1η Εργαστηριακή Άσκηση

για το μάθημα "Εργαστήριο Μικροϋπολογιστών"

*21-10-2017*

Ομάδα Γ 12

Αρβανίτης Χρήστος 03114622

Μπαγάκης Εμμανουήλ 03114157

Μπαϊρακτάρη Κωνσταντίνα 03114018

Άσκηση 1

Για την άσκηση αυτή, κατασκευάσαμε ένα χρονόμετρο δευτερολέπτων που απεικονίζει το χρόνο σε δυαδική μορφή πάνω στα LEDs εξόδου του μLab. Tα LEDs βρίσκονται στην διεύθυνση 3000Η της μνήμης. Ο μετρητής έχει άνω όριο την δυαδική τιμή x που δίνεται από τα 4 αριστερότερα dip switches της θέσης μνήμης 2000Η. Μετράει άνω μέχρι την τιμή x, ενώ στην συνέχεια μετράει κάτω μέχρι το 0. Η διαδικασία αυτή επαναλαμβάνεται συνεχώς. Το άνω όριο x μπορεί να αλλάξει ανά πάσα στιγμή της εκτέλεσης. Στην περίπτωση που ο μετρητής βρίσκεται πάνω από την νέα τιμή του x, ξεκινάει τη μέτρηση προς τα κάτω. Επιπλέον, αν ο μετρητής βρίσκεται κάτω από την νέα τιμή του x, η τρέχουσα μέτρηση (είτε άνω, είτε κάτω) συνεχίζεται με όριο το νέο x. Η τιμή του μετρητή αλλάζει ανά 1 sec. Αυτό επιτυγχάνεται με τη ρουτίνα DELB που έχει ως όρισμα το διπλό καταχωρητή B-C.

IN 10H

LXI B,03E8H

MVI D,00H ; D is the counter

UP: LDA 2000H ;dip switches input

RRC

JNC UP ;if lsb = 0 wait until lsb = 1

RRC

RRC

RRC

ANI 0FH ;4 msbs

MOV E,A ;E is the upper boundary

CMP D

JZ DOWN

JC DOWN ;if E <= D then count down

INR D ;increase counter

MOV A,D

CMA

STA 3000H ;show counter

CALL DELB ;1 sec delay

JMP UP

DOWN: LDA 2000H ;dip switches input

RRC

JNC DOWN ;if lsb = 0 wait until lsb = 1

RRC

RRC

RRC

ANI 0FH

MOV E,A

MOV A,D ; if D=0 count up

CPI 00H

JZ UP

DCR D ;decrease counter

MOV A,D

CMA

STA 3000H ;show counter

CALL DELB ;1 sec delay

JMP DOWN

HLT

END

Άσκηση 2

Σε αυτή την άσκηση υλοποιήσαμε ένα μετρητή άνω από το 0 έως το 15. Η μέτρηση εμφανίζεται στα 4 MSB των LEDs (αυτό επιτυγχάνεται προσθέτοντας το 10Η σε κάθε βήμα μέτρησης). Το πρόγραμμα επιτρέπει διακοπές μόνο όταν το LSB των dip switches έχει την τιμή 1 (φόρτωση εισόδου στον Α, ολίσθηση και έλεγχος κρατουμένου). Το πλήθος των διακοπών mod 16 εμφανίζεται στο δεξιότερο 7-segment σε δεκαεξαδική μορφή. Για την απενεργοποίηση των διακοπών δεν χρησιμοποιούμε το DI αλλά έχουμε συνεχώς ενεργοποιημένες τις διακοπές RST6.5 και στην αρχή της ρουτίνας εξυπηρέτησης ελέγχουμε το LSB των dip switches. Μετράμε διπλάσιες διακοπές, αφού ο TSIK δέχεται δύο σήματα διακοπών σε κάθε πάτημα του πλήκτρου INTR και στη συνέχεια διαιρούμε το πλήθος δια δυο (shift δεξιά κατά μία θέση). Η καθυστέρηση αυτή τη φορά υλοποιείται με τη ρουτίνα CUSTOM\_DELAY, η οποία δεν χρειάζεται ορίσματα και διαρκεί 50 msec. Απαιτείται καθυστέρηση 100 msec σε κάθε βήμα του μετρητή άνω οπότε χρησιμοποιούνται δύο κλήσεις της CUSTOM\_DELAY. Αυτή η υλοποίηση της χρονοκαθυστέρησης διευκολύνει την εμφάνιση του πλήθους των διακοπών αφού η ρουτίνα DCD (υπεύθυνη για την εμφάνιση στο 7-segment) δεν καλείται στο εσωτερικό της ρουτίνας χρονοκαθυστέρησης. Για την εμφάνιση στα 7-segment αποθηκεύουμε τις κατάλληλες τιμές στις θέσεις μνήμης 0Β30Η - 0Β35Η και στη συνέχεια μέσω του διπλού καταχωρητή D-Ε και της STDM μεταφέρουμε τις τιμές στις απαιτούμενες θέσεις μνήμης για την εμφάνιση στα 7-segment.

