Technika Mikroprocesorowa

Sprawozdanie z Laboratorium 6, cz. 1

Maksym Pervov, grupa 4.7/13

1. Zadanie 1 i Zadanie 3
   Disassembly:

```
--- C:\Users\Maksym\OneDrive\���������\Techniki microprocesorowe\Lab6\Zadanie1_3\Zadanie1_3\Debug/.././main.c
    24: void clearPosition(uint8_t b){   //funckja komend
    25: LCD_PORT |= _BV(LCD_EN);//zezwalamy komunikajce z LCD
00000041 98.b3                IN R25,0x18              In from I/O location
00000042 98.60                ORI R25,0x08             Logical OR with immediate
00000043 98.bb                OUT 0x18,R25             Out to I/O location
    26: LCD_PORT = (b & 0xF0)|(LCD_PORT & 0x0F);//wysyłamy 4 starsze bity
00000044 98.b3                IN R25,0x18              In from I/O location
00000045 28.2f                MOV R18,R24              Copy register
00000046 20.7f                ANDI R18,0xF0            Logical AND with immediate
00000047 9f.70                ANDI R25,0x0F            Logical AND with immediate
00000048 92.2b                OR R25,R18       Logical OR
00000049 98.bb                OUT 0x18,R25             Out to I/O location
    27: LCD_PORT &= ~(_BV(LCD_EN));      //mówimy, że będziemy wysyłali dane
0000004A 98.b3                IN R25,0x18              In from I/O location
0000004B 97.7f                ANDI R25,0xF7            Logical AND with immediate
0000004C 98.bb                OUT 0x18,R25             Out to I/O location
    28: asm volatile("nop");             //jeden cykl mikroproc.
0000004D 00.00                NOP              No operation
    29: LCD_PORT |= _BV(LCD_EN);//zezwalamy komunikajce z LCD
0000004E 98.b3                IN R25,0x18              In from I/O location
0000004F 98.60                ORI R25,0x08             Logical OR with immediate
00000050 98.bb                OUT 0x18,R25             Out to I/O location
    30: LCD_PORT = ((b & 0x0F) << 4)|(LCD_PORT & 0x0F);//wysyłamy 4 młodsze bity
00000051 20.e1                LDI R18,0x10             Load immediate
00000052 82.9f                MUL R24,R18              Multiply unsigned
00000053 c0.01                MOVW R24,R0              Copy register pair
00000054 11.24                CLR R1           Clear Register
00000055 98.b3                IN R25,0x18              In from I/O location
00000056 9f.70                ANDI R25,0x0F            Logical AND with immediate
00000057 89.2b                OR R24,R25       Logical OR
00000058 88.bb                OUT 0x18,R24             Out to I/O location
    31: LCD_PORT &= ~(_BV(LCD_EN));      //mówimy, że będziemy wysyłali dane
00000059 88.b3                IN R24,0x18              In from I/O location
0000005A 87.7f                ANDI R24,0xF7            Logical AND with immediate
0000005B 88.bb                OUT 0x18,R24             Out to I/O location
--- c:\program files (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\util/delay.h
   187: __builtin_avr_delay_cycles(__ticks_dc);
0000005C 83.ed                LDI R24,0xD3             Load immediate
0000005D 90.e3                LDI R25,0x30             Load immediate
0000005E 01.97                SBIW R24,0x01            Subtract immediate from word
0000005F f1.f7                BRNE PC-0x01             Branch if not equal
00000060 00.c0                RJMP PC+0x0001           Relative jump
00000061 00.00                NOP              No operation
00000062 08.95                RET              Subroutine return
--- C:\Users\Maksym\OneDrive\���������\Techniki microprocesorowe\Lab6\Zadanie1_3\Zadanie1_3\Debug/.././main.c
    35: void clearLCD(){
    36: LCD_PORT &= ~(_BV(LCD_RS));
00000063 88.b3                IN R24,0x18              In from I/O location
00000064 8b.7f                ANDI R24,0xFB            Logical AND with immediate
00000065 88.bb                OUT 0x18,R24             Out to I/O location
