Σχολή Ηλεκτρολόγων Μηχανικών
&
Μηχανικών Υπολογιστών

Εργαστήριο Μικροϋπολογιστών

Πρώτη Εργαστηριακή Άσκηση

Λαμπράκος Χρήστος, Α.Μ: 03112062
Μανδηλαράς Νικηφόρος, Α.Μ: 03112012
Σπαθαράκης Δημήτριος, Α.Μ: 03113523
Έβδομο Εξάμηνο

Παραδοτέα: 17/10/2015

Άσκηση 1ii

Αρχικά γίνεται κλήση της ρουτίνας **BEEP**, όπως και ζητείται. Για να αποφευχθούν οι όποιες επιπλοκές λόγω της επίδρασης στους καταχωρητές του μικροϋπολογιστικού συστήματος, το ζεύγος **BC**, το οποίο περιέχει την απαιτούμενη χρονοκαθυστέρηση του ενός δευτερολέπτου, παίρνει τη δεκαεξαδική τιμή **03E8H** μετά το τρέξιμο της **BEEP**.

Κατόπιν, εκλέγεται ο συσσωρευτής ως ο μετρητής του χρόνου. Τον μηδενίζουμε και εισερχόμαστε στη λούπα, όπου καθυστερούμε για ένα δευτερόλεπτο, και στη συνέχεια αυξάνουμε τον μετρητή κατά ένα. Λαμβάνοντας υπ' όψιν την αντίστροφη λογική του ανάμματος των LED, παίρνουμε το συμπλήρωμα του συσσωρευτή πριν στείλουμε το περιεχόμενό του στη θύρα εξόδου. Έπειτα τον επαναφέρουμε στη "φυσιολογική" τιμή του, και τη συγκρίνουμε με το 15. Αν προκύψει ισότητα, ο συσσωρευτής μηδενίζεται και η λούπα τρέχει από την αρχή. Αν όχι, συνεχίζουμε κανονικά.

Το πρόγραμμα είναι συνεχούς λειτουργίας. Ακολουθεί ο κώδικας, μαζί με τα απαραίτητα σχόλια.

oneone

```
CALL BEEP     ;first beep

       MVI B,03H     ;(BC)==1000
       MVI C,E8H

START: MVI A,00H     ;our counter

LOOPA: CALL DELB     ;delay of (BC)x1ms==1 sec
       INR A         ;one more second
       CMA           ;LEDs function with inverse logic
       STA 3000H     ;update output
       CMA           ;restore counter
       CPI 0FH       ;did we reach 15?
       JZ START      ;if yes, reinitialize counter
       JMP LOOPA     ;else continue

       END
```

Άσκηση 2i

Το κυρίως πρόγραμμα καλεί διαρκώς και διαδοχικά τις ρουτίνες **MYON** και **MYOFF**, οι οποίες και ελέγχουν, με βάση τις τιμές των διακοπτών, το άναμμα και το σβήσιμο των LED αντίστοιχα. Η φιλοσοφία αμφότερων των ρουτινών είναι πανομοιότυπη, γι αυτό στην παρούσα αναφορά θα αναλυθεί μόνο η πρώτη.

Η ιδέα έχει ως εξής: φορτώνουμε στον διπλό καταχωρητή **HL** την ελάχιστη καθυστέρηση των 200 msec, η οποία αντιστοιχεί στο να μην είναι ενεργός κανείς εκ των τεσσάρων αριστερότερων διακοπτών. Τελικός σκοπός είναι ο **HL** να περιέχει τη συνολική καθυστέρηση. Αυτό πραγματοποιείται με 2 λούπες, τη μία εμφωλευμένη στην άλλη--υπάρχουν αναμφισβήτητα αποδοτικότερες λύσεις.

Πριν τους επαναληπτικούς βρόχους, απομονώνουμε τα 4 MSBs των διακοπτών (δηλαδή της θύρας εισόδου **2000H**) με μία εντολή AND και τέσσερις δεξιές ολισθήσεις. Αν το αποτέλεσμα είναι μηδενικό, τότε όπως προαναφέρθηκε, ο **HL** περιέχει ήδη την απαιτούμενη καθυστέρηση, και προχωράμε στο άναμμα των LED για 200 msec πριν την

επιστροφή στο κυρίως πρόγραμμα. Αν όχι, τότε ορίζουμε 2 μετρητές: τον συσσωρευτή, ο οποίος και περιέχει τα MSBs, και τον **B**, στον οποίο φορτώνουμε την τιμή **64H** (100 στο δεκαδικό σύστημα).

Η εσωτερική λούπα βασίζεται στον **B**, και κάθε φορά προσθέτει 1 msec στον διπλό καταχωρητή **HL** (δεν ήμαστε σίγουροι για το πώς θα αντιμετωπίζαμε την υπερχείλιση σε περίπτωση που προσθέταμε απευθείας 100).

