ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΜΠ

‘’ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ”

Μπακούρος Αριστείδης 03113138
Ορφανουδάκης Φίλιππος 03113140
Ρουσσάκης Γεώργιος 03113041

6η Σειρά Ασκήσεων

Ζήτημα 6.1

Εδώ ζητήθηκε να προσομοιώσουμε τη λειτουργία κάποιων λογικών πυλών και την εμφάνιση των εξόδων τους στη θύρα B. Επίσης όταν έχουμε πάτημα κάποιου από τη θύρα C πρέπει να αντιστρέφεται η αντίστοιχη ένδειξη της θύρας B.

```
.include "m16def.inc"
.def temp=r16
.def inp=r17
.def inp0=r18
.def inp1=r19
.def inp2=r20
.def inp3=r21
.def inp4=r22
.def inp5=r23
.def inp6=r26
.def inp7=r27
.def inpC=r29
.def outp=r30
.cseg
.org 0x0

;--------.INITIALIZE STACK.------;
ldi temp,HIGH(RAMEND)              ;;αρχικοποίηση στοίβας
out SPH,temp
ldi temp,LOW(RAMEND)
out SPL,temp

;-------.DEFINE INPUT AND OUTPUT.------;

ser temp
out DDRB,temp                      ;;αρχικοποιούμε τη θύρα B
ως έξοδο
ldi temp,0
out DDRA,temp                      ;;αρχικοποιούμε τη θύρα A
ως είσοδο
out DDRC,temp                      ;;αρχικοποιούμε τη θύρα C
ως είσοδο

;------.TAKE INPUT AND SAVE EACH BIT.----;

start:
clr inp                                ;inputA
```

```
clr inpC                    ;inputC
clr outp                    ;output
clr inp0                    ;BIT0
clr inp1                    ;BIT1
clr inp2                    ;BIT2
clr inp3                    ;BIT3
clr inp4                    ;BIT4
clr inp5                    ;BIT5
clr inp6                    ;BIT6
clr inp7                    ;BIT7
in inp,PINA                 ;ανάγνωση της θύρας A
in inpC,PINC                ;ανάγνωση της θύρας C
mov inp0,inp                ;αποθήκευση
mov inp1,inp                ;αποθήκευση
mov inp2,inp                ;αποθήκευση
mov inp3,inp                ;αποθήκευση
mov inp4,inp                ;αποθήκευση
mov inp5,inp                ;αποθήκευση
mov inp6,inp                ;αποθήκευση
mov inp7,inp                ;αποθήκευση

;Βάζουμε τα μπιτ της εισόδου στους αντίστοιχους
καταχωρητές κάνοντας λογικά ΚΑΙ και τις
κατάλληλες μετατοπίσεις

ldi r28,1
and inp0,r28

ldi r28,2
and inp1,r28
lsr inp1

ldi r28,4
and inp2,r28
lsr inp2
lsr inp2

ldi r28,8
and inp3,r28
lsr inp3
lsr inp3
lsr inp3

ldi r28,16
and inp4,r28
lsr inp4
```

```
lsr inp4
lsr inp4
lsr inp4

ldi r28,32
and inp5,r28
lsl inp5
lsl inp5
lsl inp5

ldi r28,64
and inp6,r28
lsl inp6
lsl inp6

ldi r28,128
and inp7,r28
lsl inp7

;-----.ΠΥΛΗ 1 - XOR PA0-PA1.---------;

eor inp0,inp1                          ;αποθήκευση στο inp0

;-----.ΠΥΛΗ 2- OR PA2-PA3.-----------;

or inp2,inp3                           ;αποθήκευση στο inp2

;-----.ΠΥΛΗ 3 - NOR PA4-PA5.---------;

or inp4,inp5
com inp4
ldi r28,1
and inp4,r28                           ;αποθήκευση στο inp4

;----.PYLH 4 - NXOR PA6-PA7.----------;

eor inp6,inp7
com inp6
ldi r28,1
and inp6,r28                           ;αποθήκευση στο inp6

;----.PYLH 5 - AND RESULT1-RESULT2.-------;

and inp0,inp2                          ;αποθήκευση στο inp0
```

```
;------.FORMULATE OUTPUT.---------; κατάλληλες λογικές
πράξεις και μετατοπίσεις μπιτ για να
δημιουργήσουμε το αποτέλεσμα στο outp

or outp,inp0
lsl inp2
or outp,inp2
lsl inp4
lsl inp4
or outp,inp4
lsl inp6
lsl inp6
lsl inp6
or outp,inp6


;------.IMPACT OF PORTC.---------;

eor outp,inpC                          ;;xor για να γίνει η
ανιστροφή

;--------.PRINT.--------;

out PORTB,outp


rjmp start
```

