Εργαστήριο Μικροϋπολογιστών
3η Άσκηση

Ομάδα: Δ12
Βακαλόπουλος Θεόδωρος, ΑΜ: 03114013
Μαυρομμάτης Ιάσων, ΑΜ: 03114771
Νικητοπούλου Δήμητρα, ΑΜ: 03114954

Άσκηση 1

```
.include "m16def.inc"
        .def reg = r16
        .def leds = r17     //Status of Leds
        .def flag = r18     //Determines the shift of Leds left or right
        ldi flag, low(RAMEND)
        out SPL, flag
        ldi flag, high(RAMEND)
        out SPH, flag
        clr flag            //Initialise flag = 0 => move left
        ldi leds, 0x01      //Initialise Leds
        clr reg
        out DDRA, reg   //Define PortA as input
        ser reg
        out DDRB, reg   //Define PortB as output
        start:
        out PORTB, leds
        ldi r25, HIGH(500)
        ldi r24, LOW(500)       //Initialise registers for 500ms delay
        rcall wait_msec
        sbis PINA, 0x00  //Read input
        rjmp start
        cpi flag, 0x00
        breq Move_Left
        cpi leds, 0x01
        breq right_border
        lsr leds
        rjmp start

        right_border:
        com flag
        lsl leds
        rjmp start

        Move_Left:
        cpi leds, 0x80
        breq left_border
        lsl leds
        rjmp start
```

```
left_border:
com flag
lsr leds
rjmp start

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

wait_usec:
sbiw r24 ,1
nop
nop
nop
nop
brne wait_usec
ret
```

Σχόλια:

- Ο καταχωρητής που ονομάζουμε flag κρατά πληροφορία για τη φορά της κίνησης και ,κατά συνέπεια, το περιεχόμενο του αντιστρέφεται όποτε φτάνουμε στις ακραίες θέσεις.

Άσκηση 2

```
.include "m16def.inc"
.def leds = r20
.def reg = r16
.def temp = r18
ldi reg, low(RAMEND)
out SPL, r24
ldi reg, high(RAMEND)
out SPH, r24
ser reg
out DDRB, reg              //PortB as output
clr reg
out DDRA, reg              //PortA as input
flash:
rcall on
in reg, PINA
andi reg, 0x0f             //Isolate PA0-PA3
//Delay
rcall calculate_Delay
rcall wait_msec

rcall off
in reg, PINA
```

```
andi reg, 0xf0
swap reg
//Delay
rcall calculate_Delay
rcall wait_msec

rjmp flash

on:
ser leds
out PORTB, leds
ret

off:
clr leds
out PORTB, leds
ret

calculate_Delay:
clr r24
clr r25
inc reg
ldi temp, 0xc8                  //Temp = 200d
mul reg, temp
mov r24, r0
mov r25, r1
ret

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

wait_usec:
sbiw r24 ,1
nop
nop
nop
nop
brne wait_usec
ret
```

Σχόλια:

- Μέσω της συνάρτησης Calculate Delay γίνεται ο υπολογισμός της τιμής της καθυστέρησης σύμφωνα με τη δεδομένη σχέση καθώς και η ανάθεση της τιμής αυτής στους καταχωρητές r24-r25 που χρησιμοποιούν οι ρουτίνες χρονοκαθυστέρησης.

Άσκηση 3

```
#include <avr/io.h>
unsigned char z,output;

int main(){
	DDRC=0x00; //eisodos
	DDRA=0xFF; //eksodos
	unsigned char
temp4,temp3,temp2,temp1,temp0,temp4_new,temp3_new,temp2_new,temp1_new,temp0_ne
w;
	output=0x80;
	PORTA=output;
	temp4 = 0;
	temp3 = 0;
	temp2 = 0;
	temp1 = 0;
	temp0 = 0;
	while(1){
		z=PINC;
		temp4_new=z&0x10;
		temp3_new=z&0x08;
		temp2_new=z&0x04;
		temp1_new=z&0x02;
		temp0_new=z&0x01;
		if( (temp4!=0) && (temp4_new==0) ){
			output=0x80;
			PORTA=output;
		}
		else if( (temp3!=0) && (temp3_new==0) ){
			if(output==0x40){
				output=0x01;
				PORTA=output;
			}
			else if(output==0x80){
				output=0x02;
				PORTA=output;
			}
			else{
				output=output<<2;
				PORTA=output;
			}
		}
		else if( (temp2!=0) && (temp2_new==0)){
			if(output==0x02){
				output=0x80;
				PORTA=output;
			}
			else if(output==0x01){
				output=0x40;
				PORTA=output;
			}
			else{
				output=output>>2;
				PORTA=output;
			}
		}
		else if( (temp1!=0) && (temp1_new==0) ){
			if(output==0x80){
				output=0x01;
```

```
                                        PORTA=output;
                                }
                                else{
                                        output=output<<1;
                                        PORTA=output;
                                }
                        }
                        else if( (temp0!=0) && (temp0_new==0) ){
                                if(output==0x01){
                                        output=0x80;
                                        PORTA=output;
                                }
                                else{
                                        output=output>>1;
                                        PORTA=output;
                                }
                        }
                        temp0=temp0_new;
                        temp1=temp1_new;
                        temp2=temp2_new;
                        temp3=temp3_new;
                        temp4=temp4_new;
                }
                return 0;
        }
```

Σχόλια:

- Για να εξασφαλίσουμε ότι οι αλλαγές γίνονται τη στιγμή που αφήνουμε το διακόπτη κρατάμε συνεχώς προηγούμενη και τρέχουσα κατάσταση κάθε διακόπτη και ελέγχουμε αν η προηγούμενη είναι 1 και η τωρινή 0.
- Η προτεραιότητα που μας ζητείται επιτυγχάνεται με τη σειρά με την οποία τίθενται οι έλεγχοι των διακοπτών.
- Όταν φτάνουμε στο όριο της μιας πλευράς φροντίζουμε με κατάλληλες συνθήκες να μεταφέρουμε το αναμμένο λεντ στο όριο της άλλης πλευράς.