ΕΜΠ - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2021-2022

ΑΘΗΝΑ 21-11-2021

3η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

ΓΙΑ ΤΟ ΜΑΘΗΜΑ ''Εργαστήριο Μικροϋπολογιστών''

2η Εργ. Άσκ. στον Μικροελεγκτή AVR – Χρονιστές, Περιφερειακά (LCD, keyboard), ADC

(υλοποίηση στο εκπαιδευτικό σύστημα easyAVR6)

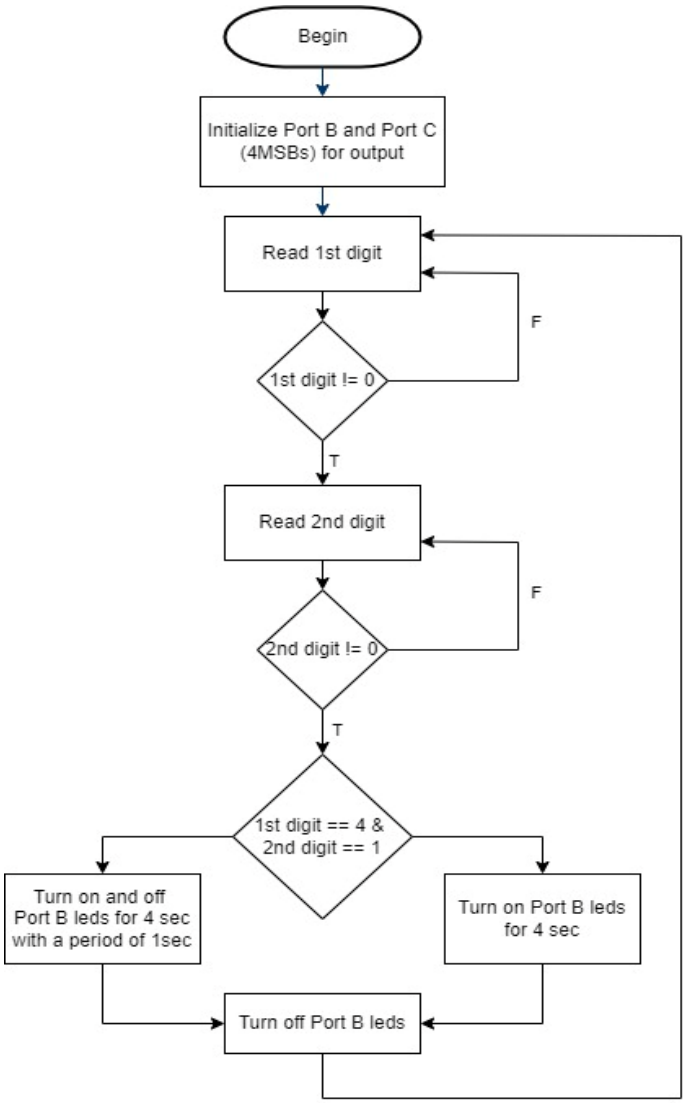
Ομάδα 41

Πεγειώτη Νάταλυ – 03117707

Ζήτημα 3.1

Αρχικά μεταφράστηκαν σε C οι συναρτήσεις scan_row_sim, scan_keypad_sim, scan_keypad_rising_edge_sim, keypad_to_ascii_sim. Στη συνέχεια δημιουργήθηκε το κύριο πρόγραμμα το οποίο διαβάζει δύο αριθμούς από το keyboard (διαβάζει κάθε φορά μέχρι να πατηθεί κάποιο πλήκτρο). Σε περίπτωση που ο διψήφιος αριθμός που σχηματίζεται είναι το 41 ανάβουν τα led της Port B για 4 δευτερόλεπτα, αλλιώς αναβοσβήνουν για 4 δευτερόλεπτα με περίοδο 1 δευτερολέπτου.

