ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ (MICROLAB)

2η Εργαστηριακη Αναφορα στο μαθημα "**ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ**" του 7ου Εξαμηνου

των φοιτητων της **ομάδας 17**,

Εμμανουήλ Αναστάσιου Σερλή, Α.Μ. 03118125
Ευάγγελου Περσόπουλου, Α.Μ.: 03121701

1η Άσκηση: -> micro_lab02_ex01.asm

Το ζητούμενο πρόγραμμα πραγματοποιεί την εξυπηρέτηση ρουτίνας διακοπής στην υλοποίηση του τροποποιημένου μετρητή του σχήματος 2.1. Συγκεκριμένα, η ρουτίνα διακοπής απαριθμεί το πλήθος των διακοπών-από 0 ως 31- όσο δεν είναι πατημένο το PD7 button. Σε αντίθεση περίπτωση, η εν λόγω απαρίθμηση παγώνει. Το δοσμένο πρόγραμμα έχει τροποποιημένη την ρουτίνα διακοπής του, ώστε να λαμβάνει υπόψιν και το φαινόμενο του σπινθηρισμού κατά το πάτημα του button διακοπής PD3 προσδίδοντας την κατάλληλη καθυστέρηση στην αρχή του interrupt routine.

```
.include "m328PBdef.inc"
.def counter=r19 ;counter register

.org 0x0
rjmp reset
.org 0x4
rjmp ISR1

;init stack pointer
ldi r24, LOW(RAMEND)
out SPL, R24
ldi r24, HIGH(RAMEND)
out SPH, r24


reset:
    ;Interrupt on rising edge of INT1 pin
    ldi r24, (1 << ISC11) | (1 << ISC10)
    sts EICRA, r24

    ;Enable the INT1 interrupt (PD3)
    ldi r24, (1 << INT1)
    out EIMSK, r24
    sei

    ;set portc as output
    ser r26
    out DDRC, r26

    ser r27
    out DDRB, r27

loop1:
    clr r26
loop2:
    out PORTB,r26 ;allagi sto PORTD mono

    ldi r24, low(16*500)
    ldi r25, high(16*500) ;set delay
    rcall delay_mS
```

```
    inc r26

    cpi r26, 16 ;compare r26 with 16
    breq loop1 ;restart if equal
    rjmp loop2 ;else start again

delay_mS: ;create ms delay
    ldi r23,249
loop_inn:
    dec r23
    nop
    brne loop_inn

    sbiw r24,1
    brne delay_mS

    ret

ISR1:
    push r25 ;push registers to stack
    push r24
    in r24, SREG
    push r24

check:
    ldi r24, (1 << INTF1) ;includes sparkle effect
    out EIFR, r24         ;at the start of int. routine
    ldi r24, low(16*5)
    ldi r25, high(16*5)
    rcall delay_mS ;5ms delay before interrupt routine
    in r24, EIFR
    cpi r24, 2
    breq check

    in r25, PIND ;PIND kanonika
    sbrs r25, 7 ;check for PD7 button
    rjmp telos
    cpi counter, 31
    brne Loop
    clr counter
loop:
    inc counter ;add to interrupt counter
    out PORTC, counter
telos:
    pop r24
    out SREG, r24
    pop r24
    pop r25
    reti
```

2η Άσκηση: -> micro_lab02_ex02a.asm και micro_lab02_ex02b.asm

α) Το ζητούμενο πρόγραμμα πραγματοποιεί την υλοποίηση ενός συνεχούς μετρητή από 0 ως και 31 με καθυστέρηση 600msec, τα αποτελέσματα του οποίου φαίνονται στην έξοδο του PORTC. Η υλοποίηση βασίζεται στο δοθέν πρόγραμμα Assembly του σχήματος 2.1 της εκφώνησης.

```
.include "m328PBdef.inc"
.equ FOSC_MHZ=16 ;operating freq of mC
.equ DEL_mS=600; ;600msec delay
.equ DEL_NU=FOSC_MHZ*DEL_mS

;init stack pointer
ldi r24, LOW(RAMEND)
out SPL, R24
ldi r24, HIGH(RAMEND)
out SPH, r24

;PORTC as output
ser r26
out DDRC, r26

loop1:
    clr r26
loop2:
    out PORTC,r26

    ldi r24, low(DEL_NU)
    ldi r25, high(DEL_NU) ;set delay
    rcall delay_mS

    inc r26

    cpi r26, 32 ;compare r26 with 32
    breq loop1 ;restart if equal
    rjmp loop2 ;else start again

delay_mS: ;delay routine
    ldi r23,249
loop_inn:
    dec r23
    nop
    brne loop_inn

    sbiw r24,1
    brne delay_mS

    ret
```

