Συστήματα Μικροϋπολογιστών
6ο Εξάμηνο
6η Ομάδα Ασκήσεων

Χαρδούβελης Γεώργιος-Ορέστης
el15100
6ο Εξάμηνο

Κυριάκου Αθηνά
el17405
6ο Εξάμηνο

Άσκηση 1:
Το πρόγραμμα που ζητείται είναι το εξής:

```
.INCLUDE "m16def.inc"   ; δηλώνουμε μικροελεγκτή
.def temp = R16         ; ονομάζομαι temp έναν καταχωρητή
.def Delay0 = r17       ; ορίζουμε την καθυστέρηση όταν τα led θα είναι σβηστά
.def Delay1 = r18       ; ορίζουμε την καθυστέρηση όταν τα led θα είναι ανάμμένα
main:
        clr temp
        out DDRD,temp   ; ορίζουμε DDRD ως θύρα εισόδου.
        ser temp
        out DDRB,temp   ; ορίζουμε DDRB ως θύρα εξόδου.
        out PORTD,temp  ; τίθενται οι αντιστάσεις πρόσδεσης pull-up.
        ldi Delay0,50   ; αν το MSB της εισόδου είναι 0 θα έχουμε 0 (σβηστά led) για
50*(10ms)
        ldi Delay1,150  ; και θα έχουμε 1 (αναμμένα led) για 150*(50ms)
        sbic PIND, 0x07 ; αν το MSB της εισόδου είναι 1 αλλάζουμε το delay του 0, αλλιώς
όχι.
        ldi Delay0,150;
        sbic PIND, 0x07 ;Αντίστοιχα, αλλάζουμε και το delay του 1
        ldi Delay1,50;
leds_on:
        ldi temp,0xFF
        out PORTB,temp  ;ανάβουμε όλα τα leds
        //rcall Delay10 ;καλούμε την Delay10 για Delay1 φορές
        dec Delay1
        brne leds_on
leds_off:
        ldi temp,0x00
        out PORTB,temp  ;σβήνουμε όλα τα leds
        //rcall Delay10 ;καλούμε στην Delay10 για Delay0 φορές
        dec Delay0
        brne leds_off
        rjmp main   ;επιστρέφουμε πάλι στην αρχή
```

Άσκηση 2:

Το ζητούμενο πρόγραμμα σε Assembly για τον μικροελεγκτή AVR είναι:

```
.include "m16def.inc"
.DEF temp=R21
.DEF F0=R16
.DEF F1=R17
.DEF F2=R18
.cseg
.org 0

        clr temp            ;initialize all the bits of temp to 0
        out DDRD,temp       ;set PORT D as output
        ser temp            ;initialize all the bits of temp to 1
        out DDRB, temp      ;set PORT B as input
        out PORTD,temp      ;pull-up of PORT D

again:
        clr F2              ;F2 initialized to zero
        in temp, PIND       ;input in temp

        mov F0,temp
        andi F0,0x01        ;F0<-A
        BREQ ONE_F0         ;F0=1 if A=1 OR BC'D'=1
        mov F0,temp         ;if A is not 1, check if BC'D'=1
        andi F0,0x0E        ;keep BCD
        cpi F0,0x02         ;F0=1 if B=1
        breq ONE_F0

        clr F0              ;F0=0
        rjmp L_F1

ONE_F0:
        ldi F2,0x80         ;F2 turns 1
        ldi F0,0x20

L_F1:
        mov F1,temp         ;F1=1 if E=G=1 or (A=C=0 and B=D=1)
        andi F1,0x30        ;keep only the bits corresponding to E,G
        cpi F1,0x30
        breq ONE_F1

        mov F1,temp         ;if E or G != 1, check if A'BC'D=1
        andi F1,0x01        ;F1<-A
        com F1              ;F1<-A'
```

```
        mov R19,temp
        andi R19,0x02       ;R19<-B
        lsr     R19         ;B in the LSB
        and F1,R19          ;F1<-A'B

        mov R19,temp
        andi R19,0x04       ;R19<-C
        com R19             ;F1<-C'
        lsr R19
        lsr R19
        and F1,R19          ;F1<-A'BC'

        mov R19,temp
        andi R19,0x08       ;R19<-D
        lsr R19
        lsr R19
        lsr R19
        and F1,R19          ;F1<-A'BC'D

        cpi F1,0x01
        breq ONE_F1

        clr F1
        clr F2              ;F2 turns 0
        rjmp END

ONE_F1:
        ldi F1,0x40

END:
        clr temp            ;temp initialized to 0, will hold the final result
        or temp,F0
        or temp,F1
        or temp,F2
        out PORTB,temp
        rjmp again
```