Το διάγραμμα ροής του ζητούμενου φαίνεται ακολούθως:



Ο κώδικας για την υλοποίηση του προγράμματος είναι ο ακόλουθος, ο οποίος επεξηγείται με αναλυτικά σχόλια:

```
#define F_CPU 8000000       // frequency is set 8MHz
#include <avr/io.h>
#include <util/delay.h>

int num_pressed, tmp;       // 16-bit number to store the key that was pressed and tmp
char first_digit, second_digit;    // digits pressed
char i, x, y;

char scan_row_sim(char y)   // y-->r25
{
       PORTC = y;    // Current line is set to 1
       _delay_us(500);      // delay for ~ 0.5usec (each 'nop' is 1/4usec
                            so it's included to 500usec)
       return PINC & 0x0F;  // keep the 4 LSB of PORTC
}

int scan_keypad_sim(void)   // x-->r24, y-->r25, z-->r26, h-->r27
{
       char z,h;     // set as parameters so as to be topical
       int result;   // result = r25:r24 to be returned

       y = 0x10;     // check 1st line
       x = scan_row_sim(y); // keep the result
       h =  x<<4;    // and save it in the 4 MSB of h

       y = 0x20;     // check 2nd line
       x = scan_row_sim(y); // keep the result
       h = h+x;      // and save it in the 4 LSB of h

       y = 0x40;     // check 3rd line
       x = scan_row_sim(y); // keep the result
       z =  x<<4;    // and save it in the 4 MSB of h

       y = 0x80;     // check 4th line
       x = scan_row_sim(y); // keep the result
       z = z+x;      // and save it in the 4 LSB of h

       result = h;
       return (result<<8) + z;     //return correct number
}

int scan_keypad_rising_edge_sim(void)
{
       int y = scan_keypad_sim();  // check the keypad for pressed button
       _delay_ms(15);       // delay for ~ 15msec
       int z = scan_keypad_sim();  // check the keypad again
       y = y & z;    // bitwise and, so to have the correct result
       z = tmp;      // load from RAM the previous value
       tmp = y;      // save in RAM the new value
       z = ~z;       // one's complement
       y = y & z;    // bitwise and
       return y;     // return value
}

char keypad_to_ascii_sim(int x)    // function that makes the number pressed
                                   to the ascii value or 0
{
       switch(x){
              case 0x01:return '*';
              case 0x02:return '0';
              case 0x04:return '#';
```

```
                case 0x08:return 'D';
                case 0x10:return '7';
                case 0x20:return '8';
                case 0x40:return '9';
                case 0x80:return 'C';
                case 0x100:return '4';
                case 0x200:return '5';
                case 0x400:return '6';
                case 0x800:return 'B';
                case 0x1000:return '1';
                case 0x2000:return '2';
                case 0x4000:return '3';
                case 0x8000:return 'A';
        }
        return 0;
}


int main(void)
{
        DDRB = 0xFF;  // initialize PB0-7 as output
        DDRC = (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4); // initialize PC4-7
                                                                    as output

        while(1)
        {
                first_digit = 0;     //initialize first digit with 0
                second_digit = 0;    //initialize first digit with 0
                do{
                        num_pressed = scan_keypad_rising_edge_sim();
                        first_digit = keypad_to_ascii_sim(num_pressed); // read and store
                                                                          1st digit
                } while(first_digit==0);    //repeat until first digit is valid
                do{
                        num_pressed = scan_keypad_rising_edge_sim();
                        second_digit = keypad_to_ascii_sim(num_pressed);// read and store
                                                                          2nd digit
                } while(second_digit==0);   //repeat until second digit is valid
                scan_keypad_rising_edge_sim();     // we call that for safety reasons


                if((first_digit=='4') && (second_digit)=='1')    // if the key is correct
                {
                        PORTB = 0xFF; // turn on LEDS in PORTB
                        _delay_ms(4000);     // for ~4sec
                }
                else  // if the key is wrong
                {
                        for(i=0; i<4; i++)   //repeat 4 times --> ~4sec
                        {
                                PORTB = 0xFF; // turn on LEDS
                                _delay_ms(500);      // for ~ 0.5sec
                                PORTB = 0x00; // turn off LEDS
                                _delay_ms(500);      // for ~ 0.5sec
                        }
                }
                PORTB = 0x00; // turn off leds
        }
}
```

