Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Μ Α Θ Η Μ Α
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ (ΡΟΗ Υ)

Ε Ρ Γ Α Σ Ι Α
7Η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

Σ Π Ο Υ Δ Α Σ Τ Ε Σ
ΚΑΡΑΜ ΚΩΝΣΤΑΝΤΙΝΟΣ
ΚΟΛΙΟΣ ΑΠΟΣΤΟΛΟΣ

2022–23

Η πλήρεις κώδικες όλων των ασκήσεων περιέχονται στο zipped αρχείο. Για την υλοποίηση των ζητημάτων έχει χρησιμοποιηθεί ο αισθητήρας DS18B20.

Ζήτημα 7.1 *(Lab7_1.c)*

Ο κώδικας του προγράμματος φαίνεται παρακάτω:

```
#include <avr/io.h>
#define     F_CPU   16000000UL
#include <util/delay.h>
#define cbi(reg,bit) (reg &= ~(1 << bit))
#define sbi(reg,bit) (reg |= (1 << bit))

uint8_t one_wire_receive_bit() {
    uint8_t bit,temp;
    sbi(DDRD,PD4);
    cbi(PORTD,PD4);
    _delay_us(2);
    cbi(DDRD,PD4);
    cbi(PORTD,PD4);
    _delay_us(10);
    temp = (PIND & 0x10);
    bit = 0x00;
    if (temp == 0x10) bit = 0x01;
    _delay_us(49);
    return bit;
}

uint8_t one_wire_receive_byte() {
```

```
    uint8_t bit;
    uint8_t byte = 0x00;
    uint8_t i = 0x08;
    while(i != 0){
        bit = one_wire_receive_bit();
        byte = (byte >> 1);
        if (bit == 0x01) bit = 0x80;
        byte = (byte | bit);
        i--;
    }
    return byte;
}

void one_wire_transmit_bit(uint8_t bit) {
    sbi(DDRD,PD4);
    cbi(PORTD,PD4);
    _delay_us(2);
    if (bit == 0x01) sbi(PORTD,PD4);
    if (bit == 0x00) cbi(PORTD,PD4);
    _delay_us(58);
    cbi(DDRD,PD4);
    cbi(PORTD,PD4);
    _delay_us(1);
    return;
}

void one_wire_transmit_byte(uint8_t byte) {
    uint8_t bit;
    uint8_t i = 0x08;
    while(i != 0){
        bit = (byte & 0x01);
        one_wire_transmit_bit(bit);
        byte = (byte >> 1);
        i--;
    }
    return;
}

uint8_t one_wire_reset() {
    sbi(DDRD,PD4);
    cbi(PORTD,PD4);
    _delay_us(480);
    cbi(DDRD,PD4);
    cbi(PORTD,PD4);
    _delay_us(100);
```

```
    uint8_t temp = PIND;
    _delay_us(380);
    temp = (temp & 0x10);
    uint8_t res = 0x00;
    if (temp == 0x00)
        res = 0x01;
    return res;
}


int main(void)
{
    DDRB = 0x3F;
    DDRD = 0xFF;

    uint8_t temp_lo, temp_hi, temp_sign;
    uint16_t temp_final, temp_hi_16, temp_final_out;

    while (1)
    {
        // Check if device is connected
        if (!one_wire_reset()) {
            temp_final_out = 0x8000;
            continue;
        }
        one_wire_transmit_byte(0xCC); // Send command 0xCC
        one_wire_transmit_byte(0x44); // Send command 0x44

        while(one_wire_receive_bit() != 0x01);

        // Recheck if device is connected
        if (!one_wire_reset()) {
            temp_final_out = 0x8000;
            continue;
        }

        one_wire_transmit_byte(0xCC); // Send command 0xCC
        one_wire_transmit_byte(0xBE); // Send command 0xBE

        /* SAVE TEMPRATURE VALUE */
        temp_lo = one_wire_receive_byte();
        temp_hi = one_wire_receive_byte();
        temp_sign = temp_hi & 0xF8;
        temp_hi_16 = temp_hi << 8;
        temp_final = temp_hi_16 + temp_lo;
```

```
        // Check if temperature is negative or positive
        if (temp_sign == 0xF8)
            temp_final_out = ~(temp_final) + 1;   // Two's compliment
        else
            temp_final_out = temp_final;
    }
}
```

Θα εξηγήσουμε εν συντομία την λειτουργία του προγράμματος. Οι συναρτήσεις υλοποιήθηκαν με παρόμοια δομή με αυτήν της assembly. Οι εντολές *cbi* και *sbi*, ορίστηκαν στην αρχή του κώδικα μέσω του *define,* ώστε να χρησιμοποιούνται όπως ακριβώς και στην assembly. Τέλος, η θερμοκρασία αποθηκεύεται στην μεταβλητή *temp_final_out*, όπως αυτή διαβάζεται με την κλίση της ρουτίνας *one_wire_receive_byte* δύο φορές. Σε περίπτωση που η θερμοκρασία είναι αρνητική, υπολογίζεται το συμπλήρωμα ως προς δύο, ενώ εάν δεν υπάρχει συνδεδεμένη σκεύη αποθηκεύεται η τιμή 0x8000.

Ζήτημα 7.2 *(Lab7_2.c)*

Το κομμάτι του κώδικα για την υλοποίηση της εμφάνισης της θερμοκρασίας στην οθόνη φαίνεται παρακάτω:

```
int main(void)
{

    twi_init();

    DDRB = 0x3F;
    DDRD = 0xFF;

    PCA9555_0_write(REG_CONFIGURATION_0, 0x00);
    PCA9555_0_write(REG_CONFIGURATION_1, 0xF0);

    uint8_t temp_lo, temp_hi, temp_sign, temp_dec;
    uint16_t temp_final, temp_hi_16, temp_final_out;

    LCD_init();
    _delay_ms(2);

    while (1)
    {
        // Check if device is connected
        if (!one_wire_reset()) {
```

```
            LCD_command(0x01);
            _delay_ms(2);
            LCD_data('N');
            LCD_data('O');
            LCD_data(' ');
            LCD_data('D');
            LCD_data('E');
            LCD_data('V');
            LCD_data('I');
            LCD_data('C');
            LCD_data('E');
            continue;
        }
        one_wire_transmit_byte(0xCC); // Send command 0xCC
        one_wire_transmit_byte(0x44); // Send command 0x44

        while(one_wire_receive_bit() != 0x01);

        // Recheck if device is connected
        if (!one_wire_reset()) {
            LCD_command(0x01);
            _delay_ms(2);
            LCD_data('N');
            LCD_data('O');
            LCD_data(' ');
            LCD_data('D');
            LCD_data('E');
            LCD_data('V');
            LCD_data('I');
            LCD_data('C');
            LCD_data('E');
            continue;
        }

        one_wire_transmit_byte(0xCC); // Send command 0xCC
        one_wire_transmit_byte(0xBE); // Send command 0xBE

        /* SAVE TEMPRATURE VALUE */

        temp_lo = one_wire_receive_byte();
        temp_hi = one_wire_receive_byte();
        temp_sign = temp_hi & 0xF8;
        temp_hi_16 = temp_hi << 8;
        temp_final = temp_hi_16 + temp_lo;
        temp_dec = temp_lo & 0x0F;
```

```
 // Check if temperature is negative or positive
if (temp_sign == 0xF8)
    temp_final_out = ~(temp_final) + 1;   // Two's compliment
else
    temp_final_out = temp_final;

/* DECIMAL PART CALCULATE */

temp_dec = temp_final_out & 0x0F;

int dec_1 = 0;
int dec_2 = 0;
int dec_3 = 0;
int dec_4 = 0;
int dec_sum = 0;
uint8_t bit_one;

bit_one = temp_dec & 0x08;
if (bit_one == 8)
    dec_sum = dec_sum + 5000;

bit_one = temp_dec & 0x04;
if (bit_one == 4)
    dec_sum = dec_sum + 2500;

bit_one = temp_dec & 0x02;
if (bit_one == 2)
    dec_sum = dec_sum + 1250;

bit_one = temp_dec & 0x01;
if (bit_one == 1)
    dec_sum = dec_sum + 625;


while (dec_sum >= 1000) {
    dec_1++;
    dec_sum = dec_sum - 1000;
}

while (dec_sum >= 100) {
    dec_2++;
    dec_sum = dec_sum - 100;
}
```

```
        while (dec_sum >= 10) {
            dec_3++;
            dec_sum = dec_sum - 10;
        }

        while (dec_sum >= 1) {
            dec_4++;
            dec_sum = dec_sum - 1;
        }

        /* AKERAIO PART CALCULATE */

        temp_final_out = temp_final_out>>4;

        int ak_100 = 0;
        int ak_10 = 0;
        int ak_1 = 0;

        while (temp_final_out >= 100) {
            ak_100++;
            temp_final_out = temp_final_out - 100;
        }

        while (temp_final_out >= 10) {
            ak_10++;
            temp_final_out = temp_final_out - 10;
        }

        while (temp_final_out >= 1) {
            ak_1++;
            temp_final_out = temp_final_out - 1;
        }

        /* PRINT ON LCD */

        ak_100 |= 0x30;
        ak_10 |= 0x30;
        ak_1 |= 0x30;
        dec_1 |= 0x30;
        dec_2 |= 0x30;
        dec_3 |= 0x30;
        dec_4 |= 0x30;

        LCD_command(0x01);
        _delay_ms(2);
```

```
        if (temp_sign == 0xF8)
            LCD_data('-');
        else
            LCD_data('+');

        if(ak_100 != 0x30)
            LCD_data(ak_100);
        if(ak_10 != 0x30)
            LCD_data(ak_10);
        LCD_data(ak_1);
        LCD_data('.');
        LCD_data(dec_1);
        LCD_data(dec_2);
        LCD_data(dec_3);
        LCD_data(dec_4);
        LCD_data(0b00100010);
        LCD_data('C');
    }
}
```

Θα εξηγήσουμε εν συντομία την λειτουργία του προγράμματος. Αρχικά αφού διαβάσουμε κι ελέγξουμε το πρόσημο της θερμοκρασίας, αποθηκεύουμε την τελική τιμή στην μεταβλητή *temp_final_out*, και χωρίζουμε το δεκαδικό και το ακέραιο μέρος. Για το δεκαδικό μέρος υπολογίζουμε ένα-ένα τα τέσσερα δεκαδικά ψηφία με τον εξής τρόπο. Ανάλογα με τη θέση που βρίσκεται ο κάθε άσσος στο δεκαδικό μέρος, δημιουργούμε ένα άθροισμα *dec_sum*, στο οποίο προσθέτουμε κάθε φορά έναν αριθμό αντίστοιχο της θέσης του άσσου, σύμφωνα με τον παρακάτω πίνακα:

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

Οι αριθμοί που προσθέτουμε κάθε φορά (2^{-1}, 2^{-2}, 2^{-3} και 2^{-4}) είναι πολλαπλασιασμένοι με το 10000, προς διευκόλυνσή μας, ώστε να μετρήσουμε τις 1000άδες, τις 100άδες, τις 10άδες και τις μονάδες και να τις αποθηκεύουμε στις αντίστοιχες μεταβλητές. Με παρόμοιο τρόπο υπολογίζουμε και το ακέραιο μέρος, αφού πρώτα πραγματοποιήσουμε μία δεξιά ολίσθηση τεσσάρων θέσεων (δεν χρειαζόμαστε πλέον το δεκαδικό μέρος) και έπειτα μετράμε τις 100άδες, τις 10άδες και τις μονάδες και τις αποθηκεύουμε και αυτές στις αντίστοιχες μεταβλητές. Τέλος, εμφανίζουμε πρώτα το πρόσημο της θερμοκρασίας και έπειτα την τιμή της, με τη χρήση των εντολών της οθόνης LCD, η οποία είναι συνδεδεμένη μέσω του port expander.