Ζήτημα 6.2

Σε αυτό το ερώτημα χρειάστηκε να υλοποιήσουμε 2 συναρτήσεις, τις εισόδους των οποίων παίρνουμε από τη θύρα A, και το άθροισμα και να το προβάλουμε στα MSB (5-7) της θύρας C.

```
#include <avr/io.h>
#include <stdbool.h>
int main(void)
{
      DDRC=0xFF;                    // θέτουμε τη θύρα C ως
έξοδο
      DDRA=0x00;                    // θέτουμε τη θύρα A ως
είσοδο
      int f0,f1,f2,a,b,c,d,e,inp ;

      while(1)
      {
            inp=PINA;               // ανάγνωση εισόδου
```

```
            a=inp&0b00000001 ;      // PA0
            b=inp&0b00000010 ;      // PA1
            c=inp&0b00000100 ;      // PA2
            d=inp&0b00001000 ;      // PA3
            e=inp&0b00010000 ;      // PA4

            b=b>>1 ;                // μια μετατόπιση δεξιά
            c=c>>2 ;                // δύο μετατοπίσεις δεξία
            d=d>>3 ;                // τρεις μετατοπίσεις
δεξιά
            e=e>>4 ;                // τέσσερις μετατοπίσεις
δεξιά

            f0= !((a&b&c) | (c&d) | (d&e)) ;    // f0=(ABC + CD + DE)'
            f1= (a&b&c) | (!(d|e)) ;            // f1=(ABC +D'E')
            f2= f0 | f1 ;                       // f2=(f0+f1)

            f0=f0<<5 ;                  //πέντε μετατοπίσεις
αριστερά
            f1=f1<<6 ;                  //έξι μετατοπίσεις
αριστερά
            f2=f2<<7 ;                  //επτά μετατοπίσεις
αριστερά

            PORTC= f0 | f1 | f2 ;       //έξοδος


      }
}
```

Ζήτημα 6.3

- Μας ζητήθηκε σε αυτή την άσκηση να υλοποιήσουμε τη λειτουργία μιας ηλεκτρονικής κλειδαριάς , της οποίας ο διψήφιος κωδικός θα είναι ο αριθμός της ομάδας μας δηλαδή “0” , “6” .

- Πιο συγκεριμένα δεχόμαστε από το 4x4 πληκτρολόγιο το input μας και αν είναι σωστό τότε “ξεκλειδώνεται” το σύστημα και έχουμε αναμμένα τα led PB0-PB7 για 4 sec

