ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣ/ΝΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 3

GROUP 3, ΟΜΑΔΑ 5

Παπαδόπουλος Κωνσταντίνος (8677)

Τοπαλίδης Ευθύμιος (8417)

Ζητούμενα

➔ Λειτουργία του προγράμματος (αλγόριθμος και συγκεκριμένα βήματα που απαιτούνται):
Η υλοποίηση του προγράμματος ακολούθησε τη σχεδιαστική πορεία που καθόριζε η εκφώνηση της άσκησης αυτή καθαυτή.
Σημεία που θα έπρεπε να τονίσουμε, εκτός των σχολίων που υπάρχουν στον πηγαίο κώδικα είναι:

- Η *rjmp RESET* είναι η πρώτη εντολή του κώδικα, ώστε ο PC να διαβάσει πράγματι εντολές και όχι άλλα τυχόν δεδομένα.

- Η αρχικοποίηση του stack pointer.

- Η αρχικοποίηση (μία φορά) των PORT εξόδου και εισόδου για τα LED και SWITCH αντίστοιχα.

- Ο έλεγχος πίεσης ενός πλήκτρου ή όχι γίνεται με την κλασική μεθοδολογία (σελ. 154 Προγραμματίζοντας τον Μικροελεγκτή AVR).

- Με την χρήση των εντολών sbrc/sbrs δεν χρειάστηκε να πάρουμε όλες τις δυνατές περιπτώσεις υγρασίας ξεχωριστά, αλλά αποθηκεύαμε για κάθε θερμοκρασία, χρόνο ποτίσματος τόσο, όσο μας υποδείκνυε ένα flag για την υγρασία (γινόταν 0b11111111 όταν το SW4 είχε πατηθεί).

- Δημιουργήσαμε μία κύρια υπο-ρουτίνα που καθόριζε τη σωστή έξοδο στα LEDs για κάθε χρονική στιγμή. Επίσης, δημιουργήσαμε υπο-ρουτίνες ελέγχου για το αν πατήθηκε το SW7 (αλλαγή προγράμματος άρδευσης) ή το SW5 (προσομοίωση χαμηλής τάσης) μειώνοντας και άλλο το μέγεθος του πηγαίου κώδικα.

- Η μορφή του αριθμού που αποθηκεύαμε σε κάθε πρόγραμμα και φορτώναμε στην έξοδο των LEDs είχε την εξής μορφή:
> +64 για να αλλάξει ο τρέχων # προγράμματος, +1 για να αλλάξει η ένδειξη του χρόνου ποτίσματος και yy = 00 ανάμεσα σε δύο διαδοχικά ποτίσματα, αλλιώς yy = 11 (είναι αρνητικής λογικής όλες οι έξοδοι) .
_0b11yy1111, 0b11yy1110, 0b11yy1101 ... για το πρώτο πρόγραμμα μέσα σε μια μέρα
_0b01yy1111, 0b01yy1110, 0b01yy1101 ... για το δεύτερο πρόγραμμα μέσα σε μια μέρα
_0b00yy1111, 0b00yy1110, 0b00yy1101 ... για το τρίτο πρόγραμμα μέσα σε μια μέρα

➔ Κώδικας με σχόλια και καθορισμός της ονομασίας των μεταβλητών που χρησιμοποιούνται:
Παρακάτω δίνεται το ζητούμενο πρόγραμμα με αναλυτικά σχόλια, στα σημεία που απαιτείται, για την κατανόηση λειτουργίας του κώδικα.

➔ Δυσκολίες που αντιμετωπίσαμε:
Η εύρεση ενός τρόπου αποτύπωσης στα LEDs, όλων των μεταβλητών που έπρεπε να προβάλλονται ταυτοχρόνως. Δηλαδή, την ίδια στιγμή να φαίνεται ο τρέχων αριθμός προγράμματος, καθώς και σε ποια χρονική στιγμή βρίσκεται το πότισμα.

