



Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2019-2020

Συστήματα Μικροϋπολογιστών, Ροή Υ, εξάμηνο 6^ο

4^η Σειρά Αναλυτικών Ασκήσεων – 1^η Εργαστηριακή Άσκηση

Στοιχεία φοιτητών:

Όνομα: Μαντζόυτας Ανδρέας, ΑΜ: 03117108, Εξάμηνο: 6^ο

Όνομα: Μιχελής Αναστάσιος, ΑΜ: 03117143, Εξάμηνο: 6^ο

Όνομα: Κωστανίκος Ιωάννης, ΑΜ: 03117082, Εξάμηνο: 6^ο

Ημερομηνία Παράδοσης: 28/05/2020

Ζήτημα 2.1

Για την υλοποίηση αυτής της άσκησης, αρχικά αρχικοποιήσαμε την έξοδο PORTB στην τιμή 11111111, ενώ την είσοδο, την θύρα PORTC, στην τιμή 11111011, ώστε να απομονώσουμε το 3^ο bit. Έπειτα, αρχικοποιήσαμε τον καταχωρητή r26, που χρησιμοποιούμε ως έξοδο στην τιμή 1. Όσο δεν έχει φτάσει το r26 στην τιμή 128, κάνουμε περιστροφή αριστερά, και ανάλογα εργαστήκαμε για την περίπτωση που κινούμαστε προς τα δεξιά.

Για τον έλεγχο υπερχείλισης συγκρίνουμε την τιμή του r26 με το 128. Όσο δεν είναι 128, συνεχίζει την λούπα, ώστε να κινείται προς τα αριστερά. Όταν κινείται προς τα δεξιά, συγκρίνουμε την τιμή του r26 με το 2. Όσο είναι μεγαλύτερη ή ίση του 2, συνεχίζουμε τη λούπα και κινούμαστε δεξιά, ενώ όταν φτάσει στην τιμή 1, μπαίνουμε στην ρουτίνα move_left.

Σε περίπτωση που πατηθεί ο διακόπτης PC2, μεταφερόμαστε στη ρουτίνα freeze, όπου δεν κάνει τίποτα έως ότου ξανακλείσει. Για να ελέγχουμε την πορεία που θα συνεχίσει το LED, μετά το «πάγωμα», κρατάμε μία τιμή στον καταχωρητή r28, και έπειτα ανάλογα με την τιμή του r28, θα συνεχίσει την ανάλογη πορεία(δεξιά ή αριστερά).

Παραθέτουμε τον κώδικα της άσκησης, ο οποίος και επισυνάπτεται.

```
.include "m16def.inc"

reset:
    ldi r24, low(RAMEND) ; initialize stack pointer
    out SPL, r24
    ldi r24, high(RAMEND)
    out SPH, r24
    ser r24 ; initialize PORTB for output
    out DDRB, r24
    ldi r26, 251 ;11111011 -> C
    out DDRC, r26
    ldi r26, 1 ;arxikopoihsh r26

move_left:
    ldi r28,1 ;san flag sthn pause
    in r27, PINC
    cpi r27, 4
    breq freeze
    out PORTB, r26
    ldi r24, low(500)
    ldi r25, high(500) ;delay 0.5 sec
    ;rcall wait_msec
    nop
    lsl r26
    cpi r26, 128
    brlo move_left ;elegxei r26 etsi wste na sunexisei h move left

move_right:
    ldi r28,2
```

```

in r27, PINC
cpi r27, 4
breq freeze
out PORTB, r26
ldi r24, low(500)
ldi r25, high(500) ;delay 0.5 sec
;rcall wait_msec
nop
lsl r26
cpi r26, 2
brsh move_right ;elegxei r26 etsi wste na sunexisei h move right
rjmp move_left

freeze:
    in r27, PINC
    cpi r27, 4
    breq freeze
    cpi r28, 1
    breq move_left
    rjmp move_right

wait_msec:
    push r24 ; 2 κύκλοι (0.250 μsec)
    push r25 ; 2 κύκλοι
    ldi r24 , low(998) ; φόρτωσε τον καταχ. r25:r24 με 998 (1 κύκλος - 0.125 μsec)
    ldi r25 , high(998) ; 1 κύκλος (0.125 μsec)
    rcall wait_usec ; 3 κύκλοι (0.375 μsec), προκαλεί συνολικά καθυστέρηση 998.375
        ; μsec
    pop r25 ; 2 κύκλοι (0.250 μsec)
    pop r24 ; 2 κύκλοι
    sbiw r24 , 1 ; 2 κύκλοι
    brne wait_msec ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
    ret ; 4 κύκλοι (0.500 μsec)

wait_usec:
    sbiw r24 ,1 ; 2 κύκλοι (0.250 μsec)
    nop ; 1 κύκλος (0.125 μsec)
    brne wait_usec ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
    ret ; 4 κύκλοι (0.500 μsec)

```

Ζήτημα 2.2

Για την υλοποίηση αυτής της άσκησης, εργαστήκαμε ως εξής:

Για να πάρουμε την τιμή του A, απομονώσαμε το 1^ο bit του PINB, για το B απομονώνουμε το 2^ο bit του PINB και κάνουμε μία περιστροφή δεξιά, ώστε να μεταφερθεί στο LSB η τιμή που θέλουμε. Για τα C και D εργαζόμαστε αναλόγως.

