Instytut Informatyki Politechniki Śląskiej
Zespół Mikroinformatyki i Teorii Automatów Cyfrowych

Laboratorium SMiW

Rok akademicki	Rodzaj studiów*: SSI/NSI/NSM	Numer ćwiczenia:	Grupa	Sekcja
2013/2014	**SSI**	**20 i 21**	**5**	**1**

Data i godzina planowana ćwiczenia: dd/mm/rrrr - gg:mm	**2013-11-25 – 13:15**	**Prowadzący**: OA/JP/KT/GD/BSz/GB	**JP**
Data i godzina wykonania ćwiczenia: dd/mm/rrrr - gg:mm	**2013-11-25 – 13:15**		

Sprawozdanie

Temat ćwiczenia:

Mikrokontrolery z serii AVR cz. 2

Skład sekcji:	
	1. Mikołaj Kloszczyk 2. Daniel Rygol 3. Dawid Jasiok 4. Miłosz Świątek 5. Mateusz Sikora 6. 7. 8. 9. 10. 11. 12. 13. 14. 15.

1. Opis zadania

Do Atmegi 128 podłączono 8 diod do portu B i 8 przycisków do portu D. Zaprogramuj Atmege 128 tak aby po naciśnięciu przycisku nr X zapali się dioda nr X. Po puszczeniu przycisku dioda będzie się świeciła jeszcze przez 1s a potem zgaśnie.

2. Kod programu w języku C

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#define F_CPU 8000000UL // 8 MHz
volatile unsigned int licznik0= 0;
volatile unsigned int licznik1 = 0;
volatile unsigned int licznik2 = 0;
volatile unsigned int licznik3 = 0;
volatile unsigned int licznik4 = 0;
volatile unsigned int licznik5 = 0;
volatile unsigned int licznik6 = 0;
volatile unsigned int licznik7 = 0;

int main (void)
{
	DDRD = 0x00; //ustawienie portów jako wejścia dla przycisków
	PORTD = 0xFF;
	TCCR0 = 6;    //częstotliwośc zegara/ 256 czyli (8 000 000Hz / 256)/256 = 122,0703125
razy na sekundę czyli co 8ms
	TIMSK |= (1<<TOIE0);
	DDRB = 0xFF; //ustawienie portów jako wyjścia dla diód
	sei(); // włącz globalną obsługę przerwań
	while(1)
	{
		if (!(PIND & (1<<PD0)))
		{
			PORTB |= (1<<PB0);
			licznik0= 0;
		}
		if (!(PIND & (1<<PD1)))
		{
			PORTB |= (1<<PB1);
			licznik1= 0;
		}
		if (!(PIND & (1<<PD2)))
		{
			PORTB |= (1<<PB2);
			licznik2= 0;
		}
		if (!(PIND & (1<<PD3)))
		{
			PORTB |= (1<<PB3);
			licznik3= 0;
		}
		if (!(PIND & (1<<PD4)))
		{
			PORTB |= (1<<PB4);
			licznik4= 0;
		}
		if (!(PIND & (1<<PD5)))
		{
			PORTB |= (1<<PB5);
			licznik5= 0;
		}
		if (!(PIND & (1<<PD6)))
		{
			PORTB |= (1<<PB6);
			licznik6= 0;
		}
```

```
		if (!(PIND & (1<<PD7)))
		{
			PORTB |= (1<<PB7);
			licznik7= 0;
		}
	}
}

ISR(TIMER0_OVF_vect)
{
	if(licznik0 > 122)
	{
		PORTB &= 0<<1; //zgaś diode nr 1
		licznik0 = 0;
	}
	if(licznik1 > 122)
	{
		PORTB &= 0<<2; //zgaś diode nr 2
		licznik1 = 0;
	}
	if(licznik2 > 122)
	{
		PORTB &= 0<<3; //zgaś diode nr 3
		licznik2 = 0;
	}
	if(licznik3 > 122)
	{
		PORTB &= 0<<4; //zgaś diode nr 4
		licznik3 = 0;
	}

	if(licznik4 > 122)
	{
		PORTB &= 0<<5; //zgaś diode nr 5
		licznik4 = 0;
	}
	if(licznik5 > 122)
	{
		PORTB &= 0<<6; //zgaś diode nr 6
		licznik5 = 0;
	}
	if(licznik6 > 122)
	{
		PORTB &= 0<<7; //zgaś diode nr 7
		licznik6 = 0;
	}
	if(licznik7 > 122)
	{
		PORTB &= 0<<8; //zgaś diode nr 8
		licznik7 = 0;
	}
	licznik0++;
	licznik1++;
	licznik2++;
	licznik3++;
	licznik4++;
	licznik5++;
	licznik6++;
	licznik7++;
}
```

3. Kod programu w języku assemblerowym

```
;/////////////////////////////////////////////////////////////////////////////
; Laboratory AVR Microcontrollers Part1
; Program template for lab 20
; Authors:
;
; Group:
; Section:
;
; Task:
;
; Todo:
;
;
; Version: 1.0
;/////////////////////////////////////////////////////////////////////////////
.nolist ;quartz assumption 4Mhz
.include "m128def.inc"
.list

