Technika Mikroprocesorowa

Sprawozdanie z Laboratorium 4

Odróbiono 07.06.2022 o, mgr inż. Patryk Panas

Maksym Pervov, grupa 4.7/13

1. Zadanie 1

   Disassembly:

```
--- C:\Users\Maksym\OneDrive\##########\Techniki microprocesorowe\Lab4\Zadanie1\Zadanie1\Debug/.././main.c
    12: {
    13:     uint8_t x, sk;      //inicjalizacja zmiennych x i sk
    14:     uint8_t dt = 0;     //zmienna sluzy do obslugi nacisniec wiecej niz jeden przycisk
    15:     for (uint8_t i = 4; i < 8; i++)
00000036 34.e0                LDI R19,0x04      Load immediate
    14:     uint8_t dt = 0;     //zmienna sluzy do obslugi nacisniec wiecej niz jeden przycisk
00000037 60.e0                LDI R22,0x00      Load immediate
    15:     for (uint8_t i = 4; i < 8; i++)
00000038 4e.c0                RJMP PC+0x004F        Relative jump
--- C:\Users\Maksym\OneDrive\##########\Techniki microprocesorowe\Lab4\Zadanie1\Zadanie1\Debug/.././main.c
    17:         PORTA= ~_BV(i); // wprowadz 0 na i-tej pozycji i 1 na innych pozycjach
00000039 81.e0                LDI R24,0x01      Load immediate
0000003A 90.e0                LDI R25,0x00      Load immediate
0000003B 03.2e                MOV R0,R19        Copy register
0000003C 02.c0                RJMP PC+0x0003        Relative jump
0000003D 88.0f                LSL R24       Logical Shift Left
0000003E 99.1f                ROL R25       Rotate Left Through Carry
0000003F 0a.94                DEC R0        Decrement
00000040 e2.f7                BRPL PC-0x03      Branch if plus
00000041 80.95                COM R24       One's complement
00000042 8b.bb                OUT 0x1B,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);
00000043 89.ef                LDI R24,0xF9      Load immediate
00000044 90.e0                LDI R25,0x00      Load immediate
00000045 01.97                SBIW R24,0x01     Subtract immediate from word
00000046 f1.f7                BRNE PC-0x01      Branch if not equal
00000047 00.c0                RJMP PC+0x0001        Relative jump
00000048 00.00                NOP       No operation
--- C:\Users\Maksym\OneDrive\##########\Techniki microprocesorowe\Lab4\Zadanie1\Zadanie1\Debug/.././main.c
    19:         sk = PINA;      //odczyt stanu klawiatury
00000049 89.b3                IN R24,0x19       In from I/O location
    20:         if(sk == 0xFF)  //sprawdzanie stanu klawiatury
0000004A 8f.3f                CPI R24,0xFF      Compare with immediate
0000004B 11.f4                BRNE PC+0x03      Branch if not equal
    22:             PORTB = 0xFF;   //jesli nacisnieto 2 i wiecej przyciskow niech wszystkie diody zaswieca
0000004C 88.bb                OUT 0x18,R24      Out to I/O location
--- C:\Users\Maksym\OneDrive\##########\Techniki microprocesorowe\Lab4\Zadanie1\Zadanie1\Debug/.././main.c
    23:             break;
0000004D 08.95                RET       Subroutine return
--- 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);
0000004E 89.ef                LDI R24,0xF9      Load immediate
0000004F 90.e0                LDI R25,0x00      Load immediate
00000050 01.97                SBIW R24,0x01     Subtract immediate from word
00000051 f1.f7                BRNE PC-0x01      Branch if not equal
00000052 00.c0                RJMP PC+0x0001        Relative jump
00000053 00.00                NOP       No operation
--- C:\Users\Maksym\OneDrive\##########\Techniki microprocesorowe\Lab4\Zadanie1\Zadanie1\Debug/.././main.c
    26:         x=PINA&0x0F;    // odczyt i zapamietowanie stanu klawiatury
00000054 29.b3                IN R18,0x19       In from I/O location
00000055 2f.70                ANDI R18,0x0F     Logical AND with immediate
    27:         if (x == (PINA&0x0F))//petla if eliminujaca mozliwosc wystapienia drgania stykow
00000056 42.2f                MOV R20,R18       Copy register
00000057 50.e0                LDI R21,0x00      Load immediate
00000058 89.b3                IN R24,0x19       In from I/O location
00000059 8f.70                ANDI R24,0x0F     Logical AND with immediate
0000005A 90.e0                LDI R25,0x00      Load immediate
0000005B 48.17                CP R20,R24        Compare
0000005C 59.07                CPC R21,R25       Compare with carry
0000005D 41.f5                BRNE PC+0x29      Branch if not equal
```

```
    29:             switch(x)   //petla switch zalezna od zmiennej x, w ktorej znajduje sie odczytany stan klawiatury
0000005E 2d.30                  CPI R18,0x0D      Compare with immediate
0000005F 89.f0                  BREQ PC+0x12      Branch if equal
00000060 28.f4                  BRCC PC+0x06      Branch if carry cleared
00000061 27.30                  CPI R18,0x07      Compare with immediate
