: 03114897
Ραφτόπουλος Ευάγγελος, Α.Μ.:03114743

ΑΣΚΗΣΗ 1

Στην άσκηση αυτή ζητείται να κατασκευαστεί ένα χρονόμετρο δευτερολέπτων που θα απεικονίζει τον χρόνο σε δυαδική μορφή πάνω στα LEDs εξόδου του μLab. Επειδή δεν διευκρινίζεται από την εκφώνηση εμείς κάναμε την εξής παραδοχή: θεωρήσαμε ότι όταν αλλάζει η είσοδος ενώ “τρέχει” το πρόγραμμα θα ολοκληρώσει την προηγούμενη εκτέλεση και μετά θα ξεκινήσει την νέα διεργασία. Δηλαδή αν έχουμε δώσει σαν είσοδο την τιμή χ=8 και βρίσκεται στην στιγμή t=6 (δηλαδή έχει εμφανίσει τα 0,1,2,3,4,5,6) και αλλάξει η είσοδος και γίνει χ=4 τότε θα ολοκληρωθεί η προηγούμενη διεργασία δηλαδή θα κάνει και τα βήματα 7,8,7,6,5,4,3,2,1,0 ΚΑΙ ΜΕΤΑ θα ξεκινήσει ή νέα λειτουργία δηλαδή η 1,2,3,4,3,2,1,0.

Η λογική για την επίλυση της άσκησης είναι η εξής. Αρχικά ολισθαίνουμε την είσοδο για να πάει στο 4 MSB ώστε να σχηματίσουμε τον δοθέν αριθμό. Στην συνέχεια γίνονται δύο loop. Στο πρώτο εμφανίζουμε τους αριθμούς ξεκινώντας από το 0 μέχρι τον αριθμό χ και στο δεύτερο Loop εμφανίζουμε τους αριθμούς από το χ μέχρι το 0. Σε κάθε επανάληψη ελέγχουμε το LSB ώστε αν είναι 0 τότε να σταματήσει η διεργασία. Επίσης σε κάθε επανάληψη εμφανίζουμε στα LEDs την έξοδο και για να γίνουν αντιληπτές οι αλλαγές ορίζουμε και καλούμε την ρουτίνα χρονοκαθυστέρησης (DELB) του 1 sec.

Ο κώδικας της άσκησης αυτής δίνεται παρακάτω:

```
        LXI B,03E8H          ;εισαγωγή της τιμής 1000=3E8 (1000ms=1sec) για την
                             χρονοκαθυστέρηρη DELB
START:
        MVI A,00H       ;αρχικοποίηση του καταχωρητή A με την τιμή 00

        LDA 2000H       ;έλεγχος της εισόδου για να διαπιστώσουμε αν το LSB είναι
        RRC             ;OFF ώστε να σταματήσει η λειτουργία
        JNC START
        LDA 2000H       ;έλεγχος για την ειδική περίπτωση που οι διακόπτες των
        CPI 01H         ;MSB είναι OFF.
        JZ ALL_OFF

        LDA 2000H       ;ανάγνωση της εισόδου
        RRC             ;και δεξιές ολισθήσεις(4) ώστε να μεταφέρουμε τα bits
        RRC             ;των MSB στα LSB ώστε να πάρουμε τον αριθμό της εισόδου
        RRC
        RRC

        ANI 0FH         ;αφού έχουμε την είσοδο στα LSB κάνουμε AND της εισόδου με
                        ;τον αριθμό 00001111 ώστε να μηδενιστούν τα MSB και να πάρουμε την είσοδο

        MOV E,A         ;κρατάμε αντίγραφο του καταχωρητή A γιατί επηρεάζεται η τιμή του
        MVI C,00H       ;ο C έχει τον ρόλο του μετρητή ο οποίος εμφανίζεται στα LEDs
```

```
LOOP1:
        LDA 2000H       ;έλεγχος για το LSB αν είναι OFF ώστε να σταματήσει η λειτουργία
        RRC
        JNC LOOP1
        MOV A,C
        CMA             ;η έξοδος είναι αρνητικής λογικής
        STA 3000H       ;εμφάνιση του καταχωρητή C στην έξοδο των LEDs
        CALL DELB       ;καλούμε την χρονοκαθυστέρησης DELB ώστε να είναι εμφανείς οι ενδείξεις των
                        ; LEDs
        INR C           ;αυξάνουμε την τιμή του C ώστε να εμφανίσει τον επόμενο αριθμό
        MOV A,C         ;ελέγχουμε αν ο C έφτασε στην άνω τιμή η οποία έχει αποθηκευτεί
        CMP E           ;στον καταχωρητή E
        JNZ LOOP1       ;αν όχι επιστρέφει ώστε να αυξηθεί ο C και να εμφανιστεί
LOOP2:                  ;αλλιώς πήγαινε και μέτρα προς τα πίσω

        LDA 2000H       ;έλεγχος για το LSB αν είναι OFF ώστε να σταματήσει η λειτουργία
        RRC
        JNC LOOP2
                                                MOV A,C
        CMA             ;η έξοδος είναι αρνητικής λογικής
        STA 3000H       ;εμφάνιση του καταχωρητή C στην έξοδο των LEDs
        CALL DELB       ;καλούμε την χρονοκαθυστέρησης DELB ώστε να είναι εμφανείς οι ενδείξεις των
                        ; LEDs
        DCR C           ;μειώνουμε την τιμή του καταχωρητή C αφού είμαστε στην περίπτωση
                        ;που μετράμε προς τα πίσω
        JNZ LOOP2       ;έλεγχος για το αν έφτασε στο 0 ώστε να ξαναμετρήσει προς τα πάνω
        JMP START       ;αν έφτασε στο 0 πήγαινε ξανά στην αρχή

ALL_OFF:                ;εμφάνιση στην έξοδο των LEDs την τιμή 0
        MVI A,00H
        CMA
        STA 3000H
        JMP START
END
```

