Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ. Και Μηχ. Υπολογιστών
Εργαστήριο Μικροϋπολογιστών , 7ο εξάμηνο - Ροή Υ
Ακαδημαϊκή Περίοδος : 2011-2012

1η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ

Γερακάρης Βασίλης Α.Μ.:03108092
Διβόλης Αλέξανδρος Α.Μ.: 03107238
Ίεντσεκ Στέργιος Α.Μ.: 03108690

Άσκηση 1i

Στην άσκηση αυτή πρέπει να υλοποιήσουμε ένα μετρητή από το 0-16 ο οποίος θα εμφανίζει τον κάθε αριθμό για 1 sec. Για να το πετύχουμε αυτό χρησιμοποιούμε: μια επαναληπτική διαδικασία η οποία ξεκινά να μετρά από το FF_H έως το $F0_H$ έχοντας πάντα στο νου μας την αρνητική λογική των LEDs και μια κλήση συνάρτησης καθυστέρησης DELB έχοντας θέσει στον BC την τιμή $1000_{10} = BE8_H$. Τέλος σε κάθε ολοκλήρωση ενός κύκλου μετρήσεων καλούμε την ρουτίνα BEEP του συστήματος. Ο κώδικας παρατίθεται παρακάτω:

```
START: LXI B,0BE8H   ;Put 1000 in counter B-C (delay 1000msec)

       MVI A,FFH     ;A<-11111111:LEDS OFF
LOOP_: STA 3000H     ;PUT LED OUTPUT
       CALL DELB     ;DELAY FOR (B-C)x1 msec
       DCR A
       CPI EFH       ;1111111~11111110~...~11110000~11101111(in reg D)
       JNZ LOOP_     ;KEEP COUNTING DOWN UNTIL FINISH
       CALL BEEP     ;BEEP routine changes B-C reg
       JMP START
END
```

Άσκηση 2i

Στην άσκηση αυτή υλοποιούμε ένα πρόγραμμα το οποίο ελέγχει τα LEDs, δηλαδή καθορίζει για πόσο χρονικό διάστημα αυτά θα είναι ON και για πόσο OFF. Έχουμε κάποιους περιορισμούς όσον αφορά το χρόνο καθυστέρησης, όταν ο χρήστης επιλέγει 0 θα πρέπει η καθυστέρηση να είναι το λιγότερο 200ms και αν επιλέγει 15 να είναι 1700ms. Για να το πετύχουμε αυτό, έχουμε θέσει στον BC την τιμή 100ms, καλούμε πάντα μια φορά την DELB και μετά ανάλογα με την είσοδο A που έχει δώσει καλούμε A+1 φορές την καθυστέρηση των 100ms. Την είσοδο που δίνει ο χρήστης την παίρνουμε εύκολα περνώντας την μέσα από κάποια κατάλληλη μάσκα ($F0_H$ ή $0F_H$ για ON ή OFF αντίστοιχα) και στην περίπτωση ON κάνουμε και τέσσερεις ολισθήσεις δεξιά. Ο κώδικας παρατίθεται παρακάτω:

```
FLASH: LXI B,0064H   ;B-C<-100
       CALL LEDS_ON  ;Leds On
       CALL DELB     ;Delay for 100msec
       LDA 2000H     ;Read switches
       ANI F0H       ;A(AND)11110000    : Mask
       RRC
       RRC
       RRC
       RRC           ;Shift right 4 times
       INR A         ;A<-A+1
LOOP_ON:
       CALL DELB     ;Delay for 100msec
       DCR A         ;A<-A-1
       JNZ LOOP_ON   ;Loop until A=0----All_del=[(A+1)x100+100]msec
       CALL LEDS_OFF;Leds Off
       CALL DELB     ;Delay for 100msec
       LDA 2000H     ;Read switches
       ANI 0FH       ;A(AND)00001111    : Mask
       INR A         ;A<-A+1
LOOP_OFF:
       CALL DELB     ;Delay for 100msec
       DCR A         ;A<-A-1
       JNZ LOOP_OFF  ;Loop until A=0----All_del=[(A+1)x100+100]msec
       JMP FLASH     ;Continues...

LEDS_ON:
       MVI A,00H
       STA 3000H
       RET

LEDS_OFF:
       MVI A,FFH
       STA 3000H
       RET
END
```

Άσκηση 2ii

Στην παρούσα άσκηση μας ζητείται να τροποποιήσουμε το συνεχούς λειτουργίας πρόγραμμα που φαίνεται στους πίνακες 1, 3, 4, ώστε να επιτρέπονται διακοπές μόνο όταν το MSB των διακοπτών είναι ON, να φαίνονται στα LEDS το τρέχον πλήθος των διακοπών που έχουν πραγματοποιηθεί (4MSB) καθώς και η μέτρηση (4LSB) με συχνότητα 1μέτρηση/100msec. Ακόμη, όταν θα εγείρεται η διακοπή RST6.5 (η μόνη επιτρεπτή) θα πρέπει να μετράμε το πλήθος των διακοπτών που είναι ON. Παρατίθεται στο zip αρχείο ο αντίστοιχος κώδικας.