➔ Διαφορές στην αποσφαλμάτωση στον προσομοιωτή και στην αναπτυξιακή κάρτα:
Στον προσομοιωτή δεν διαπιστώσαμε τη μη αποθήκευση ενός counter που μειωνόταν συνέχεια και δεν σταματούσε στο μηδέν. Επίσης, χρειάστηκε μια μικρή αλλαγή στην τοποθέτηση των flags που θέσαμε στο πρόγραμμα, καθώς αρχικά δεν είχαμε λάβει υπόψη με σωστό τρόπο την αρνητική λογική στα LEDs και στους διακόπτες.

➔ Επιπρόσθετα σχόλια για την παραπέρα βελτίωση του κώδικα που αναπτύχθηκε:
Στο τέλος της άσκησής μας διαπιστώσαμε ότι αντί να έχουμε έναν αριθμό αποθηκευμένο σε έναν καταχωρητή και να αλλάζουμε την τιμή αυτού για την αποτύπωση της εξόδου, μπορούσαμε να αποθηκεύουμε σε ξεχωριστούς καταχωρητές τον τρέχοντα αριθμό του προγράμματος και το χρόνο ποτίσματος. Βέβαια, με τη δική μας υλοποίηση πετύχαμε μικρότερο αριθμό εντολών σε σχέση με την προαναφερθείσα επίλυση.

```
;ergasia3.asm
;Papadopoulos Konstantinos, AEM 8677
;Topalidhs Efthymis, AEM 8417
;Group 3, Team 5
;Purpose: ...

.include "m16def.inc"


.cseg  ;tells the assembler that the following code is to be put into program memory
          ;This is necessary when the .dseg directive was used before

;.org $100    ;0x0100 ;set program memory address counter to 0x0100,
                    ;set the program counter to a specific value



rjmp RESET         ; ----first command of program

;---defining aliases for registers
.def temp = r16             ;the register for checking which temp mode has been selected
.def hum = r17              ;the register for checking which humidity mode has been
selected
.def start = r18       ;the register for checking if the program should start
.def lowbat = r19   ;checks if the battery is low or if SW7 is pressed
.def time = r20             ;counts program time
.def prcount = r21  ;counts current program number (programs left to execute)
.def between = r22;time between progs
.def numb = r23             ;used to display the different and variable LED value
(simultaneously)

RESET:                                        ; set initial value of stack pointer
        ldi r16, low(RAMEND)

        out SPL,r16

        ldi r16, HIGH(RAMEND)
        out SPH, r16
        rjmp main
```

```
main:

        rcall delay1calc    ;delay to block SW7 to be selected as temp
        rcall delay1calc    ;same
        rcall delay1calc    ;same

;initialization code
;---setting PORTB as output port
ldi r16, 0b11111111
out DDRB, r16               ;The data direction register of port B is named DDRB

;---setting PORTD as input port
ldi r16, 0b00000000
out DDRD, r16



;main program

ldi temp,255        ;works as a flag, if no SW is pressed then it is equal to 255
ldi hum, 0                  ;works as a flag, if SW4 is pressed then it is equal to 255

;--selecting temperature mode from SW0-SW3
get_switch_temp:
        in temp, PIND                   ;copy state of SW to PORTD
        cpi temp, 0b11111111
        breq get_switch_temp    ;if none is pressed keep waiting

;--selecting humidity mode from SW4 (if pressed -> humidity < 50%)
get_switch_hum:
        in start, PIND                  ;copy state of SW4 to PORTD

        sbrs start, 4               ;check if flag has been changed
        ldi hum, 255                    ;works as a flag, if SW4 is pressed then it is equal
to 255
        andi start,$F0                  ;clear lower nibble (0b11110000)
        cpi start, 0b10110000   ;checks if SW6 is pressed
        brne get_switch_hum             ;if SW6 has been pressed then continue
```

```
;--checking which temperature mode from SW0-SW3 has been selected
check_again:andi temp,$0F                ;clear upper nibble (0b00001111)