ΑΣΚΗΣΗ 2

Η άσκηση αυτή ζητάει να φτιάξουμε ένα πρόγραμμα που θα αναπαριστά ένα μετρητή του οποίου η τιμή θα εμφανίζεται στα MSB LEDs της εξόδου με μία καθυστέρηση ώστε να είναι ευδιάκριτες οι αλλαγές. Επίσης το πρόγραμμα θα πρέπει να μετράει τις διακοπές που γίνονται και να εμφανίζει τον αριθμό αυτό στο 7-segment display σε δεκαεξαδική μορφή.

Ξέρουμε ότι κάθε κλικ του ποντικιού (πάτημα και άφημα) προσμετράται ως δύο διακοπές. Στον παρακάτω κώδικα έχουμε προσθέσει κάποιες εντολές οι οποίες δηλώνονται στα σχόλια με ****. Αυτές διορθώνουν το πρόβλημα της διπλής προσμέτρησης. Η λογική είναι η ακόλουθη. Έχουμε μία θέση μνήμης στην οποία έχουμε ένα αριθμό που θεωρούμε ότι παίρνει τις τιμές 00H και FFH (η μία είναι συμπλήρωμα της άλλης). Όταν η τιμή αυτής της θέσης μνήμης είναι 00H προσμετράμε την διακοπή, ενώ όταν είναι FFH τότε την απορρίπτουμε. Κάθε φορά που πατάμε την διακοπή και μπαίνουμε στην ρουτίνα εξυπηρέτησης συμπληρώνουμε την τιμή αυτή (δηλαδή από 00 σε FF και το αντίστροφο). Αυτό έχει ως αποτέλεσμα το κάθε κλικ (πάτημα και άφημα) να προσμετράται ως μία φορά.

Η υλοποίηση της άσκησης αυτής αποτελείται από δύο μέρη. Στο πρώτο έχουμε ένα loop το οποίο αυξάνει τον μετρητή και τον εμφανίζει στα LEDs της εξόδου. Το δεύτερο μέρος αποτελείται από την ρουτίνα εξυπηρέτησης διακοπής. Σε αυτή την ρουτίνα αυξάνουμε το αριθμό των διακοπών και ελέγχουμε αν έγινε ίσο με F ώστε να μηδενιστεί. Επίσης συμπληρώνουμε την θέση μνήμης 0800 για να προσμετράται μία φορά το κάθε κλικ. Τέλος ελέγχουμε σε κάθε είσοδο στην ρουτίνα την τιμή του LSB έτσι ώστε αν είναι 0 να αγνοήσει την διακοπή.

