Εργαστήριο Μικρουπολογιστών

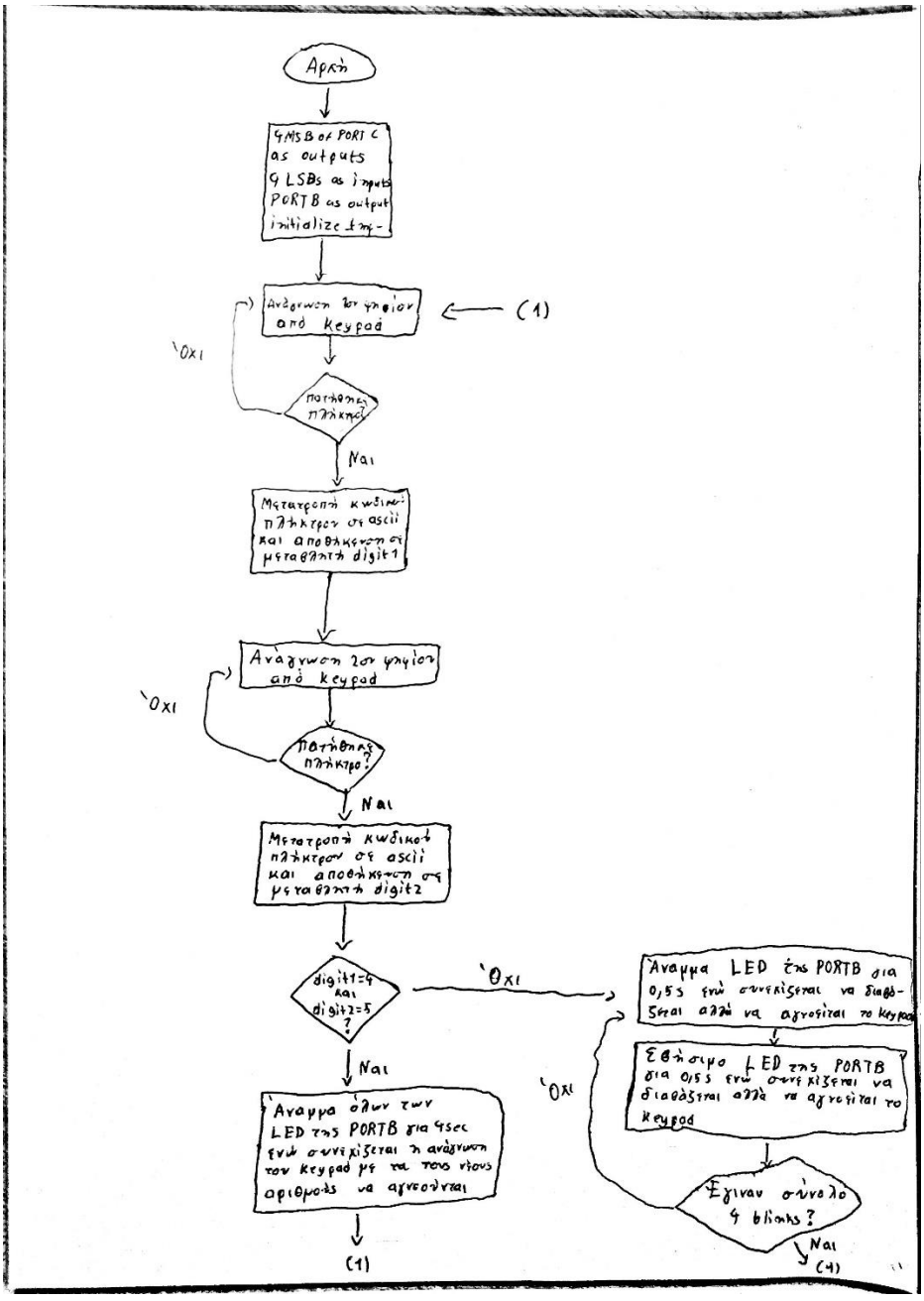
3η Άσκηση (AVR2)

Μοίρας Αλέξανδρος

A.M.: el18081

Άσκηση 1:

Το διάγραμμα ροής:



Χρησιμοποιήθηκαν 2 αρχεία -main.c και Assemblyfunctions.S- με το main.c να περιέχει το κυρίως πρόγραμμα και το Assemblyfunctions.S να περιέχει τους οδηγούς του keypad, τις συναρτήσεις χρονοκαθυστερήσεων καθώς και μία συνάρτηση που αρχικοποιεί τη μεταβλητή _tmp_ που βρίσκεται στη RAM. Το compilation το αναλαμβάνει το microchip studio το οποίο κατά το build κάνει αυτόματα link τα παράγωγα του κώδικα που περιέχεται στα δύο αρχεία. Κατά τη σχεδίαση σεβαστήκαμε τα function call conventions μεταξύ c και assembly (πχ το 16bit return value της scan_keypad_rising_edge_sim() αποθηκεύεται στους r25:r24 ώστε να ανακτηθεί από το c πρόγραμμα κατά την επιστροφή της συνάρτησης). Ο κώδικας των δύο αρχείων εμπλουτισμένος με σχόλια:

main.c:

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

int scan_keypad_rising_edge_sim(); //declaration of assembly functions. 16bit
return value must be stored in r25:r24
//void wait_msec(int msecs);
//void wait_usec(int usecs);
void initialize_variable();

char keypad_to_ascii(int btn) { //returns the ascii character that corresponds to
the first bit 1 found
    if ((btn & 0x0001)==0x0001) {
        return '*';
    }
    if ((btn & 0x0002)==0x0002) {
        return '0';
    }
    if ((btn & 0x0004)==0x0004) {
        return '#';
    }
    if ((btn & 0x0008)==0x0008) {
        return 'D';
    }
    if ((btn & 0x0010)==0x0010) {
        return '7';
    }
    if ((btn & 0x0020)==0x0020) {
        return '8';
    }
    if ((btn & 0x0040)==0x0040) {
```

```
        return '9';
    }
    if ((btn & 0x0080)==0x0080) {
        return 'C';
    }
    if ((btn & 0x0100)==0x0100) {
        return '4';
    }
    if ((btn & 0x0200)==0x0200) {
        return '5';
    }
    if ((btn & 0x0400)==0x0400) {
        return '6';
    }
    if ((btn & 0x0800)==0x0800) {
        return 'B';
    }
    if ((btn & 0x1000)==0x1000) {
        return '1';
    }
    if ((btn & 0x2000)==0x2000) {
        return '2';
    }
    if ((btn & 0x4000)==0x4000) {
        return '3';
    }
    if ((btn & 0x8000)==0x8000) {
        return 'A';
    }
    return 0;
}

int main(void)
{
    DDRC = 0xF0;    //4 MSBs of PORTC as outputs 4LSBs as inputs
    PORTC = 0x00;   //disable pull-up resistors
    DDRB = 0xFF;    //PORTB as output

    int btn;
    int digit1, digit2;
    initialize_variable(); //call to the assembly function that initializes _tmp_
    while (1)
    {
        btn=0;
        while(btn==0){
```

```
            btn=scan_keypad_rising_edge_sim();  //scan the keypad until a key is
pressed
        }
        digit1=keypad_to_ascii(btn)-48;         //translate the first key pressed
to an ascii character then to an integer
        btn=0;
        while(btn==0){
            btn=scan_keypad_rising_edge_sim();  //wait until the second key is
pressed
        }
        digit2=keypad_to_ascii(btn)-48;         //translate the second key
pressed to an ascii character then to an integer

        if((digit1==4)&&(digit2==5)) {  //if password is correct
            PORTB=0xFF;                 //light all PORTB leds
            for(int i=0; i<190; ++i) {          //we need to keep scanning the
keypad. Each scan takes longer than 19ms (19ms is the total delay time from delay
routines called by scan_keypad_rising_edge_sim())
                scan_keypad_rising_edge_sim();  //so we call it 190 times so that
the leds stay on for ~4secs
            }
            PORTB=0x00; //PORTB leds off
        }
        else {                              //if password is incorrect
            for(int i=0; i<4; ++i) {        //total of four blinks
                PORTB=0xFF;                 //light PORTB leds
                for(int i=0; i<24; ++i) {   //we need to keep scanning the
keypad. Each scan takes longer than 19ms (19ms is the total delay time from delay
routines called by scan_keypad_rising_edge_sim())
                    scan_keypad_rising_edge_sim(); //so we call it 24 times for a
total of ~0.5secs
                }
                PORTB=0x00;                 //then turn them off for another
0.5secs
                for(int i=0; i<24; ++i) {
                    scan_keypad_rising_edge_sim();
                }
            }
        } //read two numbers from the keypad again
    }
}
```

Assemblyfunctions.S:

```
#include <avr/io.h>
#define _SFR_ASM_COMPAT 1
```

```
#define __SFR_OFFSET 0 //required in order to use the I/O ports from this file
    .DATA
    _tmp_: .byte 2

    .TEXT
    .global scan_keypad_rising_edge_sim
    scan_keypad_rising_edge_sim:
    push r22 ; αποθήκευσε τους καταχωρητές r23:r22 και τους
    push r23 ; r26:r27 γιατι τους αλλάζουμε μέσα στην ρουτίνα
    push r26
    push r27
    rcall scan_keypad_sim ; έλεγξε το πληκτρολόγιο για πιεσμένους διακόπτες
    push r24 ; και αποθήκευσε το αποτέλεσμα
    push r25
    ldi r24 ,15 ; καθυστέρησε 15 ms (τυπικές τιμές 10-20 msec που καθορίζεται από
τον
    ldi r25 ,0 ; κατασκευαστή του πληκτρολογίου – χρονοδιάρκεια σπινθηρισμών)
    rcall wait_msec
    rcall scan_keypad_sim ; έλεγξε το πληκτρολόγιο ξανά και απόρριψε
    pop r23 ; όσα πλήκτρα εμφανίζουν σπινθηρισμό
    pop r22
    and r24 ,r22
    and r25 ,r23
    ldi r26 ,lo8(_tmp_) ; φόρτωσε την κατάσταση των διακοπτών στην
    ldi r27 ,hi8(_tmp_) ; προηγούμενη κλήση της ρουτίνας στους r27:r26
    ld r23 ,X+
    ld r22 ,X
    st X ,r24 ; αποθήκευσε στη RAM τη νέα κατάσταση
    st -X ,r25 ; των διακοπτών
    com r23
    com r22 ; βρες τους διακόπτες που έχουν «μόλις» πατηθεί
    and r24 ,r22
    and r25 ,r23
    pop r27 ; επανάφερε τους καταχωρητές r27:r26
    pop r26 ; και r23:r22
    pop r23
    pop r22
    ret

    .global initialize_variable
    initialize_variable:
    ldi r24, 0xFF
    sts _tmp_, r24 ; initialize _tmp_ to 0xFF
    ret
```

```
    wait_msec:
    push r24 ; 2 κύκλοι (0.250 μsec)
    push r25 ; 2 κύκλοι
    ldi r24 , lo8(998) ; φόρτωσε τον καταχ. r25:r24 με 998 (1 κύκλος - 0.125
μsec)
    ldi r25 , hi8(998) ; 1 κύκλος (0.125 μsec)
    rcall wait_usec ; 3 κύκλοι (0.375 μsec), προκαλεί συνολικά καθυστέρηση
998.375 μsec
    pop r25 ; 2 κύκλοι (0.250 μsec)
    pop r24 ; 2 κύκλοι
    sbiw r24 , 1 ; 2 κύκλοι
    brne wait_msec ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
    ret ; 4 κύκλοι (0.500 μsec)

    wait_usec:
    sbiw r24 ,1 ; 2 κύκλοι (0.250 μsec)
    nop ; 1 κύκλος (0.125 μsec)
    nop ; 1 κύκλος (0.125 μsec)
    nop ; 1 κύκλος (0.125 μsec)
    nop ; 1 κύκλος (0.125 μsec)
    brne wait_usec ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
    ret ; 4 κύκλοι (0.500 μsec)

    scan_row_sim:
    out PORTC, r25 ; η αντίστοιχη γραμμή τίθεται στο λογικό ‘1’
    push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
    push r25 ; λειτουργία του προγραμματος απομακρυσμένης
    ldi r24,0xF4 ; πρόσβασης
    ldi r25,0x01
    rcall wait_usec
    pop r25
    pop r24 ; τέλος τμήμα κώδικα
    nop
    nop ; καθυστέρηση για να προλάβει να γίνει η αλλαγή κατάστασης
    in r24, PINC ; επιστρέφουν οι θέσεις (στήλες) των διακοπτών που είναι
πιεσμένοι
    andi r24 ,0x0f ; απομονώνονται τα 4 LSB όπου τα ‘1’ δείχνουν που είναι
πατημένοι
    ret ; οι διακόπτες.

    scan_keypad_sim:
    push r26 ; αποθήκευσε τους καταχωρητές r27:r26 γιατι τους
    push r27 ; αλλάζουμε μέσα στην ρουτίνα
    ldi r25 , 0x10 ; έλεγξε την πρώτη γραμμή του πληκτρολογίου (PC4: 1 2 3 A)
    rcall scan_row_sim
```

```
    swap r24 ; αποθήκευσε το αποτέλεσμα
    mov r27, r24 ; στα 4 msb του r27
    ldi r25 ,0x20 ; έλεγξε τη δεύτερη γραμμή του πληκτρολογίου (PC5: 4 5 6 B)
    rcall scan_row_sim
    add r27, r24 ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r27
    ldi r25 , 0x40 ; έλεγξε την τρίτη γραμμή του πληκτρολογίου (PC6: 7 8 9 C)
    rcall scan_row_sim
    swap r24 ; αποθήκευσε το αποτέλεσμα
    mov r26, r24 ; στα 4 msb του r26
    ldi r25 ,0x80 ; έλεγξε την τέταρτη γραμμή του πληκτρολογίου (PC7: * 0 # D)
    rcall scan_row_sim
    add r26, r24 ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r26
    movw r24, r26 ; μετέφερε το αποτέλεσμα στους καταχωρητές r25:r24
    clr r26 ; προστέθηκε για την απομακρυσμένη πρόσβαση
    out PORTC,r26 ; προστέθηκε για την απομακρυσμένη πρόσβαση
    pop r27 ; επανάφερε τους καταχωρητές r27:r26
    pop r26
    ret
```

Άσκηση 2:

Το αρχείο main.asm το οποίο περιέχει τόσο τους οδηγούς των συσκευών όσο και το πρόγραμμά μας εμπλουτισμένο με σχόλια:

```
.DSEG
_tmp_: .byte 2

.CSEG

rjmp main ; go to the start of the program

wait_msec:
push r24 ; 2 κύκλοι (0.250 μsec)
push r25 ; 2 κύκλοι
ldi r24 , low(998) ; φόρτωσε τον καταχ. r25:r24 με 998 (1 κύκλος - 0.125 μsec)
ldi r25 , high(998) ; 1 κύκλος (0.125 μsec)
rcall wait_usec ; 3 κύκλοι (0.375 μsec), προκαλεί συνολικά καθυστέρηση 998.375
μsec
pop r25 ; 2 κύκλοι (0.250 μsec)
pop r24 ; 2 κύκλοι
sbiw r24 , 1 ; 2 κύκλοι
brne wait_msec ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
ret ; 4 κύκλοι (0.500 μsec)
```

```
wait_usec:
sbiw r24 ,1 ; 2 κύκλοι (0.250 μsec)
nop ; 1 κύκλος (0.125 μsec)
nop ; 1 κύκλος (0.125 μsec)
nop ; 1 κύκλος (0.125 μsec)
nop ; 1 κύκλος (0.125 μsec)
brne wait_usec ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
ret ; 4 κύκλοι (0.500 μsec)

scan_row_sim:
out PORTC, r25 ; η αντίστοιχη γραμμή τίθεται στο λογικό ‘1’
push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
push r25 ; λειτουργία του προγραμματος απομακρυσμένης
ldi r24,low(500) ; πρόσβασης
ldi r25,high(500)
rcall wait_usec
pop r25
pop r24 ; τέλος τμήμα κώδικα
nop
nop ; καθυστέρηση για να προλάβει να γίνει η αλλαγή κατάστασης
in r24, PINC ; επιστρέφουν οι θέσεις (στήλες) των διακοπτών που είναι πιεσμένοι
andi r24 ,0x0f ; απομονώνονται τα 4 LSB όπου τα ‘1’ δείχνουν που είναι πατημένοι
ret ; οι διακόπτες.

scan_keypad_sim:
push r26 ; αποθήκευσε τους καταχωρητές r27:r26 γιατι τους
push r27 ; αλλάζουμε μέσα στην ρουτίνα
ldi r25 , 0x10 ; έλεγξε την πρώτη γραμμή του πληκτρολογίου (PC4: 1 2 3 A)
rcall scan_row_sim
swap r24 ; αποθήκευσε το αποτέλεσμα
mov r27, r24 ; στα 4 msb του r27
ldi r25 ,0x20 ; έλεγξε τη δεύτερη γραμμή του πληκτρολογίου (PC5: 4 5 6 B)
rcall scan_row_sim
add r27, r24 ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r27
ldi r25 , 0x40 ; έλεγξε την τρίτη γραμμή του πληκτρολογίου (PC6: 7 8 9 C)
rcall scan_row_sim
swap r24 ; αποθήκευσε το αποτέλεσμα
mov r26, r24 ; στα 4 msb του r26
ldi r25 ,0x80 ; έλεγξε την τέταρτη γραμμή του πληκτρολογίου (PC7: * 0 # D)
rcall scan_row_sim
add r26, r24 ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r26
movw r24, r26 ; μετέφερε το αποτέλεσμα στους καταχωρητές r25:r24
clr r26 ; προστέθηκε για την απομακρυσμένη πρόσβαση
out PORTC,r26 ; προστέθηκε για την απομακρυσμένη πρόσβαση
pop r27 ; επανάφερε τους καταχωρητές r27:r26
```

```
pop r26
ret

scan_keypad_rising_edge_sim:
push r22 ; αποθήκευσε τους καταχωρητές r23:r22 και τους
push r23 ; r26:r27 γιατι τους αλλάζουμε μέσα στην ρουτίνα
push r26
push r27
rcall scan_keypad_sim ; έλεγξε το πληκτρολόγιο για πιεσμένους διακόπτες
push r24 ; και αποθήκευσε το αποτέλεσμα
push r25
ldi r24 ,15 ; καθυστέρησε 15 ms (τυπικές τιμές 10-20 msec που καθορίζεται από τον
ldi r25 ,0 ; κατασκευαστή του πληκτρολογίου – χρονοδιάρκεια σπινθηρισμών)
rcall wait_msec
rcall scan_keypad_sim ; έλεγξε το πληκτρολόγιο ξανά και απόρριψε
pop r23 ; όσα πλήκτρα εμφανίζουν σπινθηρισμό
pop r22
and r24 ,r22
and r25 ,r23
ldi r26 ,low(_tmp_) ; φόρτωσε την κατάσταση των διακοπτών στην
ldi r27 ,high(_tmp_) ; προηγούμενη κλήση της ρουτίνας στους r27:r26
ld r23 ,X+
ld r22 ,X
st X ,r24 ; αποθήκευσε στη RAM τη νέα κατάσταση
st -X ,r25 ; των διακοπτών
com r23
com r22 ; βρες τους διακόπτες που έχουν «μόλις» πατηθεί
and r24 ,r22
and r25 ,r23
pop r27 ; επανάφερε τους καταχωρητές r27:r26
pop r26 ; και r23:r22
pop r23
pop r22
ret

keypad_to_ascii_sim:
push r26 ; αποθήκευσε τους καταχωρητές r27:r26 γιατι τους
push r27 ; αλλάζουμε μέσα στη ρουτίνα
movw r26 ,r24   ; λογικό ‘1’ στις θέσεις του καταχωρητή r26 δηλώνουν
ldi r24 ,'*'    ; τα παρακάτω σύμβολα και αριθμούς
sbrc r26 ,0
rjmp return_ascii
ldi r24 ,'0'
sbrc r26 ,1
rjmp return_ascii
```

```
ldi r24 ,'#'
sbrc r26 ,2
rjmp return_ascii
ldi r24 ,'D'
sbrc r26 ,3 ; αν δεν είναι ‘1’παρακάμπτει την ret, αλλιώς (αν είναι ‘1’)
rjmp return_ascii ; επιστρέφει με τον καταχωρητή r24 την ASCII τιμή του D.
ldi r24 ,'7'
sbrc r26 ,4
rjmp return_ascii
ldi r24 ,'8'
sbrc r26 ,5
rjmp return_ascii
ldi r24 ,'9'
sbrc r26 ,6
rjmp return_ascii ;
ldi r24 ,'C'
sbrc r26 ,7
rjmp return_ascii
ldi r24 ,'4' ; λογικό ‘1’ στις θέσεις του καταχωρητή r27 δηλώνουν
sbrc r27 ,0 ; τα παρακάτω σύμβολα και αριθμούς
rjmp return_ascii
ldi r24 ,'5'
sbrc r27 ,1
rjmp return_ascii
ldi r24 ,'6'
sbrc r27 ,2
rjmp return_ascii
ldi r24 ,'B'
sbrc r27 ,3
rjmp return_ascii
ldi r24 ,'1'
sbrc r27 ,4
rjmp return_ascii ;
ldi r24 ,'2'
sbrc r27 ,5
rjmp return_ascii
ldi r24 ,'3'
sbrc r27 ,6
rjmp return_ascii
ldi r24 ,'A'
sbrc r27 ,7
rjmp return_ascii
clr r24
rjmp return_ascii
return_ascii:
```

```
pop r27 ; επανάφερε τους καταχωρητές r27:r26
pop r26
ret

write_2_nibbles_sim:
push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
push r25 ; λειτουργία του προγραμματος απομακρυσμένης
ldi r24 ,low(6000) ; πρόσβασης
ldi r25 ,high(6000)
rcall wait_usec
pop r25
pop r24 ; τέλος τμήμα κώδικα
push r24 ; στέλνει τα 4 MSB
in r25, PIND ; διαβάζονται τα 4 LSB και τα ξαναστέλνουμε
andi r25, 0x0f ; για να μην χαλάσουμε την όποια προηγούμενη κατάσταση
andi r24, 0xf0 ; απομονώνονται τα 4 MSB και
add r24, r25 ; συνδυάζονται με τα προϋπάρχοντα 4 LSB
out PORTD, r24 ; και δίνονται στην έξοδο
sbi PORTD, PD3 ; δημιουργείται παλμός Enable στον ακροδέκτη PD3
cbi PORTD, PD3 ; PD3=1 και μετά PD3=0
push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
push r25 ; λειτουργία του προγραμματος απομακρυσμένης
ldi r24 ,low(6000) ; πρόσβασης
ldi r25 ,high(6000)
rcall wait_usec
pop r25
pop r24 ; τέλος τμήμα κώδικα
pop r24 ; στέλνει τα 4 LSB. Ανακτάται το byte.
swap r24 ; εναλλάσσονται τα 4 MSB με τα 4 LSB
andi r24 ,0xf0 ; που με την σειρά τους αποστέλλονται
add r24, r25
out PORTD, r24
sbi PORTD, PD3 ; Νέος παλμός Enable
cbi PORTD, PD3
ret

lcd_data_sim:
push r24 ; αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
push r25 ; αλλάζουμε μέσα στη ρουτίνα
sbi PORTD, PD2 ; επιλογή του καταχωρητή δεδομένων (PD2=1)
rcall write_2_nibbles_sim ; αποστολή του byte
ldi r24 ,43 ; αναμονή 43μsec μέχρι να ολοκληρωθεί η λήψη
ldi r25 ,0 ; των δεδομένων από τον ελεγκτή της lcd
rcall wait_usec
pop r25 ;επανάφερε τους καταχωρητές r25:r24
```