;/////////////////////////////////////////////////////////////////////////////
; EEPROM - data non volatile memory segment
.ESEG

;/////////////////////////////////////////////////////////////////////////////
; StaticRAM - data memory.segment
.DSEG

.ORG 0x100; may be omitted this is default value
; Destination table (xlengthx bytes).
; Replace "xlengthx" with correct value
RAMTAB: .BYTE 256

;/////////////////////////////////////////////////////////////////////////////
; CODE - Program memory segment
; Please Remember that it is "word" address space
;
.CSEG
.org 0x0000 ; may be omitted this is default value
jmp    RESET  ; Reset Handler

; Interrupts vector table / change to your procedure only when needed
jmp    EXT_INT0      ; IRQ0 Handler
jmp    EXT_INT1      ; IRQ1 Handler
jmp    EXT_INT2      ; IRQ2 Handler
jmp    EXT_INT3      ; IRQ3 Handler
jmp    EXT_INT4      ; IRQ4 Handler
jmp    EXT_INT5      ; IRQ5 Handler
jmp    EXT_INT6      ; IRQ6 Handler
jmp    EXT_INT7      ; IRQ7 Handler
jmp    TIM2_COMP     ; Timer2 Compare Handler
jmp    TIM2_OVF      ;Timer2 Overflow Handler
jmp    TIM1_CAPT     ;Timer1 Capture Handler
jmp    TIM1_C0MPA    ;Timer1 CompareA Handler
jmp    TIM1_C0MPB    ;Timer1 CompareB Handler
jmp    TIM1_0VF      ;Timer1 Overflow Handler
jmp    TIM0_COMP     ;Timer0 Compare Handler
jmp    TIM0_OVF      ;Timer0 Overflow Handler
jmp    SPI_STC       ;SPI Transfer Complete Handler
jmp    USART0_RXC    ;USART0 RX Complete Handler
jmp    USART0_DRE    ;USART0,UDR Empty Handler
jmp    USART0_TXC    ;USART0 TX Complete Handler
jmp    ADC1          ;ADC Conversion Complete Handler
jmp    EE_RDY        ;EEPROM Ready Handler
jmp    ANA_COMP      ;Analog Comparator Handler
jmp    TIM1_C0MPC    ;Timer1 CompareC Handler
```

```
jmp    TIM3_CAPT     ;Timer3 Capture Handler
jmp    TIM3_COMPA    ;Timer3 CompareA Handler
jmp    TIM3_COMPB    ; Timer3 CompareB Handler
jmp    TIM3_COMPC    ;Timer3 CompareC Handler
jmp    TIM3_OVF      ;Timer3 Overflow Handler
jmp    USART1_RXC    ;USART1 RX Complete Handler
jmp    USART1_DRE    ;USART1,UDR Empty Handler
jmp    USART1_TXC    ;USART1 TX Complete Handler
jmp    TWI           ;Two-wire Serial Interface Interrupt Handler
jmp    SPM_RDY       ;SPM Ready Handler

;////////////////////////////////////////////////////////////////////////////
EXT_INT0:     ; IRQ0 Handler
EXT_INT1:     ; IRQ1 Handler
EXT_INT2:     ; IRQ2 Handler
EXT_INT3:     ; IRQ3 Handler
EXT_INT4:     ; IRQ4 Handler
EXT_INT5:     ; IRQ5 Handler
EXT_INT6:     ; IRQ6 Handler
EXT_INT7:     ; IRQ7 Handler
TIM2_COMP:    ; Timer2 Compare Handler
TIM2_OVF:   ;Timer2 Overflow Handler
TIM1_CAPT:  ;Timer1 Capture Handler
TIM1_C0MPA: ;Timer1 CompareA Handler
TIM1_C0MPB: ;Timer1 CompareB Handler
TIM1_0VF:   ;Timer1 Overflow Handler
TIM0_COMP:  ;Timer0 Compare Handler
TIM0_OVF:   ;Timer0 Overflow Handler
       cpi r17, 0x7A
       breq ZgasLED1
PorownajLicznik2:
       cpi r18, 0x7A
       breq ZgasLED2
PorownajLicznik3:
       cpi r19, 0x7A
       breq ZgasLED3
PorownajLicznik4:
       cpi r20, 0x7A
       breq ZgasLED4
PorownajLicznik5:
       cpi r21, 0x7A
       breq ZgasLED5
PorownajLicznik6:
       cpi r22, 0x7A
       breq ZgasLED6
PorownajLicznik7:
       cpi r23, 0x7A
       breq ZgasLED7
PorownajLicznik8:
       cpi r24, 0x7A
       breq ZgasLED8
PorownajLicznik9:
       inc r17       ;inkrementacja liczników
       inc r18
       inc r19
       inc r20
       inc r21
       inc r22
       inc r23
       inc r24
reti

ZgasLED1:
ldi r17, 0x00
cbi PORTB,0          ;zgaś diode nr 0
rjmp PorownajLicznik2