Η διακοπή RST6.5 είναι ευαίσθητη στο επίπεδο του παλμού. Αυτό σημαίνει ότι εγείρεται όσο το σήμα εισόδου βρίσκεται σε λογικό 1. Το γεγονός αυτό μπορεί να μας δημιουργήσει προβλήματα στην περίπτωση που ο κώδικας της ρουτίνας εξυπηρέτησης εκτελείται σε μικρότερο χρονικό διάστημα από την διάρκεια του παλμού που προκαλεί την διακοπή στον επεξεργαστή. Τότε μετά την επιστροφή από τη ρουτίνα εξυπηρέτησης ο επεξεργαστής θα εκλάβει το επίπεδο του παλμού σαν μια νέα αίτηση διακοπής την οποία και θα εξυπηρετήσει. Έτσι, ενώ πρακτικά έχει συμβεί μία διακοπή, ο κώδικας εξυπηρέτησης της διακοπής θα έχει εκτελεστεί 2 φορές (ή και περισσότερες). Στην περίπτωσή μας, όπου θέλουμε να μετρήσουμε το πλήθος των διακοπών που έχουν συμβεί, αυτό μας δημιουργεί σοβαρό πρόβλημα.

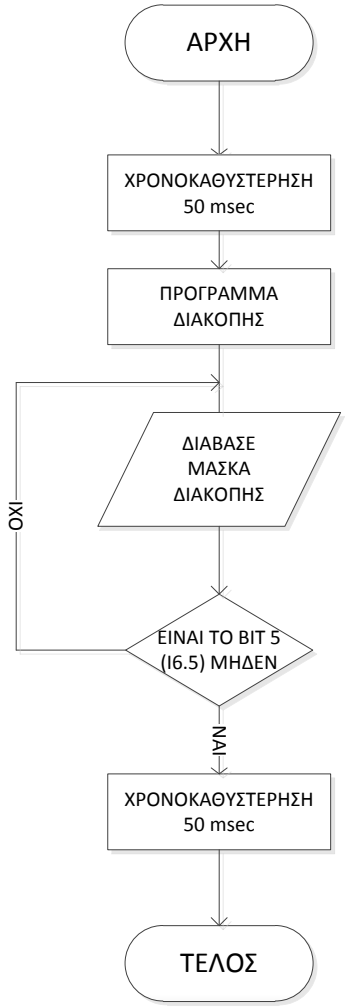
Αυτό μπορούμε να το αντιμετωπίσουμε πρακτικά αναμένοντας μέσα στην ρουτίνα εξυπηρέτησης έως ότου αναγνωρίσουμε ότι έγινε 0 το αντίστοιχο bit, το οποίο μας δείχνει αν εκκρεμεί διακοπή 6.5. Τότε απλά θα θέταμε έναν βρόχο αναμονής ώστε διαβάζοντας στον Accumulator με την εντολή RIM τις εκκρεμότητες των διακοπών, να αναμέναμε τον μηδενισμό του I6.5 bit. Όμως προκύπτει και ένα επιπλέον πρόβλημα. Αν πραγματοποιηθεί μια διακοπή τώρα το 5^{o} bit I6.5, το οποίο μας επιστρέφεται από την RIM θα είναι σωστά ενημερωμένο μετά από ~50msec, οπότε θα επιστρέψει την τιμή 0. Αυτό οφείλεται στο ότι η διακοπή προκαλείται από πλήκτρο, που είναι μία μηχανική διάταξη. Όπως και κάθε μηχανική διάταξη, δεν δίνει απευθείας σταθερή τιμή 1, αλλά υπάρχει ένα μικρό χρονικό διάστημα όπου γίνονται διάφορες αναταραχές, οι οποίες μπορεί να εκληφθούν ως πολλές διαδοχικές διακοπές. Έτσι θα πρέπει, τόσο στην αρχή όσο και στο τέλος της ρουτίνας εξυπηρέτησης, να έχουμε μία χρονοκαθυστέρηση των 50msec, ώστε να μην έχουμε προβλήματα πολλαπλής εξυπηρέτησης μίας διακοπής. Με την επαναληπτική διαδικασία της αναμονής καταφέρνουμε να βλέπουμε το αποτέλεσμα της ρουτίνας εξυπηρέτησης όσο το πλήκτρο INTR είναι πατημένο.