β) Το ζητούμενο πρόγραμμα τροποποιεί την ρουτίνα εξυπηρέτησης για το INT0, έτσι ώστε να μετρά το πλήθος των πατημένων PB buttons και να το καταγράφει στα LSB στην θύρας PORTC, μέσω κατάλληλων logical και operand shifts.

```
.include "m328PBdef.inc"
.org 0x0
rjmp reset
.org 0x2
rjmp ISR0

reset:
    ;init stack pointer
    ldi r24, LOW(RAMEND)
    out SPL, r24
    ldi r24, HIGH(RAMEND)
    out SPH, r24

    ;interrupt on the rising edge of INT1 pin
    ldi r26, (1<<ISC01) | (1 << ISC00)
    sts EICRA, r26

    ;Enable INT1
    ldi r26, (1<<INT0)
    out EIMSK, r26

    sei

IO_set:
    ;PORTB as input
    clr r27
    out DDRB, r27

    ;PORTC as output
    ser r27
    out DDRC, r27

    ldi r20, 0 ;output counter
    ldi r22, 6 ;bit counter in for loop

    clr r23 ;clr for main

main: ;default main
    nop
    nop
    rjmp main

ISR0:
    push r24
    in r24, SREG
```

```
    push r24
    push r21
    push r19
    push r22   ;save registers to stack

    clr r24
    out PORTC, r24 ;clear PORTC
    in r21, PINB ;read from PORTB ;a temp register
    com r21
loop:
    mov r19, r21
    andi r19,0x01 ;temp register
    cpi r19,0x01
    breq cont
loop_next:
    dec r22
    asr r21 ;shift right
    cpi r22,0 ;break or not for loop
    breq loop2
    rjmp loop
cont:
    inc r20   ;for loop operator
    rjmp loop_next
loop2:
    clr r19
    clr r21
    ldi r19, 0x00 ;for PORTB output
    ldi r21, 0x01 ;for shift left operator
loop2_next:
    cpi r20,0
    breq return
    or r19,r21
    dec r20
    lsl r21 ;shift left
    rjmp loop2_next
return:
    out PORTC, r19 ;output to portc

    pop r22
    pop r19
    pop r21
    pop r24
    out SREG, r24
    pop r24         ;restore registers from stack

    reti
```

3η Άσκηση: -> micro_lab02_ex03.asm και micro_lab02_ex03__.c

Το ζητούμενο πρόγραμμα προσομοιώνει τον έλεγχο ενός φωτιστικού σώματος- που αντιστοιχεί στο LSB Led του PORTB- μέσω του PD3 Button (INT1). Συγκεκριμένα, μόλις πατηθεί το PD3, ανάβει το PB0 και σβήνει με καθυστέρηση 4sec, εκτός αν ξαναπατηθεί στο ενδιάμεσο. Τότε, ανάβουν αμέσως όλα τα Leds του PORTB για 0.5sec και μετά ανανεώνεται η 4sec καθυστέρηση για το PB0. Για να γίνει αυτό, έχει δημιουργηθεί τροποποιημένη ρουτίνα delay-τόσο σε assembly όσο και σε c-ώστε να ελέγχεται τυχόν ενδιάμεση έξοδος από την ρουτίνα καθυστέρησης στην ρουτίνα εξυπηρέτησης του interrupt μέσω κατάλληλης μεταβλητής/καταχωρητή.

Assembly Program:

```
.include "m328PBdef.inc"
.org 0x0
rjmp reset
.org 0x4
rjmp ISR1

.equ FREQ=16 ;operating freq


reset:
    ;init stack pointer
    ldi r24, LOW(RAMEND)
    out SPL, r24
    ldi r24, HIGH(RAMEND)
    out SPH, r24

    ;interrupt on the rising edge of INT1 pin
    ldi r26, (1<<ISC11) | (1 << ISC10)
    sts EICRA, r26

    ;Enable INT1
    ldi r26, (1<<INT1)
    out EIMSK, r26

    sei ;enable interrupt flags

IO_set:
    ;PORTB as input
```

```
    ser r27
    out DDRB, r27
    ldi r22, 0x00 ;for portb
    ldi r21, 0x00 ;counter for interrupts/delay
    ldi r20, 0x00 ;flag for delay break

main:
    clr r20 ;clear delay break flag
    cpi r21, 1
    breq delays_1 ;check for first time interrupt
    cpi r21, 2
    brsh delays_2 ;check for intermediate interrupts
    clr r22 ;else all PORTB leds to zero
    out PORTB, r22
    rjmp main


delays_1:
    ldi r22, 0x01 ;flash LSB
    out PORTB, r22

    ldi r24, low(FREQ*4000) ;then wait for 4000ms
    ldi r25, high(FREQ*4000)
    rcall delay_mS ;delay 4sec

    clr r21
    rjmp main

delays_2:
    ldi r22, 255 ;flash all
    out PORTB, r22
    ldi r24, low(FREQ*500) ;then wait for 500ms
    ldi r25, high(FREQ*500)
    rcall delay_mS ;delay 500ms

    ldi r22, 0x01
    out PORTB, r22
    ldi r24, low(FREQ*4000) ;then wait for 4000ms
    ldi r25, high(FREQ*4000)
    rcall delay_mS

    clr r21 ;no more need for delay counter
    rjmp main

ISR1:
    inc r21 ;increase counter for delays
    ldi r20, 1 ;set delay register to 1
    reti

delay_mS: ;delay loop
    ldi r23,200  ;decrease r23 from 249 to 200 for more accurate delay
    clr r20
```

```
loop_inn:
    cpi r20, 1 ;check register right after interrupt on delay
    breq main
    dec r23
    brne loop_inn

    sbiw r24,1
    brne delay_mS

    ret
```

C Program:

```
#define F_CPU 16000000UL
#include<avr/io.h>
#include<avr/interrupt.h>
#include<util/delay.h>

unsigned char counter=0;
void delay_manual(unsigned int);
unsigned int c=0;
unsigned int a=0;
unsigned int b=0;
unsigned int flag=0;
unsigned int r23;
unsigned int r24;



ISR(INT1_vect) {
    flag=1; //check interrupt on delay routine
    counter++; //counter of interrupts
}

void delay_manual(unsigned int ms){
    c=0;
    r23=36; //needs to change for test on board
    r24=ms;
    flag=0;
    while(r24>0){
        if (flag==1) {
            break;
        }
        else{
            for (int i=0; i<r23; i++) {
                if (flag==1){
                    break;
                }
                else {
```

```
                    c++; //dummy instructions to create delay
                    PORTC=0x32;
                }
            }
        }
        r24--;
    }
}


int main(){

    EICRA=(1<<ISC11) | (1<<ISC10); //INT1-PD3
    EIMSK = (1 << INT1);
    sei(); // enable all interrupts

    DDRB=0xFF; //main output
    DDRC=0xFF; //outputs
    while(1){
        flag=0;
        if(counter==1){
            PORTB=0x01; //lsb of portb flashed
            delay_manual(4000);
            if (flag==1) {
                break;
            }
            else{
                counter=0;
            }
        }
        if (counter>1){
            PORTB=255;
             delay_manual(500); //flash all pins for 0.5sec
             PORTB=0x01;
             delay_manual(4000); //flash only LSB for 4sec
             if (flag==1) {
                break;
            }
            else{
                counter=0; //clear counter
            }
        }
        PORTB=0x00;
    }
    main();
}
```