```
pop r24
ret

lcd_command_sim:
push r24 ; αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
push r25 ; αλλάζουμε μέσα στη ρουτίνα
cbi PORTD, PD2 ; επιλογή του καταχωρητή εντολών (PD2=0)
rcall write_2_nibbles_sim ; αποστολή της εντολής και αναμονή 39μsec
ldi r24, 39 ; για την ολοκλήρωση της εκτέλεσης της από τον ελεγκτή της lcd.
ldi r25, 0 ; ΣΗΜ.: υπάρχουν δύο εντολές, οι clear display και return home,
rcall wait_usec ; που απαιτούν σημαντικά μεγαλύτερο χρονικό διάστημα.
pop r25 ; επανάφερε τους καταχωρητές r25:r24
pop r24
ret

lcd_init_sim:
push r24 ; αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
push r25 ; αλλάζουμε μέσα στη ρουτίνα

ldi r24, 40 ; Όταν ο ελεγκτής της lcd τροφοδοτείται με
ldi r25, 0 ; ρεύμα εκτελεί την δική του αρχικοποίηση.
rcall wait_msec ; Αναμονή 40 msec μέχρι αυτή να ολοκληρωθεί.
ldi r24, 0x30 ; εντολή μετάβασης σε 8 bit mode
out PORTD, r24 ; επειδή δεν μπορούμε να είμαστε βέβαιοι
sbi PORTD, PD3 ; για τη διαμόρφωση εισόδου του ελεγκτή
cbi PORTD, PD3 ; της οθόνης, η εντολή αποστέλλεται δύο φορές
ldi r24, 39
ldi r25, 0 ; εάν ο ελεγκτής της οθόνης βρίσκεται σε 8-bit mode
rcall wait_usec ; δεν θα συμβεί τίποτα, αλλά αν ο ελεγκτής έχει διαμόρφωση
 ; εισόδου 4 bit θα μεταβεί σε διαμόρφωση 8 bit
push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
push r25 ; λειτουργία του προγραμματος απομακρυσμένης
ldi r24,low(1000) ; πρόσβασης
ldi r25,high(1000)
rcall wait_usec
pop r25
pop r24 ; τέλος τμήμα κώδικα
ldi r24, 0x30
out PORTD, r24
sbi PORTD, PD3
cbi PORTD, PD3
ldi r24,39
ldi r25,0
rcall wait_usec
push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
```

```
push r25 ; λειτουργία του προγραμματος απομακρυσμένης
ldi r24 ,low(1000) ; πρόσβασης
ldi r25 ,high(1000)
rcall wait_usec
pop r25
pop r24 ; τέλος τμήμα κώδικα
ldi r24,0x20 ; αλλαγή σε 4-bit mode
out PORTD, r24
sbi PORTD, PD3
cbi PORTD, PD3
ldi r24,39
ldi r25,0
rcall wait_usec
push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
push r25 ; λειτουργία του προγραμματος απομακρυσμένης
ldi r24 ,low(1000) ; πρόσβασης
ldi r25 ,high(1000)
rcall wait_usec
pop r25
pop r24 ; τέλος τμήμα κώδικα
ldi r24,0x28 ; επιλογή χαρακτήρων μεγέθους 5x8 κουκίδων
rcall lcd_command_sim ; και εμφάνιση δύο γραμμών στην οθόνη
ldi r24,0x0c ; ενεργοποίηση της οθόνης, απόκρυψη του κέρσορα
rcall lcd_command_sim
ldi r24,0x01 ; καθαρισμός της οθόνης
rcall lcd_command_sim
ldi r24, low(1530)
ldi r25, high(1530)
rcall wait_usec
ldi r24 ,0x06 ; ενεργοποίηση αυτόματης αύξησης κατά 1 της διεύθυνσης
rcall lcd_command_sim ; που είναι αποθηκευμένη στον μετρητή διευθύνσεων και
 ; απενεργοποίηση της ολίσθησης ολόκληρης της οθόνης
pop r25 ; επανάφερε τους καταχωρητές r25:r24
pop r24
ret

main:
ldi r24, low(RAMEND) ;initialize stack pointer
out SPL, r24
ldi r24, high(RAMEND)
out SPH, r24
ldi r24, (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4) ; 4 MSB of PORTC as
outputs
out DDRC, r24
clr r24
```