Στη συνέχεια παρατίθεται το λογικό διάγραμμα της ρουτίνας εξυπηρέτησης το οποίο χρησιμοποιήσαμε στον κώδικά μας.

Επίσης, πρέπει να σημειωθεί πως αποτρέψαμε την εξυπηρέτηση διακοπής κατά την κλήση ρουτίνας χρονοκαθυστέρησης (DELB). Αυτό διότι κατά την εκτέλεση της ρουτίνας διακοπής μπορούν να επηρρεαστούν τόσο η τρέχουσα κατάσταση (Accumulator, Flags) όσο και άλλοι καταχωρητές. Βέβαια θα μπορούσαμε να προστατευθούμε από ανεπιθύμητες επηρροές της ρουτίνας, εκμεταλλευόμενοι την στοίβα, προωθόντας εκεί όποιον καταχωρητή θέλουμε να τροποποιήσουμε. Στην περίπτωση αυτή όμως θα πρέπει να αποθηκεύουμε τον

μετρητή διακοπών στην μνήμη, ώστε να επικοινωνούν το κύριο πρόγραμμα με την ρουτίνα εξυπηρέτησης, ή να ενημερώναμε μιά θέση μνήμης, χρησιμοποιώντας την ως σημαία ένδειξης διακοπής και σε κάθε loop του κύριου προγράμματος να ελέγχαμε αν έχει συμβεί διακοπή, οπότε να αυξάναμε τον interrupt counter. Είναι σαφές ότι τότε ο κώδικάς μας θα δυσχέραινε τόσο σε πολυπλοκότητα, όσο και σε χρόνο εκτέλεσης.

Αντίθετα στον κώδικά μας χρησιμοποιούμε τον καταχωρητή D ως μετρητή του πλήθους των διακοπών.



Παρατιθεται ο κωδικας:

```
        LXI D,0000H   ;(Initialize counters) D:Interrupts E:Loop counter
        MVI A,0DH     ;Interrupt Mask
        SIM
        LXI B,0064H   ;B-C=100 for 100msec delay (DELB)
LOOP_1:
        MOV A,D       ;Put Interrup Counter I7 I6 I5 I4 I3 I2 I1 I0 -> A
        CMA
        ANI 0FH       ;A<-0 0 0 0 I3 I2 I1 I0 (A<-D(mod)16)
        RLC
        RLC
        RLC
        RLC           ;A<-I3 I2 I1 I0 0 0 0 0
        MOV H,A
        MOV A,E       ;Put Loop Counter L7 L6 L5 L4 L3 L2 L1 L0 -> A
        CMA
        ANI 0FH       ;A<-0 0 0 0 L3 L2 L1 L0
        ORA H         ;A<-I3 I2 I1 I0 L3 L2 L1 L0
        STA 3000H     ;Led this!
        DI            ;Disable Interrupts to be in DELB routine
        CALL DELB
        LDA 2000H     ;Check MSB of switches to decide if you should EI
        RLC
        JNC GO_ON1
        EI
GO_ON1:
        INR E         ;Increase loop counter
        JMP LOOP_1    ;Loop!

INTR_ROUTINE:         ;This routine must be at this address 0AFC.
        PUSH B
        INR D         ;Increase Interrupt counter
        PUSH D        ;Push D-E! When POP D, the increased D will be poped!
        PUSH PSW      ;Push Flags and Accumulator
        LXI B,0032H   ;Set B-C for 50msec delay with DELB
        CALL DELB
MY_PROG:
        LXI D,0800H   ;D:Loop counter E:On-Switch counter
        LDA 2000H
AGAIN:
        RRC
        JNC GO_ON2
        INR E
GO_ON2:
        DCR D
        JNZ AGAIN
        MOV A,E
        RLC
        RLC
        RLC
        RLC
        CMA
        STA 3000H     ;MY_PROG ends
LOOP_2:
        RIM
        ANI 20H       ;A(AND)0010 0000 to check I6.5
        JNZ LOOP_2    ;Repeat until I6.5 is 0
        CALL DELB     ;Delay for another 50msec
        EI
        POP PSW
        POP D
        POP B
RET
END     ;Remember to put at 0AFC the instruction C3 2A 08-->JMP INTR_ROUTINE
        ;If DELB calls are commented (single step mode)
        ;put at 0AFC the instruction C3 27 08-->JMP INTR_ROUTINE
```