                    cpi temp, 0b00001110    ;checks if SW0 is pressed
                    breq SW_0
                    cpi temp, 0b00001101    ;checks if SW1 is pressed
                    breq SW_1
                    cpi temp, 0b00001011    ;checks if SW2 is pressed
                    breq SW_2
                    cpi temp, 0b00000111    ;checks if SW3 is pressed
                    breq SW_3
                    brne check_again



SW_0:
        ldi time,11          ;if SW4 not pressed 10+1=11
        sbrc hum,5
        ldi time,13          ;if SW4 is pressed 12+1=13
        ldi prcount, 3
        ldi between, 2
        ldi numb, 0b00111111
        rcall mainfunc

END0:       rjmp END0



SW_1:
        ldi time,5           ;if SW4 not pressed 4+1=5
        sbrc hum,5
        ldi time,7           ;if SW4 is pressed, 6+1=7
        ldi prcount,2
        ldi between, 3
        ldi numb, 0b01111111

        rcall mainfunc

        END1:       rjmp END1


SW_2:
```

```
        ldi time,2              ;if SW4 not pressed 1+1=2
        sbrc hum,1
        ldi time,4              ;if SW4 is pressed 3+1=4
        ldi prcount, 1
        ldi between, 6
        ldi numb, 0b10111111
        rcall mainfunc

END2:   rjmp END2

SW_3:
        ldi time,2              ;if SW4 not pressed 1+1=2
        sbrc hum,1
        ldi time,2              ;if SW4 is pressed 1+1=2
        ldi prcount, 1
        ldi between, 6
        ldi numb, 0b10111111
        rcall mainfunc

END3:   rjmp END3


lowbatcheck:
        in lowbat, PIND
        andi lowbat,$F0
        sbrs lowbat,5           ;check if SW5 is pressed
        rcall blink
ret


changemodecheck:
        in lowbat, PIND
        andi lowbat,$F0
        sbrs lowbat,7           ;check if SW7 is pressed
        rjmp main
ret



;--subroutine that actually displays the correct output
mainfunc:
        mov r0,time                     ;initializing saved variables
```

```
        mov r1,between
        rjmp A
C:                                  ;saving variables for every time a new
program starts
        mov r0, time
        mov r1, between
        add numb, time              ;resetting numb

A:                                  ;SW0-SW3 CURRENT TIME
        out PORTB, numb
        rcall lowbatcheck
        rcall changemodecheck
        rcall delay1calc

        dec numb        ;numb is the output that shows the correct LED sequence

        dec r0          ;time


        brne A


B:      ldi r16, 0b11001111     ;SW4-SW5 DELAY BETWEEN PROGRAMS
        out PORTB, r16

        rcall lowbatcheck
        rcall changemodecheck
        rcall delay1calc

        dec r1          ;time between
        brne B

        clc
        subi numb, -64  ;by adding 64 we have the output 00xxxxxx->prog3 shown
from LEDs [negative logic], 01xxxxxx->prog2,
                        ;10xxxxxx->prog1

        dec prcount     ;current program number (programs left to execute)
        brne C

ret
```

```
blink:
        ldi r27, 0                  ;open led lights for 1 sec
        out PORTB, r27
        rcall delay1calc
        ldi r27, 255          ;close led lights for 1 sec
        out PORTB, r27
        rcall delay1calc
        rjmp blink
ret



; delay subroutine
; Delay 3 999 997 cycles
; 999ms 999us 250 ns at 4.0 MHz -- approx 1 sec
;--as seen in the calculator
delay1calc:
                ldi  r24, 21
                ldi  r25, 75
                ldi  r26, 190
L1:             dec  r26
                brne L1
                dec  r25
                brne L1
                dec  r24
                brne L1
                nop

        ret
```