Σε αυτή την άσκηση όπως και στην 4η αντιμετωπίσαμε το εξής πρόβλημα: βάλαμε σε καταχωρητή τον αριθμό των διακοπών, αλλά διαπιστώσαμε ότι πολλές διακοπές δεν προσμετρούνταν. Αυτό οφείλεται στο γεγονός ότι η ρουτίνα χρονοκαθυστέρησης DELB όταν καλείται κάνει push όλους τους καταχωρητές. Για παράδειγμα αν έχουμε στον καταχωρητή C το πλήθος των διακοπών, όσο είναι στην ρουτίνα DELB και πατάμε διακοπές θα προσμετρούνται αλλά μόλις επιστρέψει από την DELB τότε θα επαναφέρει την τιμή του C που είχε πριν την κλήση της ρουτίνας χρονοκαθυστέρησης, με αποτέλεσμα να χαθούν όλες οι διακοπές που πατήθηκαν όσο βρίσκονταν στην DELB. Για να αντιμετωπίσουμε το πρόβλημα αυτό βάλαμε σε θέση μνήμης το πλήθος των διακοπών (στην θέση 0BF0H για να τα εμφανίσει μετά η DCD). Έτσι δεν "χάνουμε" διακοπές όσο η εκτέλεση βρίσκεται στην ρουτίνα χρονοκαθυστέρησης DELB. Ένας άλλος τρόπος διόρθωσης αυτού του προβλήματος είναι να βάλλουμε πριν την DELB την εντολή DI και μετά την DELB την εντολή EI, δηλαδή:

```
DI
CALL DELB
EI
```

Ο κώδικας αυτής της άσκησης είναι ο ακόλουθος:

```
        NOP             ;*****
        IN 10H          ;*****

        LXI B,0064H     ;εισαγωγή του αριθμού 100D=64H (δηλαδή 0,1sec) όπως ζητάει η άσκηση για την
                        ; χρονοκαθυστέρησης DELB
        MVI A,0DH       ;ενεργοποίηση των διακοπών και της INTRPT
        SIM
        EI

        LXI H,0BF0H     ;Αρχικοποίηση του τμήματος μνήμης που
        MVI M,00H       ;χρησιμοποιείται από τη ρουτίνα DCD
        INX H
        MVI M,10H       ;θέλουμε να εμφανίζεται συνεχώς στο δεξιότερο 7-segment display
        INX H
        MVI M,10H
        INX H
        MVI M,10H
        INX H
        MVI M,10H
        INX H
        MVI M,10H
                ;για την εμφάνιση του μετρητή στα MSB των LEDs
        MVI A,00H       ;αρχικοποίηση του μετρητή A με την αρχική τιμή 0
LOOP1:
        CMA             ;η έξοδος είναι αρνητικής λογικής
        STA 3000H       ;εμφάνιση του μετρητή A
        CMA             ;συμπλήρωση του A αφού έχει αλλάξει λόγω της εμφάνισης στην STA
        CALL DCD        ;καλούμε την ρουτίνα DCD ώστε να εμφανίσει στο display
        CALL DELB       ;καλούμε την χρονοκαθυστέρηση DELB για να γίνουν ευδιάκριτες οι αλλαγές στα
                        ;LEDs
        ADI 10H         ;όταν φτάσει στην 00010000 τότε προσθέτουμε 10H ώστε να μην ανοίξουν τα
                        ;LSB
        JMP LOOP1       ;συνεχής μέτρηση

INTR_ROUTINE:   ;ορισμός ρουτίνας διακοπής
        PUSH PSW        ;push των σημαιών και του A
        LDA 0800H ;**** ;εδώ έχουμε την περίπτωση για το κλικ (πάτημα και άφημα) να λαμβάνεται
        CPI 00H   ;**** ;ως μία διακοπή. αυτό το κάνουμε θεωρώντας ότι η τιμή της μνήμης 0800H
        JNZ CONT  ;**** ;παίρνει τις τιμές 00H όπου και μετράμε την διακοπή και την τιμή FFH όπου
```

```
                    ; απορρίπτει την διακοπή
        LDA 2000H           ;Φόρτωσε την είσοδο
        RRC                 ;Είναι το πρώτο LSB OFF?
        JNC CONT            ;Αν ναι τότε πήγαινε στην ετικέτα CONT ώστε να μην μετρήσει η διακοπή

        LDA 0BF0H       ;η τιμή της θέσης μνήμης που περιέχει την ένδειξη του δεξιότερου 7-segmaet
                        display
        INR A           ;αυξάνεται κατά ένα ώστε να εμφανιστεί στο display
        ANI 0FH         ;όταν φτάσει στον αριθμό 0FH τότε θέλουμε να μετρήσει από την αρχή
        STA 0BF0H       ;αποθήκευση στην θέση μνήμης που βρίσκεται η ένδειξη του αριστερότερου 7-
                        ;segment diaplay
CONT:
        LDA 0800H ;****  ;συμπληρώνεται η τιμή της θέσης μνήμης 0800H που ελέγχει αν η διακοπή θα
                    ;προσμετρηθεί
        CMA         ;****
        STA 0800H ;****
        POP PSW         ;επαναφορά των σημαιών και του A όπως ήταν πριν την εισαγωγή στην ρουτίνα
                    ;διακοπής
        EI          ;ενεργοποίηση των διακοπών αφού απενεργοποιούνται κατά την εισαγωγή στην ρουτίνα
                    ; διακοπής
        RET             ;επιστροφή στη πρόγραμμα

END
```