Το πρόγραμμα σε γλώσσα C ενσωματωμένων συστημάτων είναι:

```
#include <avr/io.h>
unsigned char A, B, C, D, E ,G, F0, F1, F2, temp;

int main(void){

        DDRD = 0x00;                    //configure portD as input
```

```
	PORTD = 0xFF;			//Turn ON LEDs of input
	DDRB  = 0xFF;			//configure portB as output
	while(1)
	{
		A = PIND;		//Read 8 bits of port PIND (switches), store
them in register A
		B = PIND;		///Read 8 bits of port PIND again(switches),
store them in register B
		C = PIND;
		D = PIND;
		E = PIND;
		G = PIND;
		A >>= 0;		//Shift register A zero positions right, interested
bit already in the LSB position
		B >>= 1;		//Shift register B one position right, to take
interested bit in the LSB position
		C >>= 2;
		D >>= 3;
		E >>= 4;
		G >>= 5;

		//All of our caclulations will take place with the LSB element of each register
		F0 = A | (B & (~C) & (~D));
		F1 = ((~A) & B & (~C) & D) | (E & G);
		F2 = F0 & F1;

		F2 &= 1;
		F1 &= 1;
		F0 &= 1;

		//Bring each register to the correct position, unite them with "or" operation and
upload the result into LEDS
		//via PortB
		F2 <<= 7;
		F1 <<= 6;
		F0 <<= 5;
		F2 = F2 | F1 | F0;
		PORTB = F2;

	};
	//We suppose display with high impudence (positive)



}
```

Άσκηση 3:

