ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ

(2020-2021)

3η ΟΜΆΔΑ ΑΣΚΉΣΕΩΝ

Ονοματεπώνυμο:

- Χρήστος Τσούφης

Αριθμός Μητρώου:

- 03117176

Ομάδα Εργαστηρίου:

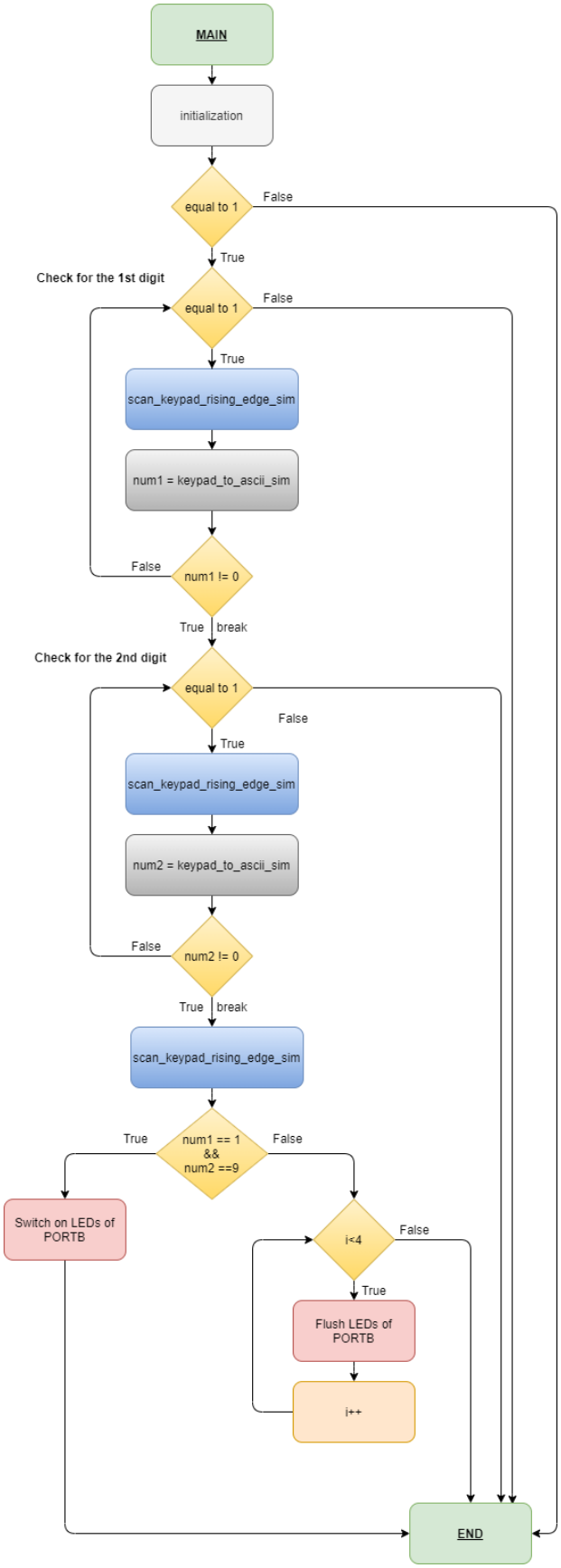
- B19

Εξέταση – Επίδειξη:

- 25/11/2020

1η Άσκηση

Το διάγραμμα ροής:



Ο πηγαίος κώδικας, μαζί με τα απαραίτητα σχόλια:

