Εργαστήριο Μικροϋπολογιστών

7η Εργαστηριακή άσκηση

Ομάδα Γ04

Συνεργάτες:
- **Σκούλος Ραφαήλ Α.Μ: 03112404**
- **Αναστάσης Σταθόπουλος Α.Μ: 03112140**
- **Τζίνης Ευθύμιος Α.Μ: 03112007**

Άσκηση 1

```
.include "m16def.inc"

.def a0=r17
.def a1=r18
.def a2=r19
.def a3=r20
.def a4=r21
.def a5=r22
.def a6=r23
.def a7=r24
.def output=r25

        clr r26
        out DDRA,r26        ;set PORTA as input
        out DDRC,r26        ;set PORTC as input
        ser r26
        out DDRB,r26        ;set PORTB as output

start:  in r26,PINA
        clr output
        mov a0,r26                  ;get PA0-PA7
        andi a0,0x01
```

```
mov a1,r26
andi a1,0x02
lsr a1

mov a2,r26
andi a2,0x04
lsr a2
lsr a2

mov a3,r26
andi a3,0x08
lsr a3
lsr a3
lsr a3

mov a4,r26
andi a4,0x10
lsr a4
lsr a4
lsr a4
lsr a4

mov a5,r26
andi a5,0x20
cpi a5,0x20
brne next6
ldi a5,1
```

```
next6:  mov a6,r26
        andi a6,0x40
        cpi a6,0x40
        brne next7
        ldi a6,1

next7:  mov a7,r26
        andi a7,0x80
        cpi a7,0x80
        brne next
        ldi a7,1

next:   eor a0,a1               ;a0= PA0 XOR PA1
        or a2,a3                ;a2=PA2 OR PA3
        cpi a2,0        ;if (PA2 OR PA3 ) == 1 set PB1=1
        breq gate5
        ldi r26,0x02
        add output,r26

gate5:  and a2,a0
        cpi a2,0        ;if ((PA2 OR PA3) AND (PA0 XOR PA1))==1 set PB0=1
        breq gate3
        ldi r26,0x01
        add output,r26

gate3:  or a4,a5
        cpi a4,0                ;if (PA4 OR PA5)==0 then (PA4 NOR PA5)==1 so set PB2=1
        brne gate4
```

```
        ldi r26,0x04
        add output,r26
gate4:  eor a6,a7
        cpi a6,0            ;if (PA6 XOR PA7)==0 then (PA6 NXOR PA7)==1 so set PB3=1
        brne show
        ldi r26,0x08
        add output,r26

show:   in r26,PINC
        eor output,r26    ;output = output XOR r26 ,to get the compliment of LEDS PB0-PB7 if
                          ; the    equivalent PUSH BUTTON PC0-PC7 is pushed
        out PORTB,output
        jmp start
```

Άσκηση 2

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

int main() {
        char a, b, c, d, e, F0, F1, F2, Fout, input;
        DDRA = 0x00;                    //configure PORTA as input
        DDRC = 0xff;                    //configure PORTC as output
        Fout =0;
        while(1)
        {
                input = PINA;           // input from PINA
```

```
        a = input%2;                    // use (input)mod2 to aqcuire the LDB
        input = input>>1;          //shift right and repeat procedure to get all input
        b = input%2;
        input = input>>1;
        c = input%2;
        input = input>>1;
        d = input%2;
        input = input>>1;
        e = input%2;
        input = input>>1;

        F0 = ((a&b&c)|(c&d) |(d&e));
        if (F0==0) F0 = 1;              //complement fo
        else F0=0;
        if (e==0) e=1;                  //complement e
        else e=0;

        F1 = ((a&b) | (c&d&e));
        F2 = (F0 |F1);

        Fout=(128*F0) |(64*F1) |(32*F2);  //show F0 at PC7, F1 at PC6 and F2 at PC5
        PORTC = Fout;                           //output at PORTC
    }
    return 0;
}
```

Άσκηση 3