Ζήτημα 3.2

Με τη χρήση των δοσμένων συναρτήσεων, υλοποιήθηκε η προηγούμενη άσκηση σε assembly με την προσθήκη της λειτουργίας της LCD display.
Για σωστό διψήφιο κωδικό εμφανίζει μήνυμα “WELCOME 41” ενώ για λανθασμένο εμφανίζει μήνυμα “ALARM ON”.

Ο κώδικας για την υλοποίηση του προγράμματος είναι ο ακόλουθος, ο οποίος επεξηγείται με αναλυτικά σχόλια:

```
.include "m16def.inc"

.DSEG
_tmp_: .byte 2
.CSEG
.def dgt1 = r20
.def dgt2 = r21

reset: ldi r24, low(RAMEND) ; initialize stack pointer
               out SPL, r24
               ldi r24, high(RAMEND)
               out SPH, r24

               ser r26
               out DDRB, r26 ; initialize PORTB for output
               out DDRD, r26 ; initialize PORTD that is connected to LCD, as output
               ldi r24, (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
               ; set as output the 4 MSB
               out DDRC, r24 ; of PORTC

main:  clr r24
       rcall lcd_init_sim   ; initialize with clear screen
       rcall scan_keypad_rising_edge_sim
       rcall keypad_to_ascii_sim   ; read 1st digit
       cpi r24, 0    ; repeat until 1st digit is valid
       breq main
       mov dgt1,r24  ; store 1st digit

next:  rcall scan_keypad_rising_edge_sim
       rcall keypad_to_ascii_sim   ; read 2nd digit
       cpi r24, 0    ; repeat until 2nd digit is valid
       breq next
       mov dgt2,r24  ; store 2nd digit
       rcall scan_keypad_rising_edge_sim  ; we call that for safety reasons
       cpi dgt1, 52  ; if 1st digit != 4
       brne wrong_key
       cpi dgt2, 49  ; or 2nd digit != 1
       brne wrong_key       ; wrong_key given, go to wrong_key

correct_key:  clr r24
              rcall lcd_init_sim
              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, '4'
                rcall lcd_data_sim
                ldi r24, '1'
                rcall lcd_data_sim   ;display "WELCOME 41"

                ser r19
                out PORTB, r19       ; turn on LEDS
                ldi r24, low(4000)
                ldi r25, high(4000)
                rcall wait_msec      ; delay 4sec
                clr r19
                out PORTB, r19       ; turn off LEDS
                rjmp main

wrong_key:      clr r24
                rcall lcd_init_sim
                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   ;display "ALARM ON"

        ldi r18, 4    ; loop for 4 times
loop:   cpi r18, 0
        breq finish
        ser r19
        out PORTB, r19       ; turn on LEDS
        ldi r24, low(500)
        ldi r25, high(500)
        rcall wait_msec      ; delay 0.5sec
        clr r19
        out PORTB, r19       ; turn off LEDS
        ldi r24, low(500)
        ldi r25, high(500)
        rcall wait_msec      ; delay 0.5sec
        dec r18
        rjmp loop

finish: rjmp main    ; return to main
```

*Για την σωστή λειτουργία του προγράμματος κλήθηκαν οι εξής δοσμένες συναρτήσεις (οι οποίες δεν προστέθηκαν στην αναφορά για λόγους συντομίας):
scan_row_sim, scan_keypad_sim, scan_keypad_rising_edge_sim, keypad_to_ascii_sim, return_ascii, lcd_init_sim, lcd_command_sim, lcd_data_sim, write_2_nibbles_sim, wait_msec, wait_usec