Ο κώδικας του προγράμματος που περιγράφεται στην παρούσα άσκηση είναι ο εξής:

```
.INCLUDE "m16def.inc"   ; δήλωση του μικροελεγκτή
.DEF reg = R16
.DEF temp = R17
main:
   clr reg
   out DDRB,reg    ; ορισμός DDRB ως θύρα εισόδου
   ser reg
   out DDRD,reg    ; ορισμός DDRD ως θύρα εξόδου
   out PORTD,reg   ; τίθενται οι αντιστάσεις πρόσδεσης pull-up
   clr reg
   out PORTD,reg   ; ανάβουμε τα leds (ανάστροφη λογική)
loop:
   sbis PINB,4     ; αν δεν έχει πατηθεί ο sw4 προσπερνάνε την επομενη εντολή
   rjmp sw4        ; αν έχει πατηθεί πηγαίνουμε στο sw4 (προτεραιότητα από sw4 προς sw0)
   sbis PINB,3     ; αν δεν έχει πατηθεί ο sw3 προσπερνάνε την επομενη εντολή
   rjmp sw3        ; αν έχει πατηθεί πηγαίνουμε στο sw3
   sbis PINB,2     ; αν δεν έχει πατηθεί ο sw2 προσπερνάνε την επομενη εντολή
         rjmp sw2        ; αν έχει πατηθεί πηγαίνουμε στο sw2
         sbis PINB,1     ; αν δεν έχει πατηθεί ο sw1 προσπερνάνε την επομενη εντολή
         rjmp sw1        ; αν έχει πατηθεί πηγαίνουμε στο sw1
         sbis PINB,0     ; αν δεν έχει πατηθεί ο sw0 προσπερνάνε την επομενη εντολή και
γυρνάμε πίσω στο loop
         rjmp sw0        ; αν έχει πατηθεί πηγαίνουμε στο sw0
         rjmp loop
sw4:
         ldi reg,0xFF    ; σβήσιμο όλων των led (ανάστροφη λογική)
         out PORTD,reg
   sbic PINB,4     ; μέχρι να επανέλθει ο αντίστοιχος διακόπτης συνεχίζουμε στο sw4
         rjmp loop       ; αλλιώς επιστρέφουμε στο loop
         rjmp sw4
sw3:
   mov temp,reg    ; αποθήκευση της προηγούμενης κατάστασης για αν επιστρέψουμε σε
αυτή όταν
         ldi reg,0xFF    ; επανέλθει ο διακόπτης
   CLc             ; μηδενισμός κρατούμενου έτσι ώστε μετά το shift να ανάψει πρώτο το led
που αντιστοιχέι στο LSB
loop_sw3:
   rol reg         ; κάνουμε shift αριστερά και εμφανίζουμε
         out PORTD,reg
         //call Delay500 ; καθυστέρηση για 0.5 sec
   sbic PINB,3     ; μέχρι να επανέλθει ο αντίστοιχος διακόπτης συνεχίζουμε στο lopp_sw3
```

```
        rjmp sw3_end    ; αλλιώς πηγαίνουμε στο sw3_end
        rjmp loop_sw3
sw3_end:
   mov reg,temp
   out PORTD,temp  ; επαναφορά στην τελευταία κατάσταση των leds πριν πατηθεί ο
διακόπτης
   rjmp loop       ; και επιστροφή στην αρχή

sw2:
   mov temp,reg    ; αποθήκευση της προηγούμενης κατάστασης για αν επιστρέψουμε σε
αυτή όταν
   ldi reg,0xFE    ; επανέλθει ο διακόπτης
   SEc             ; κάνουμε το κρατούμενο 1 έτσι να λειτουργήσει σωστά το shift στη συνέχεια
loop_sw2:
        out PORTD,reg   ; εμφανίζουμε πρώτα και μετά κάνουμε shift
        //call Delay500 ; καθυστέρηση για 0.5 sec
        ror reg
        sbic PINB,2     ; μέχρι να επανέλθει ο αντίστοιχος διακόπτης συνεχίζουμε στο
lopp_sw2
        rjmp sw2_end    ; αλλιώς πηγαίνουμε στο sw2_end
        rjmp loop_sw2
sw2_end:
   mov reg,temp
   out PORTD,temp  ; επαναφορά στην τελευταία κατάσταση των leds πριν πατηθεί ο
διακόπτης
   rjmp loop       ; και επιστροφή στην αρχή

sw1:
        ldi reg,0x0F    ; ανάβουμε τα led4-7 και σβήνουμε τα led0-3 (αρνητική λογική)
        out PORTD,reg
        sbic PINB,1     ; μέχρι να επανέλθει ο αντίστοιχος διακόπτης συνεχίζουμε στο sw1
        rjmp loop       ; αλλιώς επιστρέφουμε στο loop
        rjmp sw1

sw0:
        ldi reg,0xF0    ; ανάβουμε τα led0-3 και σβήνουμε τα led4-7 (αρνητική λογική)
        out PORTD,reg
        sbic PINB,0     ; μέχρι να επανέλθει ο αντίστοιχος διακόπτης συνεχίζουμε στο sw1
        rjmp loop       ; αλλιώς επιστρέφουμε στο loop
        rjmp sw0
```