ZgasLED2:
ldi r18, 0x00
```

```
cbi PORTB,1          ;zgaś diode nr 1
rjmp PorownajLicznik3

ZgasLED3:
ldi r19, 0x00
cbi PORTB,2          ;zgaś diode nr 2
rjmp PorownajLicznik4

ZgasLED4:
ldi r20, 0x00
cbi PORTB,3          ;zgaś diode nr 3
rjmp PorownajLicznik5

ZgasLED5:
ldi r21, 0x00
cbi PORTB,4          ;zgaś diode nr 4
rjmp PorownajLicznik6

ZgasLED6:
ldi r22, 0x00
cbi PORTB,5          ;zgaś diode nr 5
rjmp PorownajLicznik7

ZgasLED7:
ldi r23, 0x00
cbi PORTB,6          ;zgaś diode nr 6
rjmp PorownajLicznik8

ZgasLED8:
ldi r24, 0x00
cbi PORTB,7          ;zgaś diode nr 7
rjmp PorownajLicznik9

SPI_STC:    ;SPI Transfer Complete Handler
USART0_RXC: ;USART0 RX Complete Handler
USART0_DRE: ;USART0,UDR Empty Handler
USART0_TXC: ;USART0 TX Complete Handler
ADC1:       ;ADC Conversion Complete Handler
EE_RDY:     ;EEPROM Ready Handler
ANA_COMP:   ;Analog Comparator Handler
TIM1_C0MPC: ;Timer1 CompareC Handler
TIM3_CAPT:  ;Timer3 Capture Handler
TIM3_COMPA: ;Timer3 CompareA Handler
TIM3_COMPB: ; Timer3 CompareB Handler
TIM3_COMPC: ;Timer3 CompareC Handler
TIM3_OVF:   ;Timer3 Overflow Handler
USART1_RXC: ;USART1 RX Complete Handler
USART1_DRE: ;USART1,UDR Empty Handler
USART1_TXC: ;USART1 TX Complete Handler
TWI:        ;Two-wire Serial Interface Interrupt Handler
SPM_RDY:    ;SPM Ready Handler
reti          ; return from all no used

;////////////////////////////////////////////////////////////////////////////
; Program start
RESET:

cli           ; disable all interrupts
;  Set stack pointer to top of RAM
ldi R16, HIGH(RAMEND)
out SPH, R16
ldi R16, LOW(RAMEND)
out SPL, R16

;----------------------------------------------------------------------------
; Main program code place here
; 1. Place here code related to initialization of ports and interrupts
ldi r16, 0x00
out DDRD,r16
```

```
ldi r16, 0xff
out PORTD, r16
ldi r16,0xFF  ;
out PORTB,r16 ; PORTB-jako wyjscie
out DDRB,r16  ; PORTB-wyjscie w stanie wysokim
ldi r16, 0x06
out TCCR0, r16
ldi r16, 0x01
out TIMSK, r16


sei
main:
	sbic PORTD, 0
	rjmp if_1
if_2Main:
	sbic PORTD, 1
	rjmp if_2
if_3Main:
	sbic PORTD, 2
	rjmp if_3
if_4Main:
	sbic PORTD, 3
	rjmp if_4
if_5Main:
	sbic PORTD, 4
	rjmp if_5
if_6Main:
	sbic PORTD, 5
	rjmp if_6
if_7Main:
	sbic PORTD, 6
	rjmp if_7
if_8Main:
	sbic PORTD, 7
	rjmp if_8

rjmp main

if_1:
sbi PORTB,0   ;zapalenie diody 0
ldi r17, 0x00
rjmp if_2Main

if_2:
sbi PORTB,1 ;zapalenie diody 1
ldi r18, 0x00
rjmp if_3Main

if_3:
sbi PORTB,2 ;zapalenie diody 2
ldi r19, 0x00
rjmp if_4Main

if_4:
sbi PORTB,3 ;zapalenie diody 3
ldi r20, 0x00
rjmp if_5Main

if_5:
sbi PORTB,4 ;zapalenie diody 4
ldi r21, 0x00
rjmp if_6Main

if_6:
sbi PORTB,5 ;zapalenie diody 5
ldi r22, 0x00
rjmp if_7Main
```

```
if_7:
sbi PORTB,6 ;zapalenie diody 6
ldi r23, 0x00
rjmp if_8Main

if_8:
sbi PORTB,7 ;zapalenie diody 7
ldi r24, 0x00
rjmp main

;---------------------------------------------------------------------------
; Table Declaration -  place here test values
; Test with different table values and different begin addresses of table (als above 0x8000)
;
ROMTAB: .db 0xff
.EXIT
;---------------------------------------------------------------------------
```

4. Podsumowanie

Wielkość pliku *.hex z języka C	1 969 bajty
Wielkość pliku *.hex z języka assemblerowego	1 045 bajty

Pomimo tego że obydwa programy działają tak samo pod względem wykorzystanych algorytmów działania to plik wynikowy z języka assemblerowego jest prawie dwa razy mniejszy. Jednak programowanie w języku niskiego poziomu wymaga od programisty poświęcenia większego czasu i większych umiejętności niż napisanie tego samego w języku wysokiego poziomu.