ΑΣΚΗΣΗ 3

Η άσκηση αυτή ζητάει ένα πρόγραμμα που θα δέχεται δύο αριθμούς (0-F) και θα εμφανίζει το αποτέλεσμα 16*χ+y στα τρία αριστερότερα 7-segment display σε δεκαδική μορφή.

Για το διάβασμα των αριθμών χρησιμοποιούμε την ρουτίνα KING και ελέγχουμε αν πατήθηκε δεκαεξαδικό ψηφίο. Με την ρουτίνα DCD εμφανίζουμε το αποτέλεσμα στα 7-segment display έχοντας αποθηκεύσει τις τιμές στις θέσεις 0BF3-0BF5. Μετά του υπολογισμού 16*χ+y χωρίζουμε τον αριθμό σε μονάδες δεκάδες και εκατοντάδες τα οποία και εμφανίζουμε.

Ο κώδικας της άσκησης είναι ο παρακάτω:

```
        IN 10H
        MVI A,00H

        LXI H,0BF0H     ; Αρχικοποίηση του τμήματος μνήμης που
        MVI M,10H       ; χρησιμοποιείται από τη ρουτίνα DCD
        INX H
        MVI M,10H       ;θέλουμε να εμφανιστούν στα 3 αριστερότερα από τα 7-display segments
        INX H
        MVI M,10H
        INX H
        MVI M,00H
        INX H
        MVI M,00H
        INX H
        MVI M,00H

START:
READ_FIRST_NUMBER:
        CALL KIND               ;Διαβάζουμε από το πληκτρολόγιο
        CPI 10H                 ;Κοιτάμε αν είναι δεκαεξαδικό το ψηφίο (0-F)
        JNC READ_FIRST_NUMBER   ;Περιμένουμε μέχρι να έρθει ένα έγκυρο ψηφίο
```

```
        ADD A       ;A+A=2A
        ADD A       ;(2A)+(2A)=4A
        ADD A       ;(4A)+(4A)=8A
        ADD A       ;(8A)+(8A)=16A το οποίο θέλουμε
        MOV B,A     ;στον καταχωτητή B έχουμε το αποτέλεσμα της  16*X ,όπου X1 είναι ο πρώτος αριθμός
                    ;(χ)

READ_SECOND_NUMBER:
        CALL KIND               ;Διαβάζουμε από το πληκτρολόγιο
        CPI 10H                 ;Κοιτάμε αν είναι δεκαεξαδικό το ψηφίο (0-F)
        JNC READ_SECOND_NUMBER  ;Περιμένουμε μέχρι να έρθει ένα έγκυρο ψηφίο

        ADD B                   ;προσθέτουμε τον 2ο αριθμό με το αποτέλεσμα 16*X που βρίσκεται στον
                                ;καταχωρητή B

        MVI B,FFH               ;εύρεση εκατοντάδων, δεκάδων και μονάδων του αποτελέσματος
HUNDR:              ;για εκατοντάδες
        INR B       ;αφαιρούμε από το B το 64H=100D μέχρι να γίνει αρνητικός και μετράμε το πλήθος των
                    ;αφαιρέσεων
        SUI 64H
