Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2022-2023

ΑΘΗΝΑ, 26 Οκτωβρίου 2022

2η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ
ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"
Χρήση εξωτερικών διακοπών στον Μικροελεγκτή AVR

Αναφορά 2ης Εργαστηριακής Άσκησης

Ραπτόπουλος Πέτρος (el19145)
Σαφός Κωνσταντίνος (el19172)

Ζήτημα 2.1:

Παρακάτω φαίνεται ο κώδικας σε Assembly που υλοποιεί τα ζητούμενα της άσκησης (και των δύο ερωτημάτων).

Η ορθή λειτουργία του κώδικα έχει ελεγχθεί στο περιβάλλον προσομοίωσης MPLAB X, καθώς και στην αναπτυξιακή πλακέτα του εργαστηρίου.

Σημείωση: Ο ακριβής τρόπος λειτουργίας του προγράμματος υποδεικνύεται μέσω σχολίων σε εντολές του κώδικα.

```
.include "m328PBdef.inc"    ; ATmega328P microcontroller definitions

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

.equ FOSC_MHZ = 16                ; Microcontroller operating frequency in MHz
.equ DEL_BOUNCE_mS = 5            ; Delay in mS for bouncing effect
.equ DEL_mS = 500                 ; Delay in mS (valid number from 1 to 4095) for main
.equ DEL_NU=FOSC_MHZ*DEL_mS        ; delay_mS routine: (1000*DEL_NU+6) cycles
.equ DEL_NU_BOUNCE=FOSC_MHZ*DEL_BOUNCE_mS
.DEF temp=r20                     ; define temporary register
; watch out delay function uses r23
.DEF counter=r22                  ; define the interrupt counter register
.DEF mainCounter=r21              ; define the main's counter register

reset:
    ldi temp, low(RAMEND)          ;Initialize stack pointer
    out SPL, temp
    ldi temp, high(RAMEND)
    out SPH, temp

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

    ; Enable the INT1 interrupt (PD3)
    ldi temp, (1 << INT1)
    out EIMSK, temp
    sei           ; enable interrupts

    ; Init PORTD as input
    clr temp
    out DDRD, temp

    ; Init PORTB as output
    ser temp
    out DDRB, temp
    ; Init PORTC as output
    out DDRC, temp

    ; Init Count Register
    clr counter
    out PORTC, counter ; show counter

loop1:
    clr mainCounter
loop2:
    out PORTB, mainCounter
    ldi r24, low(DEL_NU)
    ldi r25, high(DEL_NU)         ; Set delay (number of cycles)
    rcall delay_mS
    inc mainCounter
    cpi mainCounter, 16           ; compare r26 with 16
    breq loop1
    rjmp loop2
```

```
; delay of 1000*F1 + 6 cycles (almost equal to 1000*F1 cycles)
delay_mS:
; total delay of next 4 instruction group = 1+(249*4-1) = 996 cycles
    ldi r23, 249
loop_inn:
    dec r23         ; 1 cycle
    nop             ; 1 cycle
    brne loop_inn   ; 1 or 2 cycles
    sbiw r24, 1     ; 2 cycles
    brne delay_mS   ; 1 or 2 cycles
    ret             ; 4 cycles


ISR1:
; interrupt routine for INT1
    push r24      ; save r24
    push r25      ; save r25
    push temp     ; save temp
    in temp, SREG ; save SREG
    push temp

bouncing:
    ldi temp, (1<<INTF1)          ; set interrupt
    out EIFR, temp                ; flag to 0
    ldi r24, low(DEL_NU_BOUNCE)
    ldi r25, high(DEL_NU_BOUNCE)
    rcall delay_mS
    in temp, EIFR
    andi temp, 0x02
    cpi temp, 0x02
    breq bouncing

    in temp, PIND
    ; check if PD7 is pressed
    andi temp, 0x80
    cpi temp, 0x00
    breq skip
    ; if pressed  dont increase
dont_freeze:
    inc counter
    cpi counter, 32
    brne skip
    clr counter
skip:
    out PORTC, counter ; show counter
    pop temp
    out SREG, temp  ; restore SREG
    pop temp      ; restore temp
    pop r25       ; restore r25
    pop r24       ; restore r24
    reti          ; return to callee
```

Ζήτημα 2.2:

Παρακάτω φαίνεται ο κώδικας σε Assembly που υλοποιεί τα ζητούμενα της άσκησης (και των δύο ερωτημάτων).

Η ορθή λειτουργία του κώδικα έχει ελεγχθεί στο περιβάλλον προσομοίωσης MPLAB X, καθώς και στην αναπτυξιακή πλακέτα του εργαστηρίου.