    37: clearPosition(0x01);             //rejestr 0x01 - wyczyść tekst z LCD
00000066 81.e0                LDI R24,0x01             Load immediate
00000067 0e.94.41.00          CALL 0x00000041          Call subroutine
    38: LCD_PORT |= _BV(LCD_RS);//zapisujemy dane
00000069 88.b3                IN R24,0x18              In from I/O location
0000006A 84.60                ORI R24,0x04             Logical OR with immediate
0000006B 88.bb                OUT 0x18,R24             Out to I/O location
--- c:\program files (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\util/delay.h
   187: __builtin_avr_delay_cycles(__ticks_dc);
0000006C 87.e9                LDI R24,0x97             Load immediate
0000006D 9a.e3                LDI R25,0x3A             Load immediate
0000006E 01.97                SBIW R24,0x01            Subtract immediate from word
```

```
0000006F f1.f7                BRNE PC-0x01              Branch if not equal
00000070 00.c0                RJMP PC+0x0001            Relative jump
00000071 00.00                NOP               No operation
00000072 08.95                RET               Subroutine return
--- C:\Users\Maksym\OneDrive\���������\Techniki
microprocesorowe\Lab6\Zadanie1_3\Zadanie1_3\Debug/.././main.c
    42: void LCDinit(){
    43: LCD_DDR = (0xF0)|(_BV(LCD_RS))|(_BV(LCD_EN));
00000073 8c.ef                LDI R24,0xFC              Load immediate
00000074 87.bb                OUT 0x17,R24              Out to I/O location
    44: LCD_PORT = 0;   //wyzerowanie portu
00000075 18.ba                OUT 0x18,R1               Out to I/O location
    45: LCD_PORT &= ~(_BV(LCD_RS));     //dajemy komende
00000076 88.b3                IN R24,0x18               In from I/O location
00000077 8b.7f                ANDI R24,0xFB             Logical AND with immediate
00000078 88.bb                OUT 0x18,R24              Out to I/O location
    47: clearPosition(0b00101000);      //inicjalizuj 4-bitowy tryb + 2 linie 5*7 macierz
00000079 88.e2                LDI R24,0x28              Load immediate
0000007A 0e.94.41.00          CALL 0x00000041           Call subroutine
    48: LCD_PORT |= _BV(LCD_RS);
0000007C 88.b3                IN R24,0x18               In from I/O location
0000007D 84.60                ORI R24,0x04              Logical OR with immediate
0000007E 88.bb                OUT 0x18,R24              Out to I/O location
    49: LCD_PORT |= _BV(LCD_RS);
0000007F 88.b3                IN R24,0x18               In from I/O location
00000080 84.60                ORI R24,0x04              Logical OR with immediate
00000081 88.bb                OUT 0x18,R24              Out to I/O location
    51: clearPosition(0b00000110);      //inicjalizuj przesunięcie kursora w prawo
00000082 86.e0                LDI R24,0x06              Load immediate
00000083 0e.94.41.00          CALL 0x00000041           Call subroutine
    52: LCD_PORT |= _BV(LCD_RS);
00000085 88.b3                IN R24,0x18               In from I/O location
00000086 84.60                ORI R24,0x04              Logical OR with immediate
00000087 88.bb                OUT 0x18,R24              Out to I/O location
    53: LCD_PORT &= ~(_BV(LCD_RS));
00000088 88.b3                IN R24,0x18               In from I/O location
00000089 8b.7f                ANDI R24,0xFB             Logical AND with immediate
0000008A 88.bb                OUT 0x18,R24              Out to I/O location
    55: clearPosition(0b00001100);      //wyświetlaj przy wyłaczonym kursorze
0000008B 8c.e0                LDI R24,0x0C              Load immediate
0000008C 0e.94.41.00          CALL 0x00000041           Call subroutine
    56: LCD_PORT |= _BV(LCD_RS);//mówimy, że będziemy zapisywali dane
0000008E 88.b3                IN R24,0x18               In from I/O location
0000008F 84.60                ORI R24,0x04              Logical OR with immediate