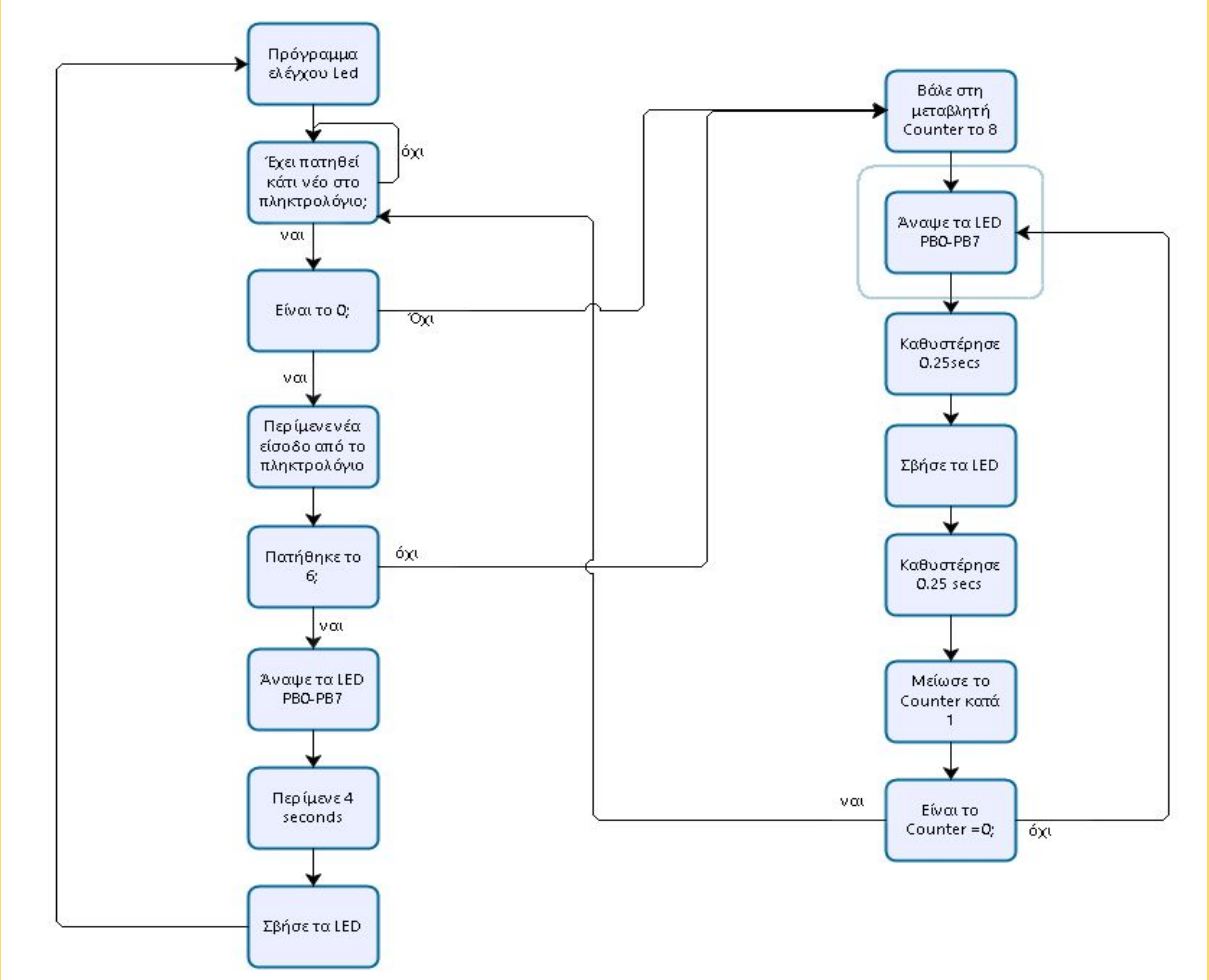
- Αν είναι λάθος τότε αναβοσβήνουν τα led PB0-PB7 για 4 sec

- Ο κώδικας μας χρησιμοποιεί τη ρουτίνα scan_keypad_rising_edge για να διαβάσει από το πληκτρολόγιο και να αποφύγει το φαινόμενο να θεωρηθεί οτι πατήθηκε ένα πλήκτρο πολλές φορές ενώ στη πραγματικότητα ήταν συνεχόμενα πατημένο αρκετή ώρα .

- Στη συνέχεια μετατρέπουμε αυτο το input στον ascii κωδικό του με την ρουτίνα keypad_to_ascii , και βλέπουμε αν είναι το επιθυμητό ή όχι για να περιμένουμε σε μια δεύτερη παρόμοια ρουτίνα το δεύτερο ψηφίο.

- Υπάρχει μια ιδιαιτερότητα στη κλειδαριά κατα την οποία αν το πρώτο πλήκτρο είναι λάθος αμέσως κλειδώνει. Θεωρήσαμε ότι υπάρχει μια σχετική ελευθερία στο συγκεκριμένο κομμάτι . Αν θέλαμε να το αλλάξουμε πολύ απλά μπορούμε να βάλουμε μια ακόμα ρουτίνα που περιμένει είσοδο και με το που την πάρει να πηγαίνει στο κλείδωμα

Το διάγραμμα ροής :