Η εξωτερική λούπα φροντίζει ώστε η εσωτερική να τρέξει τόσες φορές, όσες απαιτεί ο συσσωρευτής. Η τελική καθυστέρηση είναι:

$$(HL) = 200 + 100 \cdot (A)\, msec$$

Αυτή η καθυστέρηση μεταφέρεται τελικά στον διπλό καταχωρητή **BC**, προκειμένου να λειτουργήσει σωστά η κλήση της **DELB**, η οποία καλείται αμέσως μετά τη φόρτωση της τιμής **00H** στη θύρα εξόδου (αντίστροφη λογική) και πριν την έξοδο από τη ρουτίνα.

Το πρόγραμμα είναι και εδώ συνεχούς λειτουργίας. Ακολουθεί ο κώδικας, ο οποίος και επισυνάπτεται στο αρχείο zip της παραδιδόμενης εργασίας.

```
       IN 10H                          twoone
START: CALL MYON
       CALL MYOFF
       JMP START
MYON:  MVI H,00H
       MVI L,C8H          ;minimum delay = 200 msec
       LDA 2000H
       ANI F0H
       RRC
       RRC
       RRC
       RRC                ;A now has 4 left MSBs
       CPI 00H            ;check if additional delay is needed
       JZ SHOOT
LOOPB: MVI B,64H          ;this loop goes in LOOPA as many times as the 4 MSBs
LOOPA: INX H              ;this loop adds 100 msecs of delay
       DCR B
       JNZ LOOPA
       DCR A
       JNZ LOOPB
SHOOT: MOV B,H
       MOV C,L            ;(BC) = 200+100x(4 MSBs) msecs
       MVI A,00H
       STA 3000H          ;light 'em up
       CALL DELB
       RET
MYOFF: MVI H,00H          ;similar to MYON, but working with
       MVI L,C8H          ;the 4 LSBs
       LDA 2000H
       ANI 0FH
       CPI 00H
       JZ SHOOTB
LOOPC: MVI B,64H
LOOPD: INX H
       DCR B
       JNZ LOOPD
       DCR A
       JNZ LOOPC
SHOOTB: MOV B,H
       MOV C,L            ;(BC) = 200+100x(4 MSBs) msecs
       MVI A,FFH
       STA 3000H          ;kill 'em
       CALL DELB
       RET
       END
```

Άσκηση 2ii (α)

Το χρονόμετρο, που αφορά τα 4 λιγότερο σημαντικά LED, ρυθμίζεται με τρόπο παρόμοιο αυτού της άσκησης 1ii. Η μόνη διαφορά είναι πως εδώ, επειδή θέλουμε η έξοδος να δείχνει και το μέτρημα των διακοπών που έχουν επισυμβεί--το οποίο αποθηκεύουμε στον καταχωρητή **H**, "ρολάρουμε" το περιεχόμενο του τελευταίου 4 θέσεις στα δεξιά ακριβώς πριν στείλουμε την έξοδο στη θύρα **3000H**.

Οι βασικοί καταχωρητές που χρησιμοποιήθηκαν στην άσκηση είναι:

I) Το ζεύγος **BC**, για τη ρύθμιση της ρουτίνας **DELB** κατά 100 msec.

II) Ο καταχωρητής **H**, για τη μέτρηση των διακοπών.

III) Ο καταχωρητής **L**, ως σημαία ασφαλείας για την παράκαμψη του προβλήματος των διπλών διακοπών.

Κάθε φορά που μπαίνουμε στη λούπα που αυξάνει τη μέτρηση του χρόνου κατά ένα δέκατο του δευτερολέπτου, ελέγχουμε το MSB της θύρας εισόδου. Αν αυτό είναι μηδέν, απενεργοποιούμε τις διακοπές (ώστε να αποφευχθεί επίτευξη διακοπής με κλειστό διακόπτη σε περίπτωση που οι διακοπές είχαν ενεργοποιηθεί νωρίτερα) και συνεχίζουμε τη μέτρηση. Αν το MSB της θύρας εισόδου είναι άσσος, ενεργοποιούμε τις διακοπές, όπως απαιτεί η εκφώνηση.

Όσον αφορά τη ρουτίνα εξυπηρέτησης διακοπής, έχει τον σχετικά απλό ρόλο του να αυξάνει τον **H** κατά ένα, εκτός κι αν έχει φτάσει στο 15, οπότε και μηδενίζεται. Όμως υπάρχει και το πρόβλημα της διπλής διακοπής. Αυτό αντιμετωπίστηκε μέσω του καταχωρητή **L**, ο οποίος και επιτρέπει την εξυπηρέτηση της διακοπής μόνο σε περίπτωση που έχει την τιμή μηδέν--που σημαίνει ότι βρισκόμαστε στην πρώτη από το ζεύγος των διακοπών. Η δεύτερη αγνοείται.