00000090 88.bb                OUT 0x18,R24              Out to I/O location
    58: clearLCD();
00000091 0e.94.63.00          CALL 0x00000063           Call subroutine
00000093 08.95                RET               Subroutine return
    63: LCD_PORT &= ~(_BV(LCD_RS));     //mówimy, że podajemy komendę
00000094 98.b3                IN R25,0x18               In from I/O location
00000095 9b.7f                ANDI R25,0xFB             Logical AND with immediate
00000096 98.bb                OUT 0x18,R25              Out to I/O location
    64: clearPosition((w*0x40+h)|(0x80));//0x80 - 1 linia, 0x40 - tej 1/2 linii
00000097 90.e4                LDI R25,0x40              Load immediate
00000098 89.9f                MUL R24,R25               Multiply unsigned
00000099 60.0d                ADD R22,R0        Add without carry
0000009A 11.24                CLR R1            Clear Register
0000009B 86.2f                MOV R24,R22               Copy register
0000009C 80.68                ORI R24,0x80              Logical OR with immediate
0000009D 0e.94.41.00          CALL 0x00000041           Call subroutine
    65: LCD_PORT |= _BV(LCD_RS);//mówimy, że podajemy dane
0000009F 88.b3                IN R24,0x18               In from I/O location
000000A0 84.60                ORI R24,0x04              Logical OR with immediate
000000A1 88.bb                OUT 0x18,R24              Out to I/O location
--- c:\program files (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\util/delay.h
   187: __builtin_avr_delay_cycles(__ticks_dc);
000000A2 81.ee                LDI R24,0xE1              Load immediate
000000A3 94.e0                LDI R25,0x04              Load immediate
000000A4 01.97                SBIW R24,0x01             Subtract immediate from word
000000A5 f1.f7                BRNE PC-0x01              Branch if not equal
000000A6 00.c0                RJMP PC+0x0001            Relative jump
000000A7 00.00                NOP               No operation
000000A8 08.95                RET               Subroutine return
--- C:\Users\Maksym\OneDrive\���������\Techniki
microprocesorowe\Lab6\Zadanie1_3\Zadanie1_3\Debug/.././main.c
    69: void write(char *text, int8_t lenght){
000000A9 0f.93                PUSH R16          Push register on stack
```

```
    000000AA 1f.93                PUSH R17          Push register on stack
    000000AB cf.93                PUSH R28          Push register on stack
    000000AC df.93                PUSH R29          Push register on stack
    000000AD 8c.01                MOVW R16,R24              Copy register pair
    000000AE d6.2f                MOV R29,R22               Copy register
        71: setCursor(0,0);                         //ustawiamy kursor na początek
    000000AF 60.e0                LDI R22,0x00              Load immediate
    000000B0 80.e0                LDI R24,0x00              Load immediate
    000000B1 0e.94.94.00          CALL 0x00000094           Call subroutine
        70: int8_t i = 0;
    000000B3 c0.e0                LDI R28,0x00              Load immediate
        72: while (i < lenght){
    000000B4 0f.c0                RJMP PC+0x0010            Relative jump
        73:         clearPosition(text[i]); //czyścimy na pozycji i
    000000B5 f8.01                MOVW R30,R16              Copy register pair
    000000B6 ec.0f                ADD R30,R28               Add without carry
    000000B7 f1.1d                ADC R31,R1        Add with carry
    000000B8 c7.fd                SBRC R28,7        Skip if bit in register cleared
    000000B9 fa.95                DEC R31           Decrement
    000000BA 80.81                LDD R24,Z+0               Load indirect with displacement
    --- C:\Users\Maksym\OneDrive\���������\Techniki
    microprocesorowe\Lab6\Zadanie1_3\Zadanie1_3\Debug/.././main.c
    000000BB 0e.94.41.00          CALL 0x00000041           Call subroutine