00000062 c1.f0                  BREQ PC+0x19      Branch if equal
00000063 2b.30                  CPI R18,0x0B      Compare with immediate
00000064 89.f0                  BREQ PC+0x12      Branch if equal
00000065 1a.c0                  RJMP PC+0x001B        Relative jump
00000066 2e.30                  CPI R18,0x0E      Compare with immediate
00000067 21.f0                  BREQ PC+0x05      Branch if equal
00000068 2f.30                  CPI R18,0x0F      Compare with immediate
00000069 b1.f4                  BRNE PC+0x17      Branch if not equal
    33:                     PORTB=0;        //zaden diod nie swieci
0000006A 18.ba                  OUT 0x18,R1       Out to I/O location
    34:                     break;
0000006B 16.c0                  RJMP PC+0x0017        Relative jump
    39:                     dt++;           //inkrementacja zmiennej dt
0000006C 6f.5f                  SUBI R22,0xFF     Subtract immediate
    40:                     PORTB = i-3;    //zaswiecenie numeru przyciska w postaci binarnej (od 1 do 4)
0000006D 8d.ef                  LDI R24,0xFD      Load immediate
0000006E 83.0f                  ADD R24,R19       Add without carry
0000006F 88.bb                  OUT 0x18,R24      Out to I/O location
    41:                     break;
00000070 11.c0                  RJMP PC+0x0012        Relative jump
    46:                     dt++;           //inkrementacja zmiennej dt
00000071 6f.5f                  SUBI R22,0xFF     Subtract immediate
    47:                     PORTB = i+1;    //zaswiecenie numeru przyciska w postaci binarnej (od 5 do 8)
00000072 81.e0                  LDI R24,0x01      Load immediate
00000073 83.0f                  ADD R24,R19       Add without carry
00000074 88.bb                  OUT 0x18,R24      Out to I/O location
    48:                     break;
```

```
00000075 0c.c0                  RJMP PC+0x000D        Relative jump
    53:                     dt++;           //inkrementacja zmiennej dt
00000076 6f.5f                  SUBI R22,0xFF     Subtract immediate
    54:                     PORTB = i+5;    //zaswiecenie numeru przyciska w postaci binarnej (od 9 do 12)
00000077 85.e0                  LDI R24,0x05      Load immediate
00000078 83.0f                  ADD R24,R19       Add without carry
00000079 88.bb                  OUT 0x18,R24      Out to I/O location
    55:                     break;
0000007A 07.c0                  RJMP PC+0x0008        Relative jump
    60:                     dt++;           //inkrementacja zmiennej dt
0000007B 6f.5f                  SUBI R22,0xFF     Subtract immediate
    61:                     PORTB = i+9;    //zaswiecenie numeru przyciska w postaci binarnej (od 13 do 16)
0000007C 89.e0                  LDI R24,0x09      Load immediate
0000007D 83.0f                  ADD R24,R19       Add without carry
0000007E 88.bb                  OUT 0x18,R24      Out to I/O location
--- C:\Users\Maksym\OneDrive\[illegible]\Techniki microprocesorowe\Lab4\Zadanie1\Zadanie1\Debug/.././main.c
    62:                     break;
0000007F 02.c0                  RJMP PC+0x0003        Relative jump
    67:                     PORTB = 0xFF;   //zaswiecenie wszystkich diod
00000080 8f.ef                  SER R24       Set Register
00000081 88.bb                  OUT 0x18,R24      Out to I/O location
    71:             if (dt > 1)             // jesli podczas wykonania instrukcji case wartosc dt jest wiecej niz 1, znaczy to, ze wcisnieto dwa i wiecej przyciski
00000082 62.30                  CPI R22,0x02      Compare with immediate
00000083 10.f0                  BRCS PC+0x03      Branch if carry set
    73:                 PORTB = 0xFF;       //zaswiecenie wszystkich diod
00000084 8f.ef                  SER R24       Set Register
00000085 88.bb                  OUT 0x18,R24      Out to I/O location
    15:     for (uint8_t i = 4; i < 8; i++)
00000086 3f.5f                  SUBI R19,0xFF     Subtract immediate
--- No source file ---------------------------------------------------------
--- C:\Users\Maksym\OneDrive\[illegible]\Techniki microprocesorowe\Lab4\Zadanie1\Zadanie1\Debug/.././main.c
    80: {
    81:     DDRA = 0xF0;    //ustawienie 4 najmlodszych bitow na wejscia i 4 najstarszych na wyjscia
0000008B 80.ef                  LDI R24,0xF0      Load immediate
0000008C 8a.bb                  OUT 0x1A,R24      Out to I/O location
    82:     PORTA = 0x0F;   //ustawienie 4 najmlodszych bitow na wejscia podciagniete i 4 najstarszych na 0
0000008D 8f.e0                  LDI R24,0x0F      Load immediate
0000008E 8b.bb                  OUT 0x1B,R24      Out to I/O location
    84:     DDRB = 0xFF;    //ustawianie kierunku danych dla diod na wyjscie
0000008F 8f.ef                  SