ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

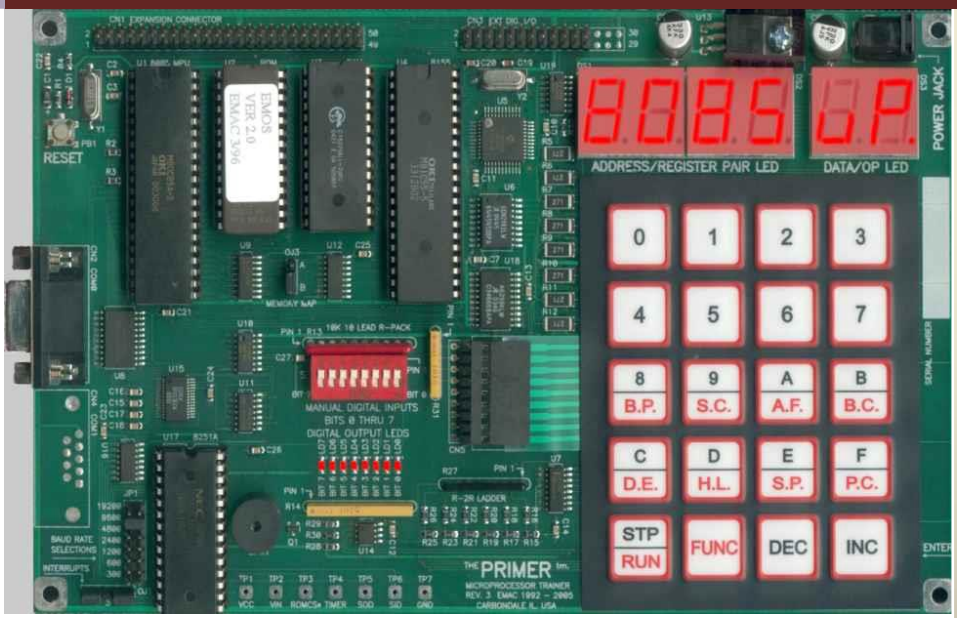
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

&

ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εργαστήριο Μικροϋπολογιστών

5η Σειρά Ασκήσεων



7ο Εξάμηνο

ΡΟΗ Υ

Αθανασίου Νικόλαος

ΑΜ 03112074

Βαβουλιώτης Γεώργιος

ΑΜ 03112083

Γιαννούλας Βασίλειος

ΑΜ 03112117

Άσκηση 1

Το παρακάτω πρόγραμμα απεικονίζει ένα αναμμένο led το οποίο κινείται στο Port B(PB0-PB7) από δεξία προς τα αριστερά και συνεχίζει κυκλικά(πηγαίνοντας από τα αριστερά προς τα δεξιά) παραμένοντας αναμμένο για 0,5 sec. Αυτό επιτυγχάνεται με τις ρουτίνες χρονοκαθυστέρησεων wait_usec και wait_msec . Η κίνηση αυτή ελέγχεται από το push button PA7 το οποίο όταν είναι πατημένο η κίνηση σταματά ενώ όταν επελευθερωθεί συνεχίζει ακριβώς απο εκει που σταμάτησε.

Assembly code :

```
.include "m16def.inc"
.def curr =r16
.def counter =r17

start:      ldi r24 , low(RAMEND)          ; initialize stack pointer
            out SPL , r24
            ldi r24 , high(RAMEND)
            out SPH , r24
            clr r24                        ; initialize PORTA for input
            out DDRA , r24
            ser r24                        ; initialize PORTB for output
            out DDRB , r24
            ;out PORTA , r24
            ldi curr,0x01
            out PORTB, curr
            ldi counter,0x07
            ldi r24 , low(500)             ; load r25:r24 with 500
            ldi r25 , high(500)            ; delay 0.5 second


left:       ldi r24 , low(500)             ; load r25:r24 with 500
            ldi r25 , high(500)            ; delay 0.5 second
            rcall wait_msec                ; led goes from right to left
            sbic PINA,0X07                 ; check if Pa7 = 1
            rjmp left                      ; if it is 1 stop until it gets 0...
            out PORTB, curr                ; show led
            lsl curr                       ; else move left
            dec counter                    ; until it reaches the (left) end
            breq right
            rjmp left

right:  ldi r24 , low(500)                 ; load r25:r24 with 500
            ldi r25 , high(500)            ; delay 0.5 second
            rcall wait_msec                ; led goes from left to right
            sbic PINA,0X07                 ; check if Pa7 = 1
            rjmp right                     ; if it is 1 stop until it gets 0...
            out PORTB, curr
            lsr curr                       ; else move right
```

```
                inc counter                 ; until it reaches the (right) end
                cpi counter,0x07
                breq left
                rjmp right


wait_usec:
                sbiw r 24 ,1                ; 2 κύκλοι (0.250 μsec)
                nop                         ; 1 κύκλος(0.125 μsec)
                nop                         ; 1 κύκλος(0.125 μsec)
                nop                         ; 1 κύκλος (0.125 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)

wait_msec:
                push r24                    ; 2 κύκλοι (0.250 μsec)
                push r25                    ; 2  κύκλοι
                ldi r24 , low(998)          ; φόρτωσε τον καταχωρητή
;r25:r24 με  την τιμη 998
                ldi r25 , high(998)         ; 1 κύκλοι (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)
```

Άσκηση 2

Στο πρόγραμμα αυτό αλλάζουμε τον παραπάνω κώδικα ώστε η καθυστέρηση στο άναμμα και στο σβήσιμο να δίνεται από τις τιμές των PB4-PB7 kai PB0-PB3 αντίστοιχα σύμφωνα με τον τύπο DELAY=(2x+1)*50, όπου x η 4bitη τιμή των switches κάθε φορά . Από τον παραπάνω τύπο συμπεραίνουμε ότι η μικρότερη καθυστέρηση είναι 50μsec ενώ η μεγαλύτερη είναι 1550μsec. Για να δημιουργήσουμε αυτή την καθυστέρηση καλούμε τις ρουτίνες χρονοκαθυστέρησης τόσες φορές όσες είναι και η τιμή της συνάρτηση DELAY για τη δεδομένη είσοδο από τα switches κάθε φορά. Για να διαμορφώσουμε την καταλληλη τιμή που πρεπει να έχει ο αντίστοιχος καταχωρητής που μας δείχνει την καθυστέρηση ακολουθούμε την εξής διαδικασία , έστω x η τιμή που πέρνουμε απο τους διακόπτες με αριστερή ολίσθηση κατά 1 θέση δημιουργούμε το 2*x και στην συνέχεια προσθέτουμε το 1.

Assembly code :

```
.include "m16def.inc"
.DEF temp=r16
.DEF c=r21

start:
        ldi r24 , low(RAMEND)        ; initialize stack pointer
        out SPL , r24
        ldi r24 , high(RAMEND)
        out SPH , r24
        clr temp                     ;orismos thuras eisodou
        out DDRB,temp
        dec temp
        out DDRA,temp                ;orismos thuras eksodou
        ;out PORTB,temp
        in temp,PINB                 ;diavasma apo eisodo B
        mov YL,temp                  ;ekxorhsh dedomenon eisodou st a1--4 lsb
                                     ;aforoun svhsimo
        lsr temp
        lsr temp
        lsr temp
        lsr     temp                 ;olisthhsh 4 theseis oste sta lsb na erthoun ta 4
msb
        mov XL,temp                  ;4 msb aforoun annama
        andi XL,15                   ;apomonosh 4 lsb --stoixeia anammatos
        andi YL,15                   ;apomonosh 4 lsb-- stoixeia svhsimatos
        lsl XL                        ;(h add XL,XL) praksh 2x g annama
        lsl YL                  ;(h add YL,YL)  praksh 2x g svhsimo
        inc XL                       ;praksh (2x+1) g anamma
        inc YL                       ;praksh (2x+1) g svhsimo
        ldi c,50                ;(2x+1)*50 msec
        mul c,XL                     ;r0 low, r1 high
        movw XL,r0                   ;r0-> a1(low), r1->a2(high)
        mul c,YL
        movw YL,r0                   ;r0-> b1(low), r1->b2(high)

flash:
   on:
                ser temp
                out PORTA,temp
                movw r24,XL          ;o diplos r24:r25 pairnei tin word tou XL
                rcall wait_msec      ;h kathusterhsh tha klhsei XL fores
```

```
off:
        clr temp
        out PORTA,temp
        movw r24,YL         ;o diplos r24:r25 pairnei tin word tou YL
        rcall wait_msec     ;kathusterhsh tha klhthei YL fores
        rjmp start

                            ;sunarthsh pou prokalei 1*(r24:r25) usec
                        kathusterhsh
wait_usec:
        sbiw r24 ,1
        nop                 ;kathe nop prokalei 0.125usec delay
        nop
        nop
        nop
        brne wait_usec
        ret

;sunarthsh pou kalei thn wait_usec gia na petuxei delay 1msec
;sugkekrimena prokalei delaey = 2.125usec(17 CC) + 998.375usec(h wait_usec me
eisodo 998) = 1.0005 msec
wait_msec:
        push r24
        push r25
        ldi r24 , low(998)
        ldi r25 , high(998)
        rcall wait_usec
        pop r25
        pop r24
        sbiw r24 , 1
        brne wait_msec
        ret
```

Άσκηση 3

Το παρακάτω πρόγραμμα σε C υλοποιεί την εξής διαδικασία : ανάβει αρχικά το led7 που είναι συνδεδεμένο στο bit 7 της Port B και στη συνέχεια με το πάτημα κατάλληλων κουμπίων της θύρας Port D , SW0-SW4 που αντιστοιχούν στα PD0-PD4 επιτελεί τις ακόλουθες λειτουργίες:

1. SW0 : δεξιά ολίσθηση του led
2. SW1 : αριστερή ολίσθηση του led
3. SW2 :δεξιά ολίσθηση του led κατα 2 θέσεις
4. SW3 :αριστερή ολίσθηση του led κατα 2 θέσεις
5. SW4 :μετακίνηση του led στην αρχική του θέση

(Όλες οι αλλαγές γίνονται με το άφημα των διακοπτών). Επιπλέον αν πατηθεί πάνω απο 1 διακόπτης ταυτοχρονα πρέπει να τηρείται η ακόλουθη προτεραιότητα → SW4,SW3,SW2,SW1,SW0.

C code :

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

void changeState (unsigned char input, unsigned char msbBit, unsigned char * state);

int main(void) {
	// I thira Port D tithetai ws eisodos
	DDRD = 0x00;
	// I thura Port B tithetai ws eksodos kai arxika anavei to LED 7
	DDRB = 0xFF;
	PORTB = 0x80;
	int flag[5];
	int i;
	for (i=0;i<5;i++){
			flag[i]=0;
	}


	// Arxikopoiisi katastasis LED
	unsigned char ledState = 0x80;
	while (1) {
		unsigned char input;
		input = PIND;		// Diavase to input apo ti thira D
		// Analoga me to poios diakoptis exei patithei tropopoiise tin katastasi
		if ((input & 0x10) == 0x10) {
			//changeState(input, input & 0x10, &ledState);
			flag[0]=1;
		}
		if ((input & 0x08) == 0x08) {
			//changeState(input, input & 0x08, &ledState);
			flag[1]=1;
		}
		if ((input & 0x04) == 0x04) {
			//changeState(input, input & 0x04, &ledState);
			flag[2]=1;
		}
		if ((input & 0x02) == 0x02) {
			//changeState(input, input & 0x02, &ledState);
			flag[3]=1;
		}
		if ((input & 0x01) == 0x01) {
			//changeState(input, input & 0x01, &ledState);
			flag[4]=1;
		}

		input=PIND;

		if (flag[0]==1 && ((input & 0x10) == 0x00)) {
			changeState(input, 0x10, &ledState);
```

```
                flag[0]=0;
        }
        else if (flag[1]==1 && ((input & 0x08) == 0x00)) {
                changeState(input, 0x08, &ledState);
                flag[1]=0;
        }
        else if (flag[2]==1 &&((input & 0x04) == 0x00)) {
                changeState(input, 0x04, &ledState);
                flag[2]=0;
        }
        else if (flag[3]==1 && ((input & 0x02) == 0x00)) {
                changeState(input, 0x02, &ledState);
                flag[3]=0;
        }
        else if (flag[4]==1 && ((input & 0x01) == 0x00)) {
                changeState(input,0x01, &ledState);
                flag[4]=0;
        }
        PORTB = ledState;
    }
    return 0;
}

void changeState (unsigned char input, unsigned char msbBit, unsigned char * state) {


    while ((PIND & msbBit) == msbBit); // Perimene mexri na afethei o piestikos
diakoptis
        switch (msbBit) {
            case 0x10: // SW4
                (* state) = 0x80;
                break;

            case 0x08: // SW3
                if ((* state) == 0x40) // Oriaki periptwsi
                    (* state) = 0x01;
                else if((*state)==0x80) //oriakh periptwsh 2
                    (*state) = 0x02;
                else    // alliws olisthisi
                    (* state) <<= 2;
                break;

            case 0x04: // SW2
                if ((* state) == 0x02) // Oriaki periptwsi
                    (* state) = 0x80;
                else if ((*state) == 0x01) //oriakh periptwsh 2
                    (*state) = 0x40;
                else    // alliws olisthisi
```

```
                        (* state) >>= 2;
                    break;

                case 0x02: // SW1
                    if ((* state) == 0x80) // Oriaki periptwsi
                        (* state) = 0x01;
                    else    // alliws olisthisi
                        (* state) <<= 1;
                    break;

                case 0x01: // SW0
                    if ((* state) == 0x01) // Oriaki periptwsi
                        (* state) = 0x80;
                    else    // alliws olisthisi
                        (* state) >>= 1;
                    break;
            }
        return;
}
```