και ο κώδικας :

```
.include "m16def.inc"
.def temp=r16
.def counter=r17
.def leds=r18
.DSEG
_tmp_: .byte 2

.CSEG
.org 0x0

;-------.INITIALIZE STACK.------;

ldi temp,HIGH(RAMEND)
out SPH,temp
ldi temp,LOW(RAMEND)
out SPL,temp

;--------.SET INPUT AND OUTPUT.-------;

ser temp
out DDRB,temp    ;PORTB is output


ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4) ;enable input
;from keyboard
out DDRC ,r24
clr r24
sts _tmp_,r24

;----------.READ FROM KEYBOARD.-------------;
input:
ldi r24, 20  ; delay time for rising edge
call scan_keypad_rising_edge ;read input and avoid sparkling
call keypad_to_ascii ;turn our input into ascii
cpi r24, 0 ; if r24=0 nothing was pressed
breq input
cpi r24, '0' ;first input must be 0
brne wrong   ; if not start blinking

second:
ldi r24, 20
call scan_keypad_rising_edge
call keypad_to_ascii
```

```
cpi r24,0
breq second
cpi r24,'6' ;second input must be 6
brne wrong    ;if not start blinking



;--------.ROUTINE FOR CORRECT COMBINATION.------;

correct:
    ser leds
    out PORTB,leds ;LIGHT IT UUUUPP
    ldi r24 ,low(4000)
    ldi r25 ,high(4000)
    rcall wait_msec
    clr leds
    out PORTB,leds ;;SHUT IT DOWN
    rjmp input



;--------.ROUTINE FOR WRONG COMBINATION.--------;

wrong:

ldi counter,8 ; 8 * (0,25+0,25) = 4sec
con:
ser leds
out PORTB,leds
ldi r24 ,low(250)
ldi r25 ,high(250)
rcall wait_msec
clr leds
out PORTB,leds
ldi r24 ,low(250)
ldi r25 ,high(250)
rcall wait_msec
dec counter
cpi counter,0
brne con
rjmp input


scan_row:
    ldi r25 , 0x08       ; ??????????? ?? 0000 1000
back_:
```

```
    lsl r25                  ; ???????? ???????? ??? 1 ????? ??????
    dec r24                  ; ???? ????? ? ??????? ??? ???????
    brne back_
    out PORTC , r25      ; ? ????????? ?????? ??????? ??? ?????? 1
    nop
    nop                      ; ?????????? ??? ?? ???????? ?? ????? ?
?????? ??????????
    in r24 , PINC        ; ?????????? ?? ?????? (??????) ??? ????????
??? ????? ????????
    andi r24 ,0x0f       ; ???????????? ?? 4 LSB ???? ?? 1 ???????? ???
????? ????????
    ret                      ; ?? ????????.

scan_keypad:
    ldi r24 , 0x01       ; ?????? ??? ????? ?????? ??? ????????????
    rcall scan_row
    swap r24             ; ?????????? ?? ??????????
    mov r27 , r24        ; ??? 4 msb ??? r27
    ldi r24 ,0x02        ; ?????? ?? ??????? ?????? ??? ????????????
    rcall scan_row
    add r27 , r24        ; ?????????? ?? ?????????? ??? 4 lsb ??? r27
    ldi r24 , 0x03       ; ?????? ??? ????? ?????? ??? ????????????
    rcall scan_row
    swap r24             ; ?????????? ?? ??????????
    mov r26 , r24        ; ??? 4 msb ??? r26
    ldi r24 ,0x04        ; ?????? ??? ??????? ?????? ??? ????????????
    rcall scan_row
    add r26 , r24        ; ?????????? ?? ?????????? ??? 4 lsb ??? r26
    movw r24 , r26       ; ???????? ?? ?????????? ????? ??????????
r25:r24
    ret

scan_keypad_rising_edge:
    mov r22 ,r24         ; ?????????? ?? ????? ??????????? ???? r22
    rcall scan_keypad    ; ?????? ?? ???????????? ??? ????????? ????????
    push r24             ; ??? ????????? ?? ??????????
    push r25
    mov r24 ,r22         ; ?????????? r22 ms (??????? ????? 10-20 msec
??? ?????????? ??? ???
    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_) ; ?????????? ????? ??? ???????? ????? r27:r26
    ld r23 ,X+
    ld r22 ,X
    st X ,r24            ; ?????????? ??? RAM ?? ??? ?????????
    st -X ,r25           ; ??? ?????????
    com r23
    com r22                   ; ???? ???? ????????? ??? ????? «?????»
???????
    and r24    ,r22
    and r25 ,r23
    ret

keypad_to_ascii:
     movw r26 ,r24
    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



;--------.WAIT ROUTINES.--------;

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
```