Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Μικροϋπολογιστών 2021-2022

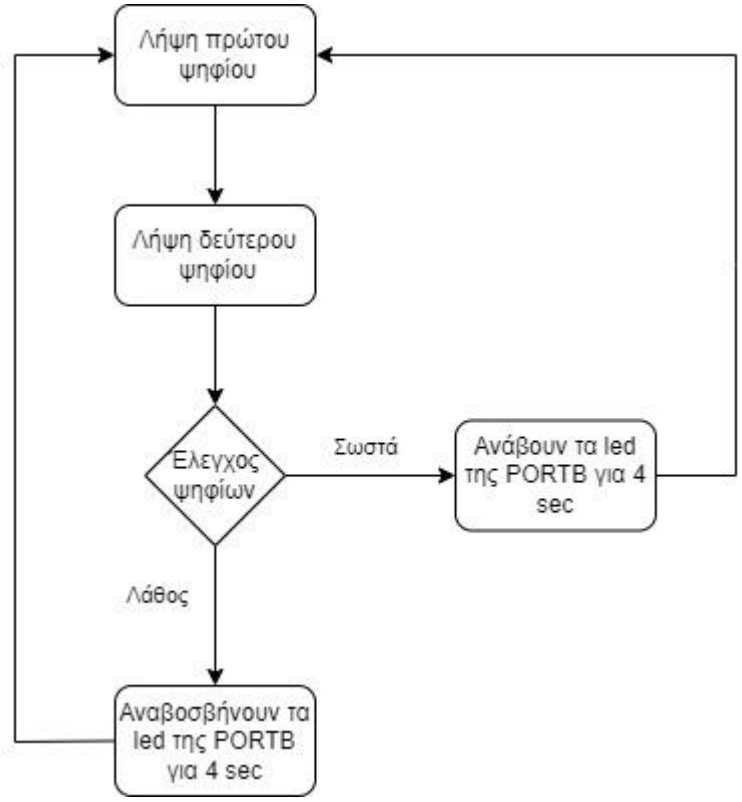
3η Εργασία (AVR)

Κωνσταντίνος Σιδέρης, Ομάδα 48

Α.Μ.: 03118134

Άσκηση 1

Ακολουθεί το διάγραμμα ροής του προγράμματος της άσκησης 1:



Ακολουθεί ο κώδικας της άσκησης 1 (C) με σχόλια:

```
#define F_CPU 8000000
#include <avr/io.h>
#include <util/delay.h>

char reg[2], mem[2], fd, sd;

char swap(char x) { //Υλοποίηση εντολής swap
	return ((x & 0x0F) << 4 | (x & 0xF0) >>4);
}

char scan_row(int r) { //Ρουτίνα scan_row_sim
	char o = 0x08;
	o = (o << r); //Ολίσθηση r φορές (όπου r αριθμός γραμμής) του o = 00001000
	PORTC = o; //Θέτουμε την γραμμή που σκανάρουμε σε 1
	_delay_us(500); //Καθυστέρηση 500ms για απομακρυσμένη λειτουργία
	return PINC & 0x0F; //Επιστροφή τεσσάρων  LSB της PORTC (στήλες πληκτρολογίου)
}

void scan_keypad() { //Ρουτίνα scan_keypad_sim
	char c;

	c = scan_row(1); //Σκανάρισμα πρώτης σειράς
	reg[1] = swap(c); //Αποθήκευση στα 4 MSB των πρώτων 8 bit του 16-bit register
reg
```

```
	c = scan_row(2); //Σκανάρισμα δεύτερης σειράς
	reg[1] = reg[1] + c; //Αποθήκευση στα 4 LSB των πρώτων 8 bit του 16-bit
register reg

	c = scan_row(3); //Σκανάρισμα τρίτης σειράς
	reg[0] = swap(c); //Αποθήκευση στα 4 MSB των δεύτερων 8 bit του 16-bit register
reg

	c = scan_row(4); //Σκανάρισμα τέταρτης σειράς
	reg[0] = reg[0] + c; //Αποθήκευση στα 4 LSB των δεύτερων 8 bit του 16-bit
register reg

	PORTC = 0x00;
}

int scan_keypad_rising_edge() {  //Ρουτίνα scan_row_rising_edge_sim
	char tmp[2];
	scan_keypad(); //Σκανάρισμα πληκτρολογίου
	tmp[0] = reg[0]; //Προσωρινή αποθήκευση του αποτελέσματος
	tmp[1] = reg[1];
	_delay_ms(15); //Αναμονή 15ms λόγω σπινθηρισμών
	scan_keypad(); //Δεύτερο σκανάρισμα πληκτρολογίου

	reg[0] = reg[0] & tmp[0]; //Απόρριψη πλήκτρων που εμφάνισαν σπινθηρισμό
	reg[1] = reg[1] & tmp[1];

	tmp[0] = mem[0]; //Λήψη προηγούμενης κατάστασης διακοπτών από την  RAM
	tmp[1] = mem[1];

	mem[0] = reg[0]; //Αποθήκευση τωρινής κατάστασης διακοπτών από την  RAM
	mem[1] = reg[1];

	reg[0] = reg[0] & (~tmp[0]); //Εύρεση διακοπτών που έχουν μόλις πατηθεί
	reg[1] = reg[1] & (~tmp[1]);

	return (reg[0] || reg[1]); //Επιστροφή των διακοπτών που μόλις πατήθηκαν (0 αν
δεν έχει πατηθεί πλήκτρο)
}

char keypad_to_ascii() { //Ρουτίνα keypad_to_ascii_sim
	if ((reg[0]&0x01) == 0x01)
	return '*'; //Εύρεση πατημένου πλήκτρου και επιστροφή του κωδικού ASCII που του
αντιστοιχεί
	if ((reg[0]&0x02) == 0x02)
	return '0';
	if ((reg[0]&0x04) == 0x04)
	return '#';
	if ((reg[0]&0x08) == 0x08)
	return 'D';
	if ((reg[0]&0x10) == 0x10)
	return '7';
	if ((reg[0]&0x20) == 0x20)
	return '8';
```

```
	if ((reg[0]&0x40) == 0x40)
	return '9';
	if ((reg[0]&0x80) == 0x80)
	return 'C';
	if ((reg[1]&0x01) == 0x01)
	return '4';
	if ((reg[1]&0x02) == 0x02)
	return '5';
	if ((reg[1]&0x04) == 0x04)
	return '6';
	if ((reg[1]&0x08) == 0x08)
	return 'B';
	if ((reg[1]&0x10) == 0x10)
	return '1';