        JNC HUNDR           ;αν εξακολουθεί να είναι θετικός πήγαινε στην HUNDR
        ADI 64H             ;προσθέτουμε το 64H για να διορθώσουμε το αρνητικό υπόλοιπο
        MOV C,A             ;στον C βάζω τον 2-ψηφιο αριθμό που μένει
        MOV A,B
        STA 0BF5H           ;αποθήκευσε τις εκατοντάδες στην κατάλληλη θέση για την DCD (αριστερότερο
                            ;ψηφίο)
        MOV A,C
        MVI B,FFH
DECA:               ;για δεκάδες
        INR B       ;αφαιρούμε από το B το 0AH=10D μέχρι να γίνει αρνητικός και μετράμε το πλήθος των
                    ;αφαιρέσεων
        SUI 0AH
        JNC DECA            ;αν εξακολουθεί να είναι θετικός πήγαινε στην DECA
        ADI 0AH             ;προσθέτουμε το 0AH για να διορθώσουμε το αρνητικό υπόλοιπο
        STA 0BF3H     ;αποθήκευσε τις μονάδες στην κατάλληλη θέση για την DCD (τρίτο από τα αριστερά
                      ;ψηφίο)
        MOV A,B
        STA 0BF4H  ;αποθήκευσε τις δεκάδες στην κατάλληλη θέση για την DCD (δεύτερο από τα αριστερά
                   ;ψηφίο)

        CALL DCD    ;καλούμε την DCD για εμφάνιση του αποτελέσματος
        JMP START   ;το πρόγραμμα είναι συνεχούς λειτουργίας
END
```

ΑΣΚΗΣΗ 4

Για την υλοποίηση της άσκησης αυτής χρησιμοποιούμε 2 loop το ένα κινεί το βαγονέτο προς τα αριστερά και το άλλο προς τα δεξιά. Η λογική για την αλλαγή πορειάς είναι η εξής. Έχουμε σε μία θέση μνήμης έναν αριθμό που θεωρούμε ότι παίρνει τιμές 00H και FFH. Αν είναι 00 θεωρούμε ότι κινείται προς τα αριστερά ενώ όταν είναι FF κινείται προς τα δεξιά. Έτσι σε κάθε διακοπή η ρουτίνα εξυπηρέτησης συμπληρώνει τον αριθμό αυτό ώστε να αλλάξει πορειά, δηλαδή να μεταβεί στην άλλη loop με τον κατάλληλο έλεγχο που γίνεται. Επίσης στην αρχή κάθε loop καθώς και στην αρχή της ρουτίνας εξυπηρέτησης γίνεται έλεγχος για το LSB. Αν είναι 0 τότε θα πρέπει το βαγονέτο να σταματήσει και αν πατηθεί διακοπή θα πρέπει να την “αγνοήσει”. Και σε αυτή την άσκηση διορθώνουμε το πρόβλημα του κλικ δηλαδή που προσμετράται δύο

φορές με τον ίδιο τρόπο που περιεγράφηκε και στην άσκηση 2. Όπως και στην άσκηση 2 αποθηκεύουμε τις διακοπές στη θέση μνήμης 0800H και όχι σε καταχωρητή για τον λόγο που εξηγήσαμε στην 2η άσκηση.