Άσκηση 4

Το πρόγραμμα σε assembly φαίνεται εδώ:

```
.include "m32def.inc"           ;Define microcontroller
```

```
.def tmp = r16
.def LEDs = r17                 ;Interrupt1 in PD3 <=> r17
.def delay1 = r19
.def delay2 = r20
.def delay3 = r21

        ;Set up the interrupt vector
        jmp reset                       ;Reset Handler
        .org INT1addr           ;INT1addr is the address of EXT_INT1
        jmp interrupt1          ;IRQ1 Handler

;note: it so happens that INT0 is hooked up to the PD0 pin for the mega 32
;Here is the mapping:
;INT0: PD0
;INT1: PD1
;(from pg2 of atmega32 datasheet)

reset:
        ;Initialize stack in the last position of the RAM
        ldi tmp,high(RAMEND)    ;Set pointer of stack to the RAM, RAMEND value is
coming of microcontroller's model
        out SPH,tmp
        ldi tmp,low(RAMEND)
        out SPL,tmp

        ;Set DDRC to 0xFF. DDRC is data direction register C. There 8 pins, so setting 8 bits
to 1 sets, sets the 8 pins for output
        ser tmp
        out DDRC,tmp                    ;digits of portC became output for the LEDs


        ldi tmp, 1<<INT1        ;1000 0000 (INT1 = 7)
        out GIMSK,temp                  ;Activate external interrupt 1, since general
register mask of interrupts is
                                                ;responsible for activating/disactivating
interrupts

        ldi temp, 0b00001100    ;Defining interrupt 1 to be executed in a rising edge, or
"ldi temp, (1 << ISC11) | (1 << ISC10)""
        out MCUCR, temp


        sei                                     ;Activate all interrupts masked, here just INT1
                                                ;Global Interrupts MUST be enabled for
the microcontroller to react to the interrupt event.
```

```
;Routine to wait for exeternal interrupt (press of button 3 - PD3)
 wait:
        rjmp wait

;This is the handler for PushButton3
interrupt1:
        ;Push conflic registers
        push tmp
        in tmp, SREG
        push tmp

;Before starting the main operation of 1 minuted turned on LEDs, they must firstly blink for 5
secs
        ldi tmp, 5                              ;(tmp) = 5

blinking:
        dec tmp
        ser LEDs
        out PORTC, LEDs
        rcall Delay
        clr LEDs
        out PORTC, LEDs
        rcall Delay
        brne blinking

main_operation:

        ;Turn on LEDs
        ser LEDs
        out PORTC, LEDs

        ldi tmp, 120                    ;(tmp) = 120


;Keep LEDs turned on for 60 sec = 1 minute, by calling DELAY routine 120 times since 1
DELAY equals to 0.5 sec time delay
;Total Delay = 120 x 1/2 sec = 60 sec
interrupt1_loop:
        dec tmp
        rcall Delay                     ;We use rcall so that when the PC gets to a "ret"
statement it will come back to the line following rcall.
        brne interrupt1_loop

        ;After 60 secs having passed, turn off the LEDS by assigning
        clr LEDs
        out PORTC, LEDs
```

```
        ;Restore conflict registers
        pop tmp
        out SREG, tmp
        pop tmp

        reti                                    ;Interrupt Return, address was loaded from
stack. Exiting the ISR, and automatically
                                                ;re-enabling the Global Interrupt Enable
bit.


Delay:
        ldi Delay1, 3
        ldi Delay2, 138
        ldi Delay3, 90
D1:
        dec Delay3
        brne D1
        dec Delay2
        brne D1
        dec Delay1
        brne D1
        ret
```

Το πρόγραμμα C φαίνεται εδώ:

```
#include <stdio.h>
#include <stdlib.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 4000000UL
#include <util/delay.h>

#define DataPort        PORTC           //Using PortC as our Dataport
#define DataDDR         DDRC
unsigned char counter,leds;

/* External Interrupt 0 service routine. When a new putton is pressed
counter initialized to 120 and the interrupts starts over again
to complete 60 secs*/

void ext_int1_isr(void){
        int i;
        counter = 120;
```

```
	/* This for loop blink LEDs on Dataport 5 times
	before LEDs turned on for 1 minute !!*/
	for(i = 0; i < 5; i++){
		DataPort = 0x00;
		_delay_ms(500);			//Turn on 0.5 sec
		DataPort = 0xFF;
		_delay_ms(500);			//Turn off 0.5 sec
	}

	leds = 0xFF;				//Leds are ON when interrup1 starts
	DataPort = leds;
	//Implement total delay = 120 x 1/2 sec = 60 sec
	while(counter!=0){
		_delay_ms(500);
	}

	leds = 0x00;				//Leds are OFF when interrupt1 finishes
	DataPort = leds;
}

int main(void){
	DDRD = 1<<PD3;				//Set PD3 as input (using for
interrupt 1) =>Configure PortD as input
	PORTD = 1<<PD3;				//Enable PD3 pull-up resisotr. Turn on
the leds of input

	DataDDR = 0xFF;				//Configure Dataport as output
	DataPort = 0x00;			//PortC as output

	GIMSK = (1<<INT1);			//Activate external interrupt1, INT1:0N
	MCUCR = 1<<ISC11 | 1<<ISC10;	//INT1 mode: activated in the a rising edge
	sei();					//Activate all interrupts masked,
here just INT1. Enable Global Interrupt


	/*Continuous operation of our program, wait for an interrupt*/
	while(1);
}
```