	if ((reg[1]&0x20) == 0x20)
	return '2';
	if ((reg[1]&0x40) == 0x40)
	return '3';
	if ((reg[1]&0x80) == 0x80)
	return 'A';

	return 0; //Εάν δεν έχει πατηθεί κάποιο πλήκτρο επιστρέφει 0
}

void welcome() { //Συνάρτηση welcome που ανάβει τα led για 4sec(σωστός κωδικός)
	char i;
	PORTB = 0xFF;//Ανάβουμε τα led της PORTB
	for (i = 1; i <= 160; i++) { //160 επαναλήψεις διάρκειας 25ms: συνολικά 4sec
		scan_keypad_rising_edge(); //Διάβασμα πληκτρολογίου κατά την διάρκεια
ανάμματος των led (19ms)
		_delay_ms(6); //Πρόσθετη καθυστέρηση 6ms ώστε να έχουμε συνολικά 25ms
	}
	PORTB = 0x00;//Σβήνουμε τα  led της PORTB
}

void alarm() { //Συνάρτηση alarm που αναβοσβήνει τα led για 4sec(λάθος κωδικός)
	char i, j;
	for (j = 1; j <= 4; j++) { //4 επαναλήψεις διάρκειας 1sec: συνολικά 4sec
		PORTB = 0xFF;//Ανάβουμε τα led της PORTB
		for (i = 1; i <= 20; i++) { //2 επαναλήψεις διάρκειας 25ms: συνολικά 0.5sec
			scan_keypad_rising_edge(); //Διάβασμα πληκτρολογίου κατά την διάρκεια
που αναβοσβήνουν τα led (19ms)
			_delay_ms(6); //Πρόσθετη καθυστέρηση 6ms ώστε να έχουμε συνολική
καθυστέρηση 25ms
		}
		PORTB = 0x00;//Σβήνουμε τα  led της PORTB
		for (i = 1; i <= 20; i++) { //2 επαναλήψεις διάρκειας 25ms: συνολικά 0.5sec
			scan_keypad_rising_edge(); //Διάβασμα πληκτρολογίου κατά την διάρκεια
που αναβοσβήνουν τα led (19ms)
			_delay_ms(6); //Πρόσθετη καθυστέρηση 6ms ώστε να έχουμε συνολική
καθυστέρηση 25ms
		}
	}
```

```
}

int main(void) {
	DDRB = 0xFF; //Αρχικοποίηση PORTB ως έξοδο
	DDRC = 0xF0; //Αρχικοποίηση 4ων MSB της PORTB ως έξοδο και 4ων LSB της PORTB ως είσοδο

	while (1) {
		mem[0] = 0x00; //Αρχικοποίηση μεταβλητής μνήμης RAM
		mem[1] = 0x00;
		PORTB = 0x00;

		while (1) {
			if (scan_keypad_rising_edge() != 0) { //Διάβασμα πληκτρολογίου μέχρι να πατηθεί το 1ο πλήκτρο
				fd = keypad_to_ascii(); //Μετατροπή σε ψηφίο ASCII
				break;
			}
		}

		while (1) {
			if (scan_keypad_rising_edge() != 0) { //Διάβασμα πληκτρολογίου μέχρι να πατηθεί το 2ο πλήκτρο
				sd = keypad_to_ascii(); //Μετατροπή σε ψηφίο ASCII
				scan_keypad_rising_edge();
				break;
			}
		}

		if (fd != '4' || sd != '8') {
			alarm(); //Εάν δεν έχουν πατηθεί τα σωστά πλήκτρα καλούμε την alarm
		}
		else {
			welcome(); //Εάν έχουν πατηθεί τα σωστά πλήκτρα καλούμε την welcome
		}

	}
	return 0;
}
```

Άσκηση 2

Ακολουθεί ο κώδικας της άσκησης 2 (Assembly) με σχόλια:

```
.DSEG
_tmp_: .byte 2

.CSEG
.include "m16def.inc"

.macro welcome              ;Macro welcome που ανάβει τα led για 4sec(σωστός κωδικός)
            rcall lcd_init_sim
            ldi r24,'W' ;Εμφάνιση μηνύματος WELCOME 48 στην οθόνη 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,'8'
            rcall lcd_data_sim

            ldi r19, 0xA0 ;160 επαναλήψεις διάρκειας 25ms: συνολική καθυστέρηση 4sec
keep_on:
            ser r18
            out PORTB,r18 ;Ανάβουμε τα led της PORTB
            rcall scan_keypad_rising_edge_sim ;Διάβασμα πληκτρολογίου κατά την διάρκεια
ανάμματος των led (19ms)
            ldi r24,low(6)
            ldi r25,high(6)
            rcall wait_msec ;Πρόσθετη καθυστέρηση 6ms ώστε να έχουμε συνολικά 25ms
            dec r19
            cpi r19, 0x00
            brne keep_on
            clr r18
            out PORTB,r18;Σβήνουμε τα  led της PORTB
.endmacro

.macro alarm                ;Macro alarm που αναβοσβήνει τα led για 4sec(λάθος κωδικός)
            rcall lcd_init_sim
            ldi r24,'A' ;Εμφάνιση μηνύματος ALARM ON στην οθόνη 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 ;4 επαναλήψεις διάρκειας 1sec: συνολική καθυστέρηση 4sec
blink:
            ldi r19, 0x14 ;20 επαναλήψεις διάρκειας 25ms: συνολική καθυστέρηση 0.5sec
blink1:
            ser r18       ;Ανάβουμε τα led της PORTB
            out PORTB,r18 ;Διάβασμα πληκτρολογίου κατά την διάρκεια που αναβοσβήνουν τα
led (19ms)
            rcall scan_keypad_rising_edge_sim
            ldi r24,low(6)
            ldi r25,high(6)
            rcall wait_msec ;Πρόσθετη καθυστέρηση 6ms ώστε να έχουμε συνολική
καθυστέρηση 25ms
            dec r19
            cpi r19, 0x00
            brne blink1

            ldi r19, 0x14 ;20 επαναλήψεις διάρκειας 25ms: συνολική καθυστέρηση 0.5sec
blink2:
            clr r18 ;Σβήνουμε τα  led της PORTB
            out PORTB,r18 ;Διάβασμα πληκτρολογίου κατά την διάρκεια που αναβοσβήνουν τα
led (19ms)
            rcall scan_keypad_rising_edge_sim
            ldi r24,low(6)
            ldi r25,high(6)
            rcall wait_msec ;Πρόσθετη καθυστέρηση 6ms ώστε να έχουμε συνολικά 25ms
            dec r19
            cpi r19, 0x00
            brne blink2

            dec r20
            cpi r20, 0x00
            brne blink
.endmacro

.org 0x00
rjmp reset

reset:
            ldi    r18, LOW(RAMEND) ;Αρχικοποίηση stack pointer
            out    spl, r18
```

```
            ldi    r18, HIGH(RAMEND)
            out    sph, r18

            ldi r24, (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
            out DDRC, r24 ;Αρχικοποίηση 4ων MSB της PORTB ως έξοδο και 4ων LSB της
PORTB ως είσοδο
            ser r24
            out DDRD, r24 ;Αρχικοποίηση PORTB ως έξοδο
            out DDRB, r24 ;Αρχικοποίηση PORTD ως έξοδο

digit_1:
            ldi r21, 0x00 ;Αρχικοποίηση flag (r21) ορθότητας 1ου ψηφίου
            rcall scan_keypad_rising_edge_sim ;Διάβασμα πληκτρολογίου
            rcall keypad_to_ascii_sim ;Μετατροπή σε ψηφίο ASCII
            cpi r24, 0x00
            breq digit_1 ;Διάβασμα πληκτρολογίου μέχρι να πατηθεί το 1ο πλήκτρο
            cpi r24, '4' ;Έλεγχος 1ου ψηφίου
            breq digit_2
            ldi r21, 0x01;Εάν είναι λάθος θέτουμε το flag σε 1

digit_2:
            rcall scan_keypad_rising_edge_sim ;Διάβασμα πληκτρολογίου
            rcall keypad_to_ascii_sim ;Μετατροπή σε ψηφίο ASCII
            cpi r24, 0x00
            breq digit_2 ;Διάβασμα πληκτρολογίου μέχρι να πατηθεί το 2ο πλήκτρο
            cpi r21, 0x01 ;Έλεγχος flag ορθότητας 1ου ψηφίου
            breq wrong_pass ;Αν είναι 1 πηγαίνουμε στο λάθος password
            cpi r24, '8' ;Έλεγχος 2ου ψηφίου
            brne wrong_pass ;Αν δεν είναι σωστό πηγαίνουμε στο λάθος password

correct_pass:               ;Εάν φτάσουμε εδώ έχουμε σωστό password
            rcall scan_keypad_rising_edge_sim
            welcome ;Κλήση macro welcome
            rjmp digit_1

wrong_pass:                 ;Εάν φτάσουμε εδώ έχουμε λάθος  password
            rcall scan_keypad_rising_edge_sim
            alarm ;Κλήση macro alarm
            rjmp digit_1

scan_row_sim:               ;Υλοποίηση ρουτίνας scan_row_sim
            out PORTC, r25
            push r24
            push r25
            ldi r24,low(500)
            ldi r25,high(500)
            rcall wait_usec
            pop r25
            pop r24
            nop
            nop
            in r24, PINC
            andi r24 ,0x0f
```

```
            ret

scan_keypad_sim:          ;Υλοποίηση ρουτίνας scan_keypad_sim
            push r26
            push r27
            ldi r25 , 0x10
            rcall scan_row_sim
            swap r24
            mov r27, r24
            ldi r25 ,0x20
            rcall scan_row_sim
            add r27, r24
            ldi r25 , 0x40
            rcall scan_row_sim
            swap r24
            mov r26, r24
            ldi r25 ,0x80
            rcall scan_row_sim
            add r26, r24
            movw r24, r26
            clr r26
            out PORTC,r26
            pop r27
            pop r26
            ret

scan_keypad_rising_edge_sim:     ;Υλοποίηση ρουτίνας scan_row_rising_edge_sim
            push r22
            push r23
            push r26
            push r27
            rcall scan_keypad_sim
            push r24
            push r25
            ldi r24 ,15
            ldi r25 ,0
            rcall wait_msec
            rcall scan_keypad_sim
            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
            pop r27
```

```
            pop r26
            pop r23
            pop r22
            ret

keypad_to_ascii_sim:            ;Υλοποίηση ρουτίνας keypad_to_ascii_sim
            push r26
            push r27
            movw r26 ,r24
            ldi r24 ,'*'
            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
            rjmp return_ascii
            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'
            sbrc r27 ,0
            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
            pop r26
            ret

write_2_nibbles_sim:            ;Υλοποίηση ρουτίνας write_2_nibbles_sim
            push r24
            push r25
            ldi r24 ,low(6000)
            ldi r25 ,high(6000)
            rcall wait_usec
            pop r25
            pop r24
            push r24
            in r25, PIND
            andi r25, 0x0f
            andi r24, 0xf0
            add r24, r25
            out PORTD, r24
            sbi PORTD, PD3
            cbi PORTD, PD3
            push r24
            push r25
            ldi r24 ,low(6000)
            ldi r25 ,high(6000)
            rcall wait_usec
            pop r25
            pop r24
            pop r24
            swap r24
            andi r24 ,0xf0
            add r24, r25
            out PORTD, r24
            sbi PORTD, PD3
            cbi PORTD, PD3
            ret

lcd_data_sim:             ;Υλοποίηση ρουτίνας lcd_data_sim
            push r24
            push r25
            sbi PORTD, PD2
            rcall write_2_nibbles_sim
            ldi r24 ,43
            ldi r25 ,0
            rcall wait_usec
            pop r25
            pop r24
```

```
            ret

lcd_command_sim:          ;Υλοποίηση ρουτίνας lcd_command_sim
            push r24
            push r25
            cbi PORTD, PD2
            rcall write_2_nibbles_sim
            ldi r24, 39
            ldi r25, 0
            rcall wait_usec
            pop r25
            pop r24
            ret

lcd_init_sim:             ;Υλοποίηση ρουτίνας lcd_init_sim
            push r24
            push r25

            ldi r24, 40
            ldi r25, 0
            rcall wait_msec
            ldi r24, 0x30
            out PORTD, r24
            sbi PORTD, PD3
            cbi PORTD, PD3
            ldi r24, 39
            ldi r25, 0
            rcall wait_usec
            push r24
            push r25
            ldi r24,low(1000)
            ldi r25,high(1000)
            rcall wait_usec
            pop r25
            pop r24
            ldi r24, 0x30
            out PORTD, r24
            sbi PORTD, PD3
            cbi PORTD, PD3
            ldi r24,39
            ldi r25,0
            rcall wait_usec
            push r24
            push r25
            ldi r24 ,low(1000)
            ldi r25 ,high(1000)
            rcall wait_usec
            pop r25
            pop r24
            ldi r24,0x20
            out PORTD, r24
            sbi PORTD, PD3
            cbi PORTD, PD3
```

```
            ldi r24,39
            ldi r25,0
            rcall wait_usec
            push r24
            push r25
            ldi r24 ,low(1000)
            ldi r25 ,high(1000)
            rcall wait_usec
            pop r25
            pop r24
            ldi r24,0x28
            rcall lcd_command_sim
            ldi r24,0x0c
            rcall lcd_command_sim
            ldi r24,0x01
            rcall lcd_command_sim
            ldi r24, low(1530)
            ldi r25, high(1530)
            rcall wait_usec
            ldi r24 ,0x06
            rcall lcd_command_sim
            pop r25
            pop r24
            ret

wait_msec:              ;Υλοποίηση ρουτίνας 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:              ;Υλοποίηση ρουτίνας wait_usec
            sbiw r24 ,1
            nop
            nop
            nop
            nop
            brne wait_usec
            ret
```