Τέλος, υπολογίζουμε τις λογικές συναρτήσεις f0 και f1 και ανάλογα με τις τιμές, εμφανίζουμε την κατάλληλη έξιδο ως εξής(το f0 αντιστοιχεί στο 1^ο led και το f1 στο 2^ο):

- Αν f0 = 0 και f0=0, τότε θέλουμε τον PORTA να πάρει την τιμή 0(00), δηλαδή να μην ανάψει κανένα led.
- Αν f0 = 1 και f0=0, τότε η PORTA παίρνει την τιμή 1(01), δηλαδή ανάβει μόνο το 1^ο led.
- Αν f0=0 και f1=1, τότε PORTA = 2(10), δηλαδή να ανάψει μόνο το 2^ο led.
- Αν f0=f1=1, τότε PORTA = 3(11), δηλαδή να ανάψουν και τα 2 led.

Παραθέτουμε τον κώδικα που συγγράψαμε:

```
#include <avr/io.h>
char a, b,notb, c, notc, d, f0, f1;
int main(void)
{
    //initialize PORTA 0-1 for output and PORTB 0-3 for input
    DDRA=0x03;
    DDRB=0xF0;
    while(1) {
        a = PINB & 0x01;
        b = PINB & 0x02;
        b = b>>1;
        notb = ~b & 0x01;// & 0x01;
        c = PINB & 0x04;
        c = c>>2;
        notc = ~c & 0x01;
        d = PINB & 0x08;
        d = d>>3;
        f0 = ~((a & notb) | (b & notc & d));
        f0 = f0 & 0x01;
        f1 = ((a | c) &(b | d));
        if (f0==0 && f1==0){
            PORTA = 0;
        }
        else if (f0 == 1 && f1 == 0){
            PORTA = 1;
        }
        else if (f0==0 && f1 == 1){
            PORTA = 2;
        }
        else {
            PORTA = 3;
        }
    }
    return 0;
}
```

Ζήτημα 2.3

Για την υλοποίηση αυτής της άσκησης, αρχικοποιήσαμε τις πρώτες 4 θύρες του PORTA για είσοδο και όλες τις θύρες του PORTB για έξοδο. Έπειτα, χρησιμοποιήσαμε μία μεταβλητή, την x, για να υποδεικνύει ποιο led να ανάψει κάθε φορά. Έπειτα:

- Αν πατηθεί ο πρώτος διακόπτης, κάνουμε μία περιστροφή δεξιά. Αν βρισκόμαστε στο LSB, τότε πρέπει να ανάψει το MSB.
- Αν πατηθεί ο δεύτερος, κάνουμε περιστροφή αριστερά. Αντίστοιχα, αν βρισκόμαστε στο MSB, τότε πρέπει να ανάψει το LSB.
- Τέλος, αν πατηθεί ο τρίτος ή ο τέταρτος, ανάβει το LSB ή το MSB αντίστοιχα.

Για να δούμε σε ποια περίπτωση βρισκόμαστε, απομονώνουμε το αντίστοιχο bit από το PORTA. Επίσης, για να εκτελέσουμε την αντίστοιχη εντολή, θα πρέπει πρώτα να βεβαιωθούμε ότι έχει αφεθεί το κουμπί, άρα χρησιμοποιούμε ένα ατέρμονο loop, που σταματάει την εκτέλεση όταν αφεθεί το κουμπί.

```
#include <avr/io.h>

char x;

int main(void)
{
    //initialize PORTB as output and PORTA as input
    DDRA=0xF0;
    DDRB=0xFF;
    x = 1; //arxikopoisi gia na anapsei to led0 stin arxi
    while(1) {

        PORTB = x; //show result

        if ((PIN1 & 0x01) == 1){ // An patithei to proto koumpsi -> shift
right
            while ((PIN1 & 0x01) == 1);

            if (x== 1) { //an eimaste sto LSB -> pigaine sto MSB
                x = 128;
            }
            else {
                x = x >> 1; //allios shift deksia
            }
        }

        if ((PIN1 & 0x02) == 2){ // 2o koumpsi -> shift left
            while ((PIN1 & 0x02) == 2);

            if (x== 128) { //an eimaste sto MSB -> pigaine sto LSB
                x = 1;
            }
            else { //allios shift aristera
                x = x << 1;
            }
        }
    }
}
```

```
if ((PIN_A & 0x04) == 4){ // an patithei to 3o -> pigaine sto LSB
    while ((PIN_A & 0x04) == 4);

    x = 1;
}

if ((PIN_A & 0x08) == 8){ // an patithei to 4o -> pigaine sto MSB
    while ((PIN_A & 0x08) == 8);

    x = 128;
}

return 0;
}
```