Ο κώδικας της άσκησης αυτής είναι ο ακόλουθος:

```
        CMA
        STA 3000H

LEFT:
        MVI M,00H       ;στην θέση μνήμης 0800H βάζω το 00H δηλαδή η κίνηση είναι προς τα αριστερά
        CALL DELB       ;καλώ την ρουτίνα χρονοκαθυστέρησης
        IN 20H          ;Φορτώνω την είσοδο
        RRC             ;δεξιά ολίσθηση ώστε να ελέγξουμε το LSB
        JNC LEFT        ;αν είναι OFF τότε περιμένουμε την ενεργοποίησή του
        MOV A,M         ;έλεγχος της θέσης μνήμης 0800H που δηλώνει την φορά της κίνησης
        CPI 00H         ;αν δεν είναι 00 τότε άλλαξε κατεύθυνση
        JNZ RIGHT
        MOV A,D         ;Μετακίνησε το περιεχόμενο του καταχωρητή D στον καταχωρητή A
        RLC             ;Ολίσθηση προς τα αριστερά
        CPI 01H         ;Είναι το MSB ψηφίο ένα ? Έχει φτάσει στο 10000000 δηλαδή?
        JZ RIGHT        ;Αν ναι τότε πρέπει να γυρνάνε τα LED προς τα πίσω. Αυτό κάνει η παραπάνω
                        ;ετικέτα RIGHT
        CMA             ;η έξοδος είναι αρνητικής λογικής
        OUT 30H         ;εμφάνιση στην έξοδο
        CMA             ;Συμπλήρωμα του A για να πάρω το αρχικό περιεχόμενο του
        MOV D,A         ;ανανέωσε τον καταχωρητή D
        JMP LEFT        ;Ξανακάλεσε την ιδιά διαδικασία


RIGHT:
        MVI M,FFH       ;στην θέση μνήμης 0800H βάζω το FFH δηλαδή η κίνηση είναι προς τα δεξιά
        CALL DELB       ;καλώ την ρουτίνα χρονοκαθυστέρησης
        IN 20H          ;Φορτώνω την είσοδο
        RRC             ;δεξιά ολίσθηση ώστε να ελέγξουμε το LSB
        JNC RIGHT       ;αν είναι OFF τότε περιμένουμε την ενεργοποίησή του
        MOV A,M         ;έλεγχος της θέσης μνήμης 0800H που δηλώνει την φορά της κίνησης
        CPI 00H         ;αν είναι 00 τότε άλλαξε κατεύθυνση
        JZ LEFT
        MOV A,D         ;Μετακίνησε το περιεχόμενο του D στον A
        RRC             ;Ολίσθηση προς τα δεξιά
        CPI 80H         ;Είναι το LSB ψηφίο 1?Εχει φτάσει δηλαδή στο 00000001?
        JZ LEFT         ;Αν ναι τότε πρέπει να πηγαίνουν τα LED προς τα μπροστά. Οπότε καλούμε ξανά την
                        ;ετικέτα LEFT
        CMA             ;η έξοδος είναι αρνητικής λογικής
        OUT 30H         ;εμφάνιση στην έξοδο
        CMA             ;Συμπλήρωμα του A για να πάρω το αρχικό περιεχόμενο του
        MOV D,A         ;ανανέωσε τον καταχωρητή D
        JMP RIGHT       ;Ξανακάλεσε την ιδια διαδικασία

INTR_ROUTINE:   ;ορισμός ρουτίνας διακοπής
        PUSH PSW
        LDA 0801H       ;****   εδώ έχουμε την περίπτωση για το κλικ (πάτημα και άφημα) να λαμβάνεται
        CPI 00H         ;****   ως μία διακοπή. αυτό το κάνουμε θεωρώντας ότι η τιμή της μνήμης 0801H
        JNZ END_ROUTINE ;****   ;παίρνει τις τιμές 00H όπου και μετράμε την διακοπή και την τιμή FFH
                                ;όπου LEFT
```

```
        LDA 2000H       ;έλεγχος για το LSM ώστε αν είναι OFF να αγνοήσει την διακοπή
        ANI 01H
        JZ END_ROUTINE
        MOV A,M         ;άλλαξε την τιμή της θέσης μνήμης 0800H δηλαδή άλλαξε την κατεύθυνση της
                        ;κίνησης
        CMA             ;αυτό επιτυγχάνεται με την συμπλήρωση της τιμής αυτής
        MOV M,A
END_ROUTINE:
        LDA 0801H       ;****   συμπληρώνεται η τιμή της θέσης μνήμης 0801H που ελέγχει αν η διακοπή
                        ;θα προσμετρηθεί
        CMA             ;****
        STA 0801H       ;****
        POP PSW
        EI              ;ενεργοποίηση διακοπών αφού απενεργοποιούνται αυτόματα
        RET

END
```