```
out PORTC, r24  ;disable pull-ups
ser r24
out DDRB, r24   ;PORTB as output
out DDRD, r24   ;PORTD as output
sts _tmp_, r24  ;initialize _tmp_
clr r24
rcall lcd_init_sim ;initialize the lcd

digit1:
rcall scan_keypad_rising_edge_sim   ;read the first number
rcall keypad_to_ascii_sim           ;convert the button pressed to ascii
cpi r24,0x00                        ;check if a button was pressed otherwise read
again
breq digit1
mov r20,r24                         ;Store the ascii code in r20
subi r20,0x30                       ;then convert it to an integer

digit2:
rcall scan_keypad_rising_edge_sim   ;same for the second number
rcall keypad_to_ascii_sim
cpi r24,0x00
breq digit2
mov r21,r24                         ;Store the ascii code in r21
subi r21,0x30                       ;then convert it to an integer

cpi r20,0x04 ;Check if both digits of the password are correct
brne wrong
cpi r21,0x05
brne wrong

correct:        ;if they are
ser r20
out PORTB,r20   ;all leds on
ldi r24, 'W'    ;Print "WELCOME 45" on the lcd
rcall lcd_data_sim
ldi r24, 'E'
rcall lcd_data_sim
ldi r24, 'L'
rcall lcd_data_sim
ldi r24, 'C'
rcall lcd_data_sim
ldi r24, 'O'
rcall lcd_data_sim
ldi r24, 'M'
rcall lcd_data_sim
```

```
ldi r24, 'E'
rcall lcd_data_sim
ldi r24, ' '
rcall lcd_data_sim
ldi r24, '4'
rcall lcd_data_sim
ldi r24, '5'
rcall lcd_data_sim
ldi r20,0xBE    ;each call of scan_keypad_rising_edge_sim takes longer than 19ms
(19ms is the total delay time from delay routines called by
scan_keypad_rising_edge_sim()).
                ;we call it 190 times for a total delay ~4s

loop1:
dec r20
rcall scan_keypad_rising_edge_sim
cpi r20,0x00
brne loop1      ;keep reading from keypad and ignoring until 4s have passed (r20
becomes 0)
clr r20
out PORTB,r20   ;turn off the leds
ldi r24,0x01    ;clear the lcd
rcall lcd_command_sim
rjmp digit1     ;read a new code

wrong:          ;if the wrong password was inserted
ldi r24, 'A'    ;print "ALARM ON" on the lcd
rcall lcd_data_sim
ldi r24, 'L'
rcall lcd_data_sim
ldi r24, 'A'
rcall lcd_data_sim
ldi r24, 'R'
rcall lcd_data_sim
ldi r24, 'M'
rcall lcd_data_sim
ldi r24, ' '
rcall lcd_data_sim
ldi r24, 'O'
rcall lcd_data_sim
ldi r24, 'N'
rcall lcd_data_sim
ldi r20,0x04    ;total of 4 blinks
outerloop:
dec r20
```

```
ser r21
out PORTB,r21   ;PORTB leds on for 0.5s
ldi r21,0x18    ;each call of scan_keypad_rising_edge_sim takes longer than 19ms
(19ms is the total delay time from delay routines called by
scan_keypad_rising_edge_sim()).
                ;we call it 24 times for a total delay of ~0.5s

inner1:
dec r21
rcall scan_keypad_rising_edge_sim ;keep reading digits but ignoring them for 0.5s
cpi r21,0x00
brne inner1

clr r21
out PORTB,r21   ;then PORTB leds off for another 0.5s
ldi r21,0x18    ;each call of scan_keypad_rising_edge_sim takes longer than 19ms
(19ms is the total delay time from delay routines called by
scan_keypad_rising_edge_sim()).
                ;we call it 24 times for a total delay of ~0.5s

inner2:
dec r21
rcall scan_keypad_rising_edge_sim ;keep reading digits but ignoring them for 0.5s
cpi r21,0x00
brne inner2

cpi r20,0x00
brne outerloop  ;total of four blinks
ldi r24,0x01    ;clear the lcd
rcall lcd_command_sim
rjmp digit1     ;read a new password
```