IN 10H

;Αρχικοποίηση τιμών για τύπωμα στα 7 seg displays στο κενό(αρχικά σβηστά displays)

MVI A,00H

STA 0B30H

MVI A,10H

STA 0B31H

STA 0B32H

STA 0B33H

STA 0B34H

STA 0B35H

LXI D,0B30H

CALL STDM

CALL DCD

MVI C,02H ;Μετρητής πλήθους διακοπών. Αρχικά 1 αφού στο πρώτο print δεν θα αυξηθεί ο C

MVI B,00H ;Στον B το current output της μέτρησης στα LED (3000H)

MVI A,0DH ;Μάσκα ώστε να επιτρέπονται διακοπές RST 6.5 (Αυτές προκαλούνται από το INTRPT)

SIM

EI

JMP LOOP1 ;Άλμα στο κυρίως πρόγραμμα

CUSTOM\_DELAY:

PUSH PSW

LXI D,0032H ;Στο ζεύγος B-C ο χρόνος για την DELB 100ms

MSEC: MVI A,8CH

NOP

NOP

LOOPDELAY: DCR A

JNZ LOOPDELAY

NOP

DCX D

MOV A,D

ORA E

JNZ MSEC

POP PSW

RET

;Διαδικασία με την οποία υπολογίζεται το MOD του πλήθους των Intrpt's με το 16 και τυπώνεται στο δεξιότερο 7 segment display

CALC\_16\_MODULO:

PUSH PSW

MOV A,C ;Στον C το πλήθος διακοπών\*2 λόγω bug του TSIK

RRC ;Shift προς τα αριστερά ώστε να διαιρέσουμε το πλήθος των διακοπών με το 2 αφού σε κάθε πάτημα υπάρχει bug και παράγονται 2 διακοπές

ANI 0FH ;Το υπόλοιπο με το 16 είναι ίσο με τα 4 LSB ψηφία του πλήθους

STA 0B30H ;Μεταφορά αποτελεσμάτων στη θέση για τη DCD

POP PSW

RET

INTR\_ROUTINE:

PUSH PSW

LDA 2000H ;Αν το LSB των dip switches είναι off τότε δεν αντιλαμβάνομαι διακοπές(υπάρχει γιατί σε κάθε φορά μας κρατούσε μία διακοπή)

RAR

JNC EXIT\_INT

CALL CALC\_16\_MODULO

LEAVE\_INT:

INR C ;Αύξηση του πλήθους των διακοπών σε κάθε έγκυρη διακοπή

EXIT\_INT:

POP PSW

EI

RET

;Κυρίως πρόγραμμα (το λέμε LOOP1 αφού έτσι και στο βιβλίο

LOOP1:

MOV A,B ;Μεταφέρω στα LEDs την τρέχουσα έξοδο

CMA

STA 3000H

CMA

ADI 10H

MOV B,A ;Κρατάμε το current output από τον accumulator

;Αντί για μια μεγάλη καθυστέρηση κάνουμε 2 μικρότερες ώστε να μην διακόπτεται το τύπωμα στα segments