Σημείωση: Ο ακριβής τρόπος λειτουργίας του προγράμματος υποδεικνύεται μέσω σχολίων σε εντολές του κώδικα.

```
.include "m328PBdef.inc"    ; ATmega328P microcontroller definitions

.org 0x0
rjmp reset
.org 0x2
rjmp ISR0

.equ FOSC_MHZ = 16                ; Microcontroller operating frequency in MHz
.equ DEL_BOUNCE_mS = 5            ; Delay in mS for bouncing effect
.equ DEL_mS = 600                 ; Delay in mS (valid number from 1 to 4095) for main
.equ DEL_NU=FOSC_MHZ*DEL_mS        ; delay_mS routine: (1000*DEL_NU+6) cycles
.equ DEL_NU_BOUNCE=FOSC_MHZ*DEL_BOUNCE_mS
.DEF temp=r23                     ; define temporary register
; watch out delay function uses r23
.DEF counter=r22                  ; define the interrupt counter register
.DEF mainCounter=r21              ; define the main's counter register
.DEF temp2=r20                    ; define temporary register
.DEF temp3=r19                    ; define temporary register

reset:
    ldi temp, low(RAMEND)         ; Initialize stack pointer
    out SPL, temp
    ldi temp, high(RAMEND)
    out SPH, temp

    ; Interrupt on rising edge of INT1 pin
    ldi temp, (1 << ISC01) | (1 << ISC00)
    sts EICRA, temp

    ; Enable the INT1 interrupt (PD3)
    ldi temp, (1 << INT0)
    out EIMSK, temp
    sei           ; enable interrupts

    ; Init PORTD as input
    clr temp
    out DDRD, temp
    ; Init PORTB as input
    out DDRB, temp
    ; Init PORTC as output
    ser temp
    out DDRC, temp

loop1:
    clr mainCounter
loop2:
    out PORTC, mainCounter
    ldi r24, low(DEL_NU)
    ldi r25, high(DEL_NU)             ; Set delay (number of cycles)
    rcall delay_mS
    inc mainCounter
    cpi mainCounter, 32               ; compare r26 with 32
    breq loop1
    rjmp loop2

; delay of 1000*F1 + 6 cycles (almost equal to 1000*F1 cycles)
delay_mS:
; total delay of next 4 instruction group = 1+(249*4-1) = 996 cycles
    ldi r23, 249
loop_inn:
    dec r23              ; 1 cycle
    nop                  ; 1 cycle
    brne loop_inn        ; 1 or 2 cycles
    sbiw r24, 1          ; 2 cycles
    brne delay_mS        ; 1 or 2 cycles
    ret                  ; 4 cycles
```

```
ISR0:
; interrupt routine for INT1
    push r24        ; save r24
    push r25        ; save r25
    push temp       ; save temp
    in temp, SREG   ; save SREG
    push temp

bouncing:
    ldi temp, (1<<INTF0)          ; set interrupt
    out EIFR, temp                ; flag to 0
    ldi r24, low(DEL_NU_BOUNCE)
    ldi r25, high(DEL_NU_BOUNCE)
    rcall delay_mS                ; delay
    in temp, EIFR                 ; see if the interrupt
    andi temp, 0x01               ; bit is activated
    cpi temp, 0x01                ; if it is then bouncing
    breq bouncing                 ; is underway

    in temp, PINB   ; keep the port's state
    ldi temp2, 6    ; loop counter
    ldi temp3, 0    ; keep how many buttons
                    ; are pressed
countLoop:
    ror temp      ; rotate right and carry
                  ; the last bit
    brcs skip     ; if carry == 1 dont increase (reversed logic)
                  ; if a button is pressed we have 0
    inc temp3     ; increase button counter
  skip:
    dec temp2     ; loopCounter--
    brne countLoop

    ldi temp, 0x00  ; temp will store the output state
    ldi temp2, 0x01 ; temp2 will alter the bits of temp
outputLoop:
    cpi temp3, 0x00 ; check if the wanted number of leds
    breq cont       ; became on
    dec temp3
    or temp, temp2  ; turn current bit at logical 1
    lsl temp        ; rotate left
    rjmp outputLoop

  cont:
    lsr temp      ; last bit from outputLoop remained
                  ; zero, a shift to the right is needed
    out PORTC, temp
    ldi r24, low(500)
    ldi r25, high(500)
    rcall delay_mS  ; delay to see results clearly
    pop temp
    out SREG, temp  ; restore SREG
    pop temp        ; restore temp
    pop r25         ; restore r25
    pop r24         ; restore r24
    reti            ; return to callee
```

Ζήτημα 2.3:

Παρακάτω φαίνεται ο κώδικας σε Assembly αλλά και σε C που υλοποιεί τα ζητούμενα.

Η ορθή λειτουργία των κωδίκων έχει ελεγχθεί στο περιβάλλον προσομοίωσης MPLAB X, καθώς και στην αναπτυξιακή πλακέτα του εργαστηρίου.

Σημείωση: Ο ακριβής τρόπος λειτουργίας των προγραμμάτων υποδεικνύεται μέσω σχολίων σε εντολές του κώδικα.