        74:         if(i==16){                              //jezeli tekst wiekszy od 16
    000000BD c0.31                CPI R28,0x10              Compare with immediate
    000000BE 21.f4                BRNE PC+0x05              Branch if not equal
        75:                 setCursor(1,0);         //ustawiamy na poczatek pierwszej linijki
    000000BF 60.e0                LDI R22,0x00              Load immediate
    000000C0 81.e0                LDI R24,0x01              Load immediate
    000000C1 0e.94.94.00          CALL 0x00000094           Call subroutine
        77:         i++;
    000000C3 cf.5f                SUBI R28,0xFF             Subtract immediate
        72: while (i < lenght){
    000000C4 cd.17                CP R28,R29        Compare
    000000C5 7c.f3                BRLT PC-0x10              Branch if less than, signed
        79: }
    000000C6 df.91                POP R29           Pop register from stack
    000000C7 cf.91                POP R28           Pop register from stack
    000000C8 1f.91                POP R17           Pop register from stack
    000000C9 0f.91                POP R16           Pop register from stack
    000000CA 08.95                RET               Subroutine return
        90: void LCDclear_y(char number, char lenght) {
    000000CB cf.93                PUSH R28          Push register on stack
    000000CC df.93                PUSH R29          Push register on stack
    000000CD c8.2f                MOV R28,R24               Copy register
    000000CE d6.2f                MOV R29,R22               Copy register
        91: setCursor(number,number-1);             //ustawiam kursor na linijke oraz wiersz
    000000CF 6f.ef                SER R22           Set Register
    000000D0 68.0f                ADD R22,R24               Add without carry
    000000D1 0e.94.94.00          CALL 0x00000094           Call subroutine
        92: for(char i = number; i < lenght; i++)
    000000D3 0e.c0                RJMP PC+0x000F            Relative jump
        94:         if(i > 16 ) {
    000000D4 c1.31                CPI R28,0x11              Compare with immediate
    000000D5 40.f0                BRCS PC+0x09              Branch if carry set
        95:                 setCursor(1, 0);        //przemieścienie na nową linijkę
    000000D6 60.e0                LDI R22,0x00              Load immediate
    000000D7 81.e0                LDI R24,0x01              Load immediate
    000000D8 0e.94.94.00          CALL 0x00000094           Call subroutine
        96:                 clearPosition(0b00010100);//przemieścienie kursora w prawo
    000000DA 84.e1                LDI R24,0x14              Load immediate
    000000DB 0e.94.41.00          CALL 0x00000041           Call subroutine
    000000DD 03.c0                RJMP PC+0x0004            Relative jump
        98:         else{clearPosition(0b00010100);}
    000000DE 84.e1                LDI R24,0x14              Load immediate
    000000DF 0e.94.41.00          CALL 0x00000041           Call subroutine
        92: for(char i = number; i < lenght; i++)
000000E1 cf.5f                SUBI R28,0xFF             Subtract immediate
```

Source code:

```
        /*
     * Zadanie1_3.c
     *
     * Created: 02.06.2022 12:26:01
     * Author : Maksym Pervov
     */
```

```c
#include <avr/io.h>
#include <util/delay.h>
#include <string.h>

#define F_CPU 1000000L

#define LCD_DDR DDRB
#define LCD_PORT PORTB
#define LCD_RS 2
#define LCD_EN 3
#define LCD_DB4 4
#define LCD_DB5 5
#define LCD_DB6 6
#define LCD_DB7 7

void clearPosition(uint8_t b){    //funckja komend
       LCD_PORT |= _BV(LCD_EN);   //zezwalamy komunikajce z LCD
       LCD_PORT = (b & 0xF0)|(LCD_PORT & 0x0F);//wysyłamy 4 starsze bity
       LCD_PORT &= ~(_BV(LCD_EN));//mówimy, że będziemy wysyłali dane
       asm volatile("nop");       //jeden cykl mikroproc.