CALL CUSTOM\_DELAY

LXI D,0B30H ;Εδώ ξεκινάνε τα δεδομένα προς τα displays

CALL STDM

CALL DCD

CALL CUSTOM\_DELAY

JMP LOOP1 ;Συνεχές μέτρημα

END

Άσκηση 3

Για την τρίτη άσκηση, υλοποιήσαμε ένα πρόγραμμα το οποίο δέχεται έναν διψήφιο δεκαεξαδικό αριθμό από το πληκτρολόγιο και τον μετατρέπει σε δεκαδικό. Το αποτέλεσμα εμφανίζεται στα τρία αριστερότερα 7-segment displays. Για την είσοδο από το πληκτρολόγιο καλούμε τη ρουτίνα KIND και στη συνέχεια ελέγχουμε αν πατήθηκε έγκυρο δεκαεξαδικό ψηφίο, αλλιώς συνεχίζουμε την ανάγνωση μέχρι να δεχθούμε δύο έγκυρα ψηφία. Το πρώτο ψηφίο τοποθετείται στα 4 MSB του καταχωρητή με ολίσθηση αριστερά κατά 4 ενώ το δεύτερο ψηφίο τοποθετείται στα 4 LSB με πρόσθεση στον καταχωρητή. Η μετατροπή σε δεκαδικό εκτελείται με διαδοχικές αφαιρέσεις, αρχικά του 100(64Η) ώστε να μετρηθούν οι εκατοντάδες και στη συνέχεια του 10(0ΑΗ), από το υπόλοιπο της προηγούμενης πράξης, ώστε να μετρηθούν οι δεκάδες. Το υπόλοιπο αυτής της διαδικασίας είναι οι μονάδες του δεκαδικού αριθμού προς εμφάνιση. Το πλήθος των μονάδων, δεκάδων και εκατοντάδων αποθηκεύονται σε τρείς διαδοχικές θέσεις μνήμης ώστε να εμφανιστούν στα 7-segment displays. Το πρόγραμμα είναι συνεχούς λειτουργίας.

IN 10H ; protection disabled

MVI A,10H ; 7-segment blank

STA 0B00H

STA 0B01H

STA 0B02H

STA 0B03H

STA 0B04H

STA 0B05H

LXI D,0B00H

CALL STDM

CALL DCD

SUB A ; A = 0

START1: ;

CALL DCD ; appear 0B00-0B05 to 7-segments

CALL KIND ; wait for keyboard input , store it on acc

CPI 10H ; is input in range 0-F ?

JNC START1 ; if not jump to start1 , read another

MOV B,A ; if input is valid store first input on B

START2:

CALL KIND ; read the second input

CPI 10H ; is second input valid ?

JC START\_CONV ; if so , start the conversion

JMP START2 ; if not , read another input

START\_CONV: ; store input number 16\*x + y

MOV C,A ; C

MOV A,B ; store x on 4lsb of acc , then shift it 4 times so 16\*x appears

RLC

RLC

RLC

RLC

ADD C ; add C (y) to acc (16\*x) -> acc has 16\*x + y

HEX\_TO\_DEC: ; C will be used to store hundreds , B tens , A units

MVI B,00H

MVI C,00H

HUN\_COUNTER: ; hundred counter

CPI 64H ; if A is over 64H (100dec) hundreds++; A is A-64H

JC DEC\_COUNTER

INR C

SUI 64H

JMP HUN\_COUNTER

DEC\_COUNTER: ; if A is over 0AH ( 10dec) tens++; A is A-0AH

CPI 0AH

JC PRINT

INR B

SUI 0AH

JMP DEC\_COUNTER

PRINT:

STA 0B03H ; store to 3rd msb 7-segment A which has units

MOV A,B

STA 0B04H ; store to 2nd msb 7-segment B which has tens

MOV A,C

STA 0B05H ; store to msb 7-segment C which has hundreds

LXI D,0B00H ; starting memory point

CALL STDM ; transfer 0B00-0B05 to 7-segments

CALL DCD

JMP START1 ; wait for new input

END

Άσκηση 4

Για αυτή την άσκηση υλοποιείται πρόγραμμα εξομοίωσης ενός αυτοματισμού βαγονέτου. Τη θέση του βαγονέτου δείχνει το αναμμένο LED εξόδου. Κάθε κίνηση έχει καθυστέρηση 0,5 sec η οποία υλοποιείται με τη ρουτίνα DELB. Η κίνηση του βαγονέτου ελέγχεται από τον LSB διακόπτη της θύρας 20Η (ΟΝ: κίνηση, OFF: στάση). Η κίνηση αλλάζει φορά όταν το βαγονέτο φτάσει σε ένα από τα δύο άκρα. Στην υλοποίησή μας, η φορά ελέγχεται από έναν καταχωρητή ο οποίος μπορεί να μεταβληθεί και από την πρόκληση διακοπής RST6.5 μόνο αν το βαγονέτο βρίσκεται σε κίνηση. Όταν το βαγονέτο είναι σταματημένο, αν και οι διακοπές είναι ενεργοποιημένες δεν αλλάζει η κατεύθυνση της κίνησης. Για να ξεπεράσουμε το bug του TSIK το οποίο δίνει δυο σήματα διακοπή σε κάθε πάτημα του πλήκτρου INTR, μετράμε το πλήθος των διακοπών έτσι ώστε μόνο οι ζυγές διακοπές να αλλάζουν την κατεύθυνση.