Το μόνο πρόβλημα που δεν αντιμετωπίστηκε ήταν αυτό της "μνήμης" που παρουσίασε το σύστημα, σε περίπτωση κλειστού διακόπτη και πρόκλησης διακοπής. Τότε, όταν το MSB της θύρας εισόδου ενεργοποιούνταν, η μέτρηση των διακοπών αυξανόταν κατά ένα.

Ακολουθεί ο κώδικας της άσκησης.

```
                              twotwoa
BEGIN: IN 10H               ;interrupts are initially disabled
       MVI H,00H            ;here we save the interrupt counter!
       MVI L,00H            ;a safety flag
       MVI B,00H
       MVI C,64H            ;for the DELB routine
AGAIN: MVI A,00H
LOOPA: CPI 0FH              ;reached a limit?
       JZ AGAIN
       MOV D,A              ;this checks the switches' MSB
       LDA 2000H
       RLC
       JNC DROP
       MVI A,0DH
       SIM
       EI                   ;if MSB is set, enable 6.5 interrupts
STAND: MOV A,H              ;...and properly modify output
       RLC                  ;(interrupt count to the left,
       RLC                  ;normal count to the right)
       RLC
       RLC
       ADD D
       CMA                  ;inverse logic
       STA 3000H
       MVI A,0FH            ;no interrupts wanted during DELB!
       SIM
       DI
       CALL DELB
```

```
       MOV A,D              ;retrieve normal count
       INR A
       JMP LOOPA
DROP:  MVI A,0FH
       SIM
       DI
       JMP STAND


INTR_ROUTINE:

       PUSH D
       PUSH PSW             ;we don't save HL because we NEED H
       MOV A,L              ;check the flag
       CPI 01H
       JZ FLAG
       INR L                ;update the flag
       MOV A,H
       CPI 0FH
       JZ INIT
       INR H
BACK:  POP PSW
       POP D
       RET                  ;we don't enable interrupts again
INIT:  MVI H,00H            ;trusting the trick above!
       JMP BACK
FLAG:  DCR L                ;reinitialize flag
       JMP BACK

       END
```

Άσκηση 2ii (β)

Η λογική της ενημέρωσης των LED παρέμεινε ίδια με την προηγούμενη άσκηση. Εδώ δεν χρειάστηκε καταχωρητής σημαία. Η μόνη διαφορά έγκειται στη ρουτίνα εξυπηρέτησης.

Συγκεκριμένα, ο συσσωρευτής αποκτά την τιμή της θύρας εισόδου. Ο **H** μηδενίζεται. Ρολάρουμε ένα-ένα τα bits του συσσωρευτή αριστερά, και για κάθε ένα από αυτά που είναι άσσος, αυξάνουμε το περιεχόμενο του **H** κατά ένα. Κατόπιν γίνεται κατάλληλα επιστροφή στο κυρίως πρόγραμμα.

Ακολουθεί ο κώδικας της άσκησης. Στην παραδοτέα εργασία επισυνάπτουμε και το εκτελέσιμο αρχείο.

```
                            twotwob
BEGIN: IN 10H               ;interrupts are initially disabled
       MVI B,00H
       MVI C,64H            ;for the DELB routine
AGAIN: MVI A,00H
LOOPA: CPI 0FH              ;reached a limit?
       JZ AGAIN
       MOV D,A
       MVI A,0DH
       SIM
       EI
STAND: MOV A,H              ;...and properly modify output
       RLC                  ;(interrupt count to the left
       RLC                  ;and normal count to the right)
       RLC
       RLC
       ADD D
```

```
       CMA                 ;inverse logic
       STA 3000H
       CALL DELB
       MOV A,D             ;retrieve normal count
       INR A
       JMP LOOPA
DROP:  MVI A,0FH
       SIM
       DI
       JMP STAND


INTR_ROUTINE:

       PUSH B
       PUSH D
       PUSH PSW            ;we don't save HL because we NEED H
       MVI H,00H           ;initialize switch counter
       LDA 2000H
       RLC                 ;check the MSB
       JNC SEC             ;if it's zero, don't increase the count
       INR H
SEC:   RLC                 ;check next bit
       JNC THIR
       INR H
THIR:  RLC
       JNC FOU
       INR H
FOU:   RLC
       JNC FIF
       INR H
FIF:   RLC
       JNC SIX
       INR H
SIX:   RLC
       JNC SEVE
       INR H
SEVE:  RLC
       JNC LAST
       INR H
LAST:  RLC
       JNC BACK
       INR H
BACK:  POP PSW
       POP D
       POP B
       RET                 ;we don't enable interrupts again

       END
```

ΤΕΛΟΣ ΕΡΓΑΣΙΑΣ