       LCD_PORT |= _BV(LCD_EN);   //zezwalamy komunikajce z LCD
       LCD_PORT = ((b & 0x0F) << 4)|(LCD_PORT & 0x0F);//wysyłamy 4 młodsze bity
       LCD_PORT &= ~(_BV(LCD_EN));//mówimy, że będziemy wysyłali dane
       _delay_ms(50);
}

void clearLCD(){
       LCD_PORT &= ~(_BV(LCD_RS));
       clearPosition(0x01);       //rejestr 0x01 - wyczyść tekst z LCD
       LCD_PORT |= _BV(LCD_RS);   //zapisujemy dane
       _delay_ms(60);
}

void LCDinit(){
       LCD_DDR = (0xF0)|(_BV(LCD_RS))|(_BV(LCD_EN));
       LCD_PORT = 0; //wyzerowanie portu
       LCD_PORT &= ~(_BV(LCD_RS));//dajemy komende

       clearPosition(0b00101000); //inicjalizuj 4-bitowy tryb + 2 linie 5*7 macierz
       LCD_PORT |= _BV(LCD_RS);
       LCD_PORT |= _BV(LCD_RS);

       clearPosition(0b00000110); //inicjalizuj przesunięcie kursora w prawo
       LCD_PORT |= _BV(LCD_RS);
       LCD_PORT &= ~(_BV(LCD_RS));

       clearPosition(0b00001100); //wyświetlaj przy wyłaczonym kursorze
       LCD_PORT |= _BV(LCD_RS);   //mówimy, że będziemy zapisywali dane

       clearLCD();
}

void setCursor(unsigned char w, unsigned char h)
{
       LCD_PORT &= ~(_BV(LCD_RS));//mówimy, że podajemy komendę
       clearPosition((w*0x40+h)|(0x80));//0x80 - 1 linia, 0x40 - tej 1/2 linii
       LCD_PORT |= _BV(LCD_RS);   //mówimy, że podajemy dane
       _delay_ms(5);
}

void write(char *text, int8_t lenght){
       int8_t i = 0;
       setCursor(0,0);                         //ustawiamy kursor na początek
       while (i < lenght){
              clearPosition(text[i]); //czyścimy na pozycji i
              if(i==16){                         //jezeli tekst wiekszy od 16
                     setCursor(1,0);             //ustawiamy na poczatek pierwszej linijki
              }
              i++;
       }
}

/*void LCDclear_y(unsigned char n, unsigned char lenght){
       setCursor(n, lenght);
       while (n<=16)
       {
              clearPosition(' ');
              n++;
       }
}*/

void LCDclear_y(char number, char lenght) {
       setCursor(number,number-1);        //ustawiam kursor na linijke oraz wiersz
       for(char i = number; i < lenght; i++)
       {
              if(i > 16 ) {
                     setCursor(1, 0);            //przemieszczenie na nową linijkę
                     clearPosition(0b00010100);//przemieszczenie kursora w prawo
              }
              else{clearPosition(0b00010100);}
       }
}

void zadanie1()
{
       LCDinit();
       char text[] = "Hello World     !!! I love you";
       write(text,30);                           //wypisanie tekstu
       LCDclear_y(0,11);                  //oczyścienie LCD z pewnej pozycji i dlugością
}

void zadanie3()
{
       char symbol1[] = {'ć','ż','ź','ą','ę','ł','ś','ó','ń','€'};
       char symbol2[] = {'U','b','e','r','k','o','t'};
       char symbol3[] = {'T','h','u','r','s','d','a','y'};
       LCDinit();
       while(1)
       {
              write(symbol1, 10);         //wypisanie tekstu
              _delay_ms(500);                    //opóżnienie co 0,5 s
              clearLCD();                        //oczyśccienie LCD
              write(symbol2, 7);          //wypisanie tekstu
              _delay_ms(500);                    //opóżnienie co 0,5 s
              clearLCD();                        //oczyśccienie LCD
              write(symbol3, 8);          //wypisanie tekstu
              _delay_ms(500);                    //opóżnienie co 0,5 s
              clearLCD();                        //oczyśccienie LCD
       }
}

int main(void)
{
       zadanie1();                        //funkcja zadania 1
       //zadanie3();                      //Funkcja zadania 3
    while (1){}
}
```