```
.dseg
_tmp_: .byte 2

.cseg
.include "m16def.inc"
.org 0x00
rjmp start

start:
   ldi r26,low(RAMEND)                    ;initialize stack pointer
   out SPL,r26
   ldi r26,high(RAMEND)
   out SPH,r26
   ser r24
   out DDRB,r24                  ; PORTB output
   ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
   out DDRC ,r24                 ; 4x4 keypad output initiallization
   ldi r24,0x00
   ldi r25,0x00
   ldi r26 ,low(_tmp_)           ;r26:r27 makes thw register X
   ldi r27 ,high(_tmp_)
   st X+ ,r24                    ;store in the address of X zero
   st X ,r25
```

```
locked:
   ldi r24, 0x14                ;r24 has the bouncing time lets say 20ms
   rcall scan_keypad_rising_edge        ;read the first num from keyb
   rcall keypad_to_ascii        ;turn it to ascii and get the result in r24
   cpi r24, '0'                 ;r24 has the result, if first number == 0
   brne locked                  ;else check again

loop2:
   ldi r24, 0x14
   rcall scan_keypad_rising_edge
   rcall keypad_to_ascii
    cpi r24,'0'
    breq loop2
    cpi r24,0
   breq loop2
   cpi r24, '4'                 ;if the second number is 4 then unlock
   brne locked
   rjmp open

open:
   ldi r24,0xff                 ;open leds of b for 3 secs
   out PORTB, r24
   ldi r24,low(3000)
   ldi r25,high(3000)
   rcall wait_msec
   ldi r24,0x00
   out PORTB, r24                       ;turn leds off
```

```
   rjmp locked                          ;continuous functionality


.org 0x300
rjmp start


scan_row:
        ;nop
        ldi r25 ,0x08
back_:
        lsl r25
        dec r24
        brne back_
        out PORTC ,r25
        nop
        nop
        in r24 ,PINC
        andi r24 ,0x0f
        ret
scan_keypad:
        ;nop
        ldi r24 ,0x01
        rcall scan_row
        swap r24
        mov r27 ,r24
        ldi r24 ,0x02
        rcall scan_row
        add r27 ,r24
```

```
        ldi r24 ,0x03
        rcall scan_row
        swap r24
        mov r26 ,r24
        ldi r24 ,0x04
        rcall scan_row
        add r26 ,r24
        movw r24 ,r26
        ret
scan_keypad_rising_edge:
        ;nop
        mov r22 ,r24
        rcall scan_keypad
        push r24
        push r25
        mov r24 ,r22
        ldi r25 ,0
        rcall wait_msec
        rcall scan_keypad
        pop r23
        pop r22
        and r24 ,r22
        and r25 ,r23
        ldi r26 ,low(_tmp_)
        ldi r27 ,high(_tmp_)
        ld r23 ,X+
        ld r22 ,X
```

```
        st X ,r24
        st -X ,r25
        com r23
        com r22
        and r24 ,r22
        and r25 ,r23
        ret



;delay routines
wait_usec:
        sbiw r24 ,1                     ;2 circle (0.250 micro sec)
        nop                             ;1 cilrcle (0.125
micro sec)
        nop
        nop
        nop
        brne wait_usec                  ;1 or 2 circles (0.125 or 0.250 micro
sec)
        ret                             ;4 circles (0.500 mic
sec)

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

keypad_to_ascii:        ;routine to tranduce the result in r24 to an ascii code in
        movw r26 ,r24   ;order to compare it afterwards
        ldi r24 ,'*'
        sbrc r26 ,0
        ret
        ldi r24 ,'0'
        sbrc r26 ,1
        ret
        ldi r24 ,'#'
        sbrc r26 ,2
        ret
        ldi r24 ,'D'
        sbrc r26 ,3
        ret
        ldi r24 ,'7'
        sbrc r26 ,4
        ret
        ldi r24 ,'8'
        sbrc r26 ,5
        ret
        ldi r24 ,'9'
        sbrc r26 ,6
```

```
ret
ldi r24 ,'C'
sbrc r26 ,7
ret
ldi r24 ,'4'
sbrc r27 ,0
ret
ldi r24 ,'5'
sbrc r27 ,1
ret
ldi r24 ,'6'
sbrc r27 ,2
ret
ldi r24 ,'B'
sbrc r27 ,3
ret
ldi r24 ,'1'
sbrc r27 ,4
ret
ldi r24 ,'2'
sbrc r27 ,5
ret
ldi r24 ,'3'
sbrc r27 ,6
ret
ldi r24 ,'A'
sbrc r27 ,7
```

```
ret

clr r24

ret
```