**Technika Mikroprocesorowa**

**Sprawozdanie z Laboratorium 5**

**Maksym Pervov, grupa 4.7/13**

1. Zadanie 1

Disassembly:

--- C:\Users\Maksym\OneDrive\���������\Techniki microprocesorowe\Lab5\Zadanie1\Zadanie1\Debug/.././main.c

11: ISR(INT0\_vect){

00000036 1f.92 PUSH R1 Push register on stack

00000037 0f.92 PUSH R0 Push register on stack

00000038 0f.b6 IN R0,0x3F In from I/O location

00000039 0f.92 PUSH R0 Push register on stack

0000003A 11.24 CLR R1 Clear Register

0000003B 8f.93 PUSH R24 Push register on stack

0000003C 9f.93 PUSH R25 Push register on stack

12: PORTA ^= 1 << PA0;

0000003D 9b.b3 IN R25,0x1B In from I/O location

0000003E 81.e0 LDI R24,0x01 Load immediate

0000003F 89.27 EOR R24,R25 Exclusive OR

00000040 8b.bb OUT 0x1B,R24 Out to I/O location

13: }

00000041 9f.91 POP R25 Pop register from stack

00000042 8f.91 POP R24 Pop register from stack

00000043 0f.90 POP R0 Pop register from stack

00000044 0f.be OUT 0x3F,R0 Out to I/O location

00000045 0f.90 POP R0 Pop register from stack

00000046 1f.90 POP R1 Pop register from stack

00000047 18.95 RETI Interrupt return

15: ISR(INT2\_vect){

00000048 1f.92 PUSH R1 Push register on stack

00000049 0f.92 PUSH R0 Push register on stack

0000004A 0f.b6 IN R0,0x3F In from I/O location

0000004B 0f.92 PUSH R0 Push register on stack

0000004C 11.24 CLR R1 Clear Register

0000004D 8f.93 PUSH R24 Push register on stack

0000004E 9f.93 PUSH R25 Push register on stack

16: PORTA ^= 1 << PA2;

0000004F 9b.b3 IN R25,0x1B In from I/O location

00000050 84.e0 LDI R24,0x04 Load immediate

00000051 89.27 EOR R24,R25 Exclusive OR

00000052 8b.bb OUT 0x1B,R24 Out to I/O location

17: }

00000053 9f.91 POP R25 Pop register from stack

00000054 8f.91 POP R24 Pop register from stack

00000055 0f.90 POP R0 Pop register from stack

00000056 0f.be OUT 0x3F,R0 Out to I/O location

00000057 0f.90 POP R0 Pop register from stack

00000058 1f.90 POP R1 Pop register from stack

00000059 18.95 RETI Interrupt return

22: DDRA = 1 << PA0 | 1 << PA2;

0000005A 85.e0 LDI R24,0x05 Load immediate

0000005B 8a.bb OUT 0x1A,R24 Out to I/O location

24: DDRB = 0xFF;

0000005C 8f.ef SER R24 Set Register

0000005D 87.bb OUT 0x17,R24 Out to I/O location

25: PORTB = 0xFF;

0000005E 88.bb OUT 0x18,R24 Out to I/O location

27: DDRD = 0xFF;

0000005F 81.bb OUT 0x11,R24 Out to I/O location

28: PORTD = 0xFF;

00000060 82.bb OUT 0x12,R24 Out to I/O location

30: tmp = MCUCR;

00000061 85.b7 IN R24,0x35 In from I/O location

33: MCUCR = tmp;

00000062 85.bf OUT 0x35,R24 Out to I/O location

35: MCUCSR &= ~(1 << ISC2);

00000063 84.b7 IN R24,0x34 In from I/O location

00000064 8f.7b ANDI R24,0xBF Logical AND with immediate

00000065 84.bf OUT 0x34,R24 Out to I/O location

38: GICR |= 1 << INT0 | 1 << INT2;

00000066 8b.b7 IN R24,0x3B In from I/O location

00000067 80.66 ORI R24,0x60 Logical OR with immediate

00000068 8b.bf OUT 0x3B,R24 Out to I/O location

39: GIFR |= 1 << INTF0 | 1 << INTF2;

00000069 8a.b7 IN R24,0x3A In from I/O location

0000006A 80.66 ORI R24,0x60 Logical OR with immediate

0000006B 8a.bf OUT 0x3A,R24 Out to I/O location

40: sei();

0000006C 78.94 SEI Global Interrupt Enable

0000006D ff.cf RJMP PC-0x0000 Relative jump

Source code:

/\*

\* Zadanie1.c

\* Created: 26.05.2022 12:39:04

\* Author : Maksym Pervov

\*/

#include <avr/io.h>

#include <avr/interrupt.h>

ISR(INT0\_vect){ //definiowanie obsługi przerwań INT0

PORTA ^= 1 << PA0; //negowanie diody na linii 0

}

ISR(INT2\_vect){ //definiowanie obsługi przerwań INT2

PORTA ^= 1 << PA2; //negowanie diody na linii 2

}

int main(void)

{

*uint8\_t* tmp; //definiowanie przemiennej tmp

DDRA = 1 << PA0 | 1 << PA2; //PA0 i PA2 na wyjście

DDRB = 0xFF; //obsługiwanie PB2 dla INT2

PORTB = 0xFF;

DDRD = 0xFF; //obsługiwanie PD2 dla INT0

PORTD = 0xFF;

tmp = MCUCR; //zapisywanie stanu rejestru do pamięci

tmp &= ~(1 << ISC00) | (1 << ISC00); //konfiguracja przerwania INT0 dla zbocza

opadającego

//tmp |= 1 << ISC01 | 1 << ISC00; //konfiguracja przerwania INT0 dla zbocza

narastającego

MCUCR = tmp;

MCUCSR &= ~(1 << ISC2); //konfiguracja przerwania INT2 dla zbocza

opadającego

//MCUCSR |= 1 << ISC2; //konfiguracja przerwania INT2 dla zbocza

narastającego

GICR |= 1 << INT0 | 1 << INT2; //zezwolenie na przerwania dla PB2 i PD2

GIFR |= 1 << INTF0 | 1 << INTF2; //ustawianie flagów w przypadku

zewnętrznego przerwania

sei(); //inicjalizacja obsługi przerwań

while (1){}}

1. Zadanie 2 (wykonane częściowo)

Source code:

/\*

\* Zadanie2.c

\*

\* Created: 26.05.2022 13:46:57

\* Author : Maksym Pervov

\*/

#include <avr/io.h>

#include <avr/interrupt.h>

volatile int n;

ISR(TIMER0\_COMP\_vect){

if(n == 10){

PORTA ^= 1 << PA0;

n = 0;

}

PORTA ^= 1 << PA2;

n++;

}

int main(void)

{

DDRA = 0xFF;

TCCR0 |= (1 << WGM01)|(1 << CS02)|(1 << CS00);

TIMSK |= (1 << OCIE0);

OCR0 = 97;

GICR |= 1 << INT0 | 1 << INT2;

GIFR |= 1 << INTF0 | 1 << INTF2;

sei();

while (1){}

}