Κώδικας σε Assembly

```
.include "m328PBdef.inc"    ; ATmega328P microcontroller definitions

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

.equ FOSC_MHZ = 16              ; Microcontroller operating frequency in MHz
.equ DEL_BOUNCE_mS = 5          ; Delay in mS for bouncing effect
.equ DEL_mS = 500               ; Delay in mS (valid number from 1 to 4095) for main
.equ DEL_mS_LAMP = 4000         ; Delay in mS (valid number from 1 to 4095) for main
.equ DEL_mS_ALL_LEDS = 500
.equ DEL_NU=FOSC_MHZ*DEL_mS     ; delay_mS routine: (1000*DEL_NU+6) cycles
.equ DEL_NU_BOUNCE=FOSC_MHZ*DEL_BOUNCE_mS
.equ DEL_NU_LAMP=FOSC_MHZ*DEL_mS_LAMP
.equ DEL_NU_ALL_LEDS=FOSC_MHZ*DEL_mS_ALL_LEDS
.equ DEL_NU_LIGHT2=FOSC_MHZ*3500
.DEF temp=r23                    ; define temporary register
; watch out delay function uses r23
.DEF counter=r22                 ; define the interrupt counter register
.DEF mainCounter=r21             ; define the main's counter register
.DEF flag=r20
reset:
    ldi temp, low(RAMEND)        ;Initialize stack pointer
    out SPL, temp
    ldi temp, high(RAMEND)
    out SPH, temp

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

    ; Enable the INT1 interrupt (PD3)
    ldi temp, (1 << INT1)
    out EIMSK, temp
    sei                           ; enable interrupts

    ; Init PORTD as input
    clr temp
    out DDRD, temp

    ; Init PORTB as output
    ser temp
    out DDRB, temp

    ; Init interrupt flag
    ldi flag, 0x00

main:
    rjmp main

; delay of 1000*F1 + 6 cycles (almost equal to 1000*F1 cycles)
delay_mS:
; total delay of next 4 instruction group = 1+(249*4-1) = 996 cycles
    ldi r23, 249
loop_inn:
    dec r23         ; 1 cycle
    nop             ; 1 cycle
    brne loop_inn   ; 1 or 2 cycles
    sbiw r24, 1     ; 2 cycles
    brne delay_mS   ; 1 or 2 cycles
    ret             ; 4 cycles


ISR1:             ; interrupt routine
                  ; for INT1
    pop temp      ; pop the program counter
    pop temp      ; saved in two stack places
    push temp     ; save temp
    in temp, SREG ; save SREG
    push temp
```

```
bouncing:
    ldi temp, (1<<INTF1) ; set interrupt
    out EIFR, temp       ; flag to 0
    ldi r24, low(DEL_NU_BOUNCE)
    ldi r25, high(DEL_NU_BOUNCE)
    rcall delay_mS
    in temp, EIFR
    andi temp, 0x02
    cpi temp, 0x02
    breq bouncing

    ; if flag == 0 then first interrupt
    cpi flag, 0x00
    brne light2

    inc flag
    sei              ; enable interrupts while in interrupt handler
    ldi temp, 0x01
    out PORTB, temp ;let there be light
    ldi r24, low(DEL_NU_LAMP)
    ldi r25, high(DEL_NU_LAMP)
    rcall delay_mS  ; delay 4 secs
    ; during that delay interrupts could take place.
    ; As a result the delay time will reset
    ldi temp, 0x00
    out PORTB, temp ; lights off
    rjmp cont

light2:
    inc flag
    sei             ; enable interrupts while in interrupt handler
    ldi temp, 0xFF
    out PORTB, temp ;light up all lamps
    ldi r24, low(DEL_NU_ALL_LEDS)
    ldi r25, high(DEL_NU_ALL_LEDS)
    rcall delay_mS  ; delay 0.5 secs
    ldi temp, 0x01
    out PORTB, temp ;let there be light
    ldi r24, low(DEL_NU_LIGHT2)
    ldi r25, high(DEL_NU_LIGHT2)
    rcall delay_mS  ; delay 3.5 secs
    ; during that delay interrupts could take place.
    ; As a result the delay time will reset
    ldi temp, 0x00
    out PORTB, temp ; lights off

popLoop:
    cpi flag, 0x01  ; compare flag with zero
    breq cont       ; continue
    dec flag        ; decrease flag
    pop temp
    pop temp        ; restore temp
    rjmp popLoop
cont:
    pop temp
    out SREG, temp  ; restore SREG
                    ; enable interrupts for main
                    ; reti did that by default
    pop temp        ; restore temp
    sei
    ldi flag, 0x00  ; restore flag to 0 (first interrupt)
    rjmp main       ; jump to main and not return
```

Κώδικας σε C

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

long int flag;

ISR (INT1_vect) {
	do {
		EIFR = (1<<INTF1);
		_delay_ms(100);
	} while((EIFR & 0x02) == 0x02);

	if (flag++ != 0) {
		sei();
		PORTB = 0xFF;
		_delay_ms(500);
		PORTB = 0x01;
		_delay_ms(3500);
		PORTB = 0x00;
		flag = 0;
	}
	else {
		sei();
		PORTB = 0x01;
		_delay_ms(4000);
		PORTB = 0x00;
		flag = 0;
	}
}

int main() {
	// Interrupt on riging edge of INT1 pin
	EICRA = (1 << ISC11) | (1 << ISC10);
	// Enable the INT1 interrupt (PD3)
	EIMSK = (1 << INT1);
	sei();  // Enable global interrupts
	DDRB = 0xFF;    // Set PORTB as output
	flag = 0;
	while(1);
}
```