IN 10H

MVI A,0DH ;Μάσκα ενεργοποίησης διακοπών

SIM

EI ;Εφαρμογή μάσκας

LXI B,01F4H ;Στο ζεύγος B-C ο χρόνος για την DELB

MVI L,00H ;Αρχικοποίηση flag (L=0-> ΔΕΞΙΑ |-| L=1-> ΑΡΙΣΤΕΡΑ)

MVI H,00H

JMP MAIN

INTR\_ROUTINE:

PUSH PSW

IN 20H

RRC

JNC LEAVE\_INT

INR H

MOV A,H

RRC

JC LEAVE\_INT

MOV A,L ;Έλεγχος του flag κίνησης

CPI 00H

JZ GO\_LEFT ;Aν είναι 0 τότε κινούμαι προς τα δεξιά αλλιώς προς τα αριστερά

MVI L,00H ;Αλλαγή πορείας προς τα δεξιά

JMP LEAVE\_INT

GO\_LEFT:

MVI L,01H ;Αλλαγή πορείας προς τα αριστερά

LEAVE\_INT:

POP PSW

EI

RET

ENABLER:

MVI A,00H ;Μάσκα απενεργοποίησης διακοπών αφού το DI δεν λειτουργει ορθα εδω

SIM

EI ;Apply της μάσκας με το EI

LOOP1: IN 20H ;Loop μέχρι να γίνει πάλι ON o LSB διακόπτης

RAR

JNC LOOP1 ;check LSB for ON

MVI A,0DH ;Μάσκα ενεργοποίησης διακοπών

SIM

EI

RET

MAIN: MVI A,FEH ;Αρχικά αναμμένο το LSB Led και το βαγόνι στη δεξιότερη θέση

OUT 30H

MOV D,A ;Αποθήκευση της τρέχουσας θέσης του βαγονιού στον D

CALL DELB

MOVE\_LEFT\_INIT: MVI L,01H ;Το flag (καταχωρητής L) δείχνει αριστερή κίνηση

MOVE\_LEFT:

MOV A,L ;Αν το flag έγινε 0 (από αλλαγή διεύθυνσης με INTR) τότε κίνηση προς τα δεξιά

CPI 00H

JZ MOVE\_RIGHT\_INIT ;Παράλληλα, αλλαγή του flag για να δείχνει πλέον δεξιά κίνηση

CALL ENABLER ;Έλεγχος του LSB switch

MOV A,D

CPI 7FH ;Αν το βαγόνι στην αριστερότερη θέση τότε άλλαξε κατεύθυνση

JZ MOVE\_RIGHT\_INIT ;Παράλληλα, αλλαγή του flag για να δείχνει πλέον δεξιά κίνηση

RLC

OUT 30H ;Μετακίνηση στην επόμενη θέση

MOV D,A ;Αποθήκευση νέας θέσης ώς τρέχουσα

CALL DELB ;Καθυστέρηση 0.5 sec

JMP MOVE\_LEFT ;Συνέχεια κίνησης μεχρι INTR ή άφιξη σε άκρο

MOVE\_RIGHT\_INIT: MVI L,00H ;Το flag δείχνει δεξιά κίνηση

MOVE\_RIGHT:

MOV A,L ;Αν το flag έγινε διάφορο του 0 τότε κίνηση προς τα αριστερά

CPI 00H

JNZ MOVE\_LEFT\_INIT ;Παράλληλα, αλλαγή του flag για να δείχνει πλέον αριστερή κίνηση

CALL ENABLER ;Έλεγχος του LSB switch

MOV A,D

CPI FEH ;Αν το βαγόνι στη δεξιότερη θέση τότε άλλαξε κατεύθυνση

JZ MOVE\_LEFT\_INIT ;Παράλληλα, αλλαγή του flag για να δείχνει πλέον αριστερή κίνηση

RRC

OUT 30H ;Μετακίνηση στην επόμενη θέση

MOV D,A ;Αποθήκευση νέας θέσης ως τρέχουσα

CALL DELB ;Καθυστέρηση 0.5 sec

JMP MOVE\_RIGHT ;Συνέχεια κίνησης μέχρι INTR ή άφιξη σε άκρο

END