Κώδικας σε **C**:

```
#include <avr/io.h> // for avr library
#include <util/delay.h> // for functions with time delay
#define F_CPU 8000000UL // frequency

// the following function returns the line that has its bit pressed
// it also takes as input in r25, the state of the line and puts it in PORT C
char scan_row_sim(char grammi){
	char seira;
	PORTC = grammi;
	_delay_us(500); // time delay
	asm("nop"); // delay needed in order to change state
	asm("nop");
	seira = PINC & 0x0F; // keep 4 LSB of PINC
	return seira;
}

// this function sets an array that has two elements of 8 bits each
// it reads the previous func and puts in the 1st line 4 MSBs or in the
// 2nd line 4 LSBs in location[0] and similarly for location[1]
void scan_keypad_sim(char location[]){
	location[0] = (scan_row_sim(0x10)<<4) | (scan_row_sim(0x20));
	location[1] = (scan_row_sim(0x40)<<4) | (scan_row_sim(0x80));
	PORTC = 0; // used for distant access
}

// this function also has an array about location
// it is used in order to put the 1st call of the previous func which is
// the beginning of 'sparkle' control
// 1st result of the call goes to position[2] and 2nd goes to new_location[2]
void scan_keypad_rising_edge_sim(char tmp[], char result[]){
	char location[2], new_location[2];
	scan_keypad_sim(location);
	_delay_ms(15);
	scan_keypad_sim(new_location);
	location[0] = location[0] & new_location[0]; // & is used to keep ones
	location[1] = location[0] & new_location[1];
	result[0] = location[0] & ~tmp[0]; // ~ is used to keep the 0to1 states
	result[1] = location[1] & ~tmp[1];
	tmp[0] = location[0]; // has current state
	tmp[1] = location[1];
}

// returns ASCII code
char keypad_to_ascii_sim(char location[]){
	if(location[0] & 1) return '4';
	if(location[0] & 2) return '5';
	if(location[0] & 4) return '6';
	if(location[0] & 8) return 'B';
	if(location[0] & 16) return '1';
	if(location[0] & 32) return '2';
	if(location[0] & 64) return '3';
	if(location[0] & 128) return 'A';
	if(location[1] & 1) return '*';
	if(location[1] & 2) return '0';
	if(location[1] & 4) return '#';
	if(location[1] & 8) return 'D';
	if(location[1] & 16) return '7';
	if(location[1] & 32) return '8';
```

```
        if(location[1] & 64) return '9';
        if(location[1] & 128) return 'C';
        return 0;
}

int main(void)
{
        DDRC = 0xF0; // initialize PORT C as output
        DDRB = 0xFF; // initialize PORT B as input
        char tmp[2], result[2];
        tmp[0] = 0; tmp[1] = 0;
        result[0] = 0; result[1] = 1;
        char num1 = 0, num2 = 0;
        while(1){ // used for continuous operation
                while (1){ // continuous reading until ASCII ≠ 0 for 1st digit
                        scan_keypad_rising_edge_sim(tmp, result);
                        num1 = keypad_to_ascii_sim(result);
                        if(num1 != 0) break;
                }
                while (1){ // continuous reading until ASCII ≠ 0 for 2nd digit
                        scan_keypad_rising_edge_sim(tmp, result);
                        num2 = keypad_to_ascii_sim(result);
                        if(num2 != 0) break;
                }
                scan_keypad_rising_edge_sim(tmp, result); // read keyboard
                if ((num1 == '1') && (num2 == '9')) {
                        PORTB = 0xFF;
                        _delay_ms(4000);
                        PORTB = 0x00; // switch on LEDs of PORT B
                }
                else {
                        for (int i = 0; i<4; i++){ // flush LEDs 4 times
                                PORTB = 0xFF;
                                _delay_ms(500);
                                PORTB = 0x00;
                                _delay_ms(500);
                        }
                }
        }
}
```

2η Άσκηση

Ο πηγαίος κώδικας, μαζί με τα απαραίτητα σχόλια:

Κώδικας σε **assembly**:

```
        .INCLUDE "m16def.inc"

                                ; initialization
                                ; Αρχή τμήματος δεδομένων
        .DSEG
        _tmp_: .byte 2

                                ; Τέλος τμήματος δεδομένων
        .CSEG

        .org 0
        jmp start
```

```
start:
	ldi r24, (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
                                ; θέτει ως εξόδους τα 4 MSB
	out DDRC, r24               ; της θύρας PORTC

	ser r16
	out DDRB, r16               ; έξοδος το B
	ser r24
	out DDRD, r24               ; αρχικοποίηση στοίβας
	ldi r24, low(RAMEND)
	out SPL, r24
	ldi r24, high(RAMEND)
	out SPH, r24

	rjmp main

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)

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)

scan_row_sim:                   ; διαβάζει μια γραμμή του πληκτρολογίου και
επιστρέφει στο r24, r25 ποιο bit ποιας γραμμής είναι άσσος
	out PORTC, r25              ; η αντίστοιχη γραμμή τίθεται στο λογικό ‘1’
	push r24                    ; τμήμα κώδικα που προστίθεται για τη σωστή
λειτουργία του προγράμματος απομακρυσμένης πρόσβασης
	push r25
	ldi r24,low(500)
	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 οι διακόπτες
	ret
```

```
scan_keypad_sim:                ; αφορά τον έλεγχο του πληκτρολογίου για
πιεσμένους διακόπτες
        push r26                ; αποθήκευσε τους καταχωρητές r27:r26 γιατί
αλλάζουν μέσα στην ρουτίνα
        push r27
        ldi r25 , 0x10          ; έλεγξε την πρώτη γραμμή του πληκτρολογίου
(PC4: 1 2 3 A)
        rcall scan_row_sim
        swap r24                ; αποθήκευσε το αποτέλεσμα στα 4 msb του r27
        mov r27, r24
        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 στα 4 msb του r26
        mov r26, r24
        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:    ; ελέγχει τον σπινθηρισμό και κάνει το
διάβασμα
        push r22                ; αποθήκευσε τους καταχωρητές r23:r22 και
τους r26:r27 γιατί τους αλλάζουμε μέσα στην ρουτίνα
        push r23
        push r26
        push r27
        rcall scan_keypad_sim   ; έλεγξε το πληκτρολόγιο για πιεσμένους
διακόπτες και αποθήκευσε το αποτέλεσμα
        push r24
        push r25
        ldi r24 ,15             ; καθυστέρησε 15 ms (τυπικές τιμές 10-20 msec
που καθορίζεται από τον κατασκευαστή του πληκτρολογίου – χρονοδιάρκεια
σπινθηρισμών)
        ldi r25 ,0
        rcall wait_msec
        rcall scan_keypad_sim   ; έλεγξε το πληκτρολόγιο ξανά και απόρριψε
όσα πλήκτρα εμφανίζουν σπινθηρισμό
        pop r23
        pop r22
        and r24 ,r22
        and r25 ,r23
        ldi r26 ,low(_tmp_)     ; φόρτωσε την κατάσταση των διακοπτών στην
προηγούμενη κλήση της ρουτίνας στους r27:r26
        ldi r27 ,high(_tmp_)
        ld r23 ,X+
        ld r22 ,X
        st X ,r24               ; αποθήκευσε στη RAM τη νέα κατάσταση των
διακοπτών
        st -X ,r25
        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 γιατί
τους αλλάζουμε μέσα στη ρουτίνα
        push r27
        movw r26 ,r24               ; λογικό ‘1’ στις θέσεις του καταχωρητή r26
δηλώνουν τα παρακάτω σύμβολα και αριθμούς
        ldi r24 ,'*'
                                    ; r26
                                    ; C 9 8 7 D # 0 *
        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 τα παρακάτω σύμβολα και αριθμούς
        sbrc r27 ,0
        rjmp return_ascii
        ldi r24 ,'5'
                                    ; r27
                                    ; A 3 2 1 B 6 5 4
        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 προγράμματος απομακρυσμένης
πρόσβασης
        ldi r24 ,low(6000)
        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 χαλάσει η 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 PD3=0
        push r24                    ; τμήμα κώδικα που προστίθεται για τη σωστή
        push r25                    ; λειτουργία του προγράμματος 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 αλλάζουμε μέσα στη ρουτίνα
        push r25
        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			; hint: υπάρχουν δύο εντολές, οι clear
display και return home, που απαιτούν σημαντικά μεγαλύτερο χρονικό διάστημα
	rcall wait_usec
	pop r25				; επανάφερε τους καταχωρητές r25:r24
	pop r24
	ret

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

	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 δεν θα συμβεί τίποτα
	rcall wait_usec			; 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 προγράμματος 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 προγράμματος απομακρυσμένης
πρόσβασης
	ldi r24 ,low(1000)
	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 προγράμματος απομακρυσμένης
πρόσβασης
        ldi r24 ,low(1000)
        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:                               ; αρχικοποίηση
        rcall scan_keypad_rising_edge_sim

attempt1:

        rcall scan_keypad_rising_edge_sim
                                    ; για να είναι συνεχούς λειτουργίας
        rcall keypad_to_ascii_sim   ; μετατροπή σε ASCII
        cpi r24, 0x00
        breq attempt1               ; loop μέχρι ≠ 0
        mov r28, r24

attempt2:
        rcall scan_keypad_rising_edge_sim
                                    ; για να είναι συνεχούς λειτουργίας
        rcall keypad_to_ascii_sim   ; μετατροπή σε ASCII
        cpi r24, 0x00
        breq attempt2               ; loop μέχρι ≠ 0
        mov r29, r24
        rcall scan_keypad_rising_edge_sim
        cpi r28, '1'                ; αν δεν είναι 1, πήγαινε στο alarmon
        brne alarmon
        cpi r29, '9'                ; αν δεν είναι 9, πήγαινε στο alarmon
        brne alarmon
        rjmp welc

wrong_password:                     ; αναβοσβήνει 4 φορές τα LED

        ser r16
        out PORTB, r16
        ldi r24, 0xF4
        ldi r25, 0x01
```

```
        rcall wait_msec
        clr r16
        out PORTB, r16
        ldi r24, 0xF4
        ldi r25, 0x01
        rcall wait_msec
        ser r16
        out PORTB, r16
        ldi r24, 0xF4
        ldi r25, 0x01
        rcall wait_msec
        clr r16
        out PORTB, r16
        ldi r24, 0xF4
        ldi r25, 0x01
        rcall wait_msec
        ser r16
        out PORTB, r16
        ldi r24, 0xF4
        ldi r25, 0x01
        rcall wait_msec
        clr r16
        out PORTB, r16
        ldi r24, 0xF4
        ldi r25, 0x01
        rcall wait_msec
        ser r16
        out PORTB, r16
        ldi r24, 0xF4
        ldi r25, 0x01
        rcall wait_msec
        clr r16
        out PORTB, r16
        ldi r24, 0xF4
        ldi r25, 0x01
        rcall wait_msec
        rcall lcd_init_sim          ; αρχικοποίηση οθόνης
        rjmp attempt1

correct_password:                   ; κρατάει αναμμένα τα LED
        ser r16
        out PORTB, r16
        ldi r24, 0xA0
        ldi r25, 0x0F
        rcall wait_msec
        clr r16
        out PORTB, r16
        rcall lcd_init_sim          ; αρχικοποίηση οθόνης
        rjmp attempt1

alarmon:                            ; εμφανίζει το μήνυμα "ALARM ON"
        clr r24
        rcall lcd_init_sim          ; αρχικοποίηση οθόνης
        ldi r24, 'A'
        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
        rcall scan_keypad_rising_edge_sim
        rjmp wrong_password


welc:                               ; εμφανίζει το μήνυμα “WELCOME 19”
        clr r24
        rcall lcd_init_sim          ; αρχικοποίηση οθόνης
        ldi r24, 'W'
        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, '1'
        rcall lcd_data_sim
        ldi r24, '9'
        rcall lcd_data_sim
        rcall scan_keypad_rising_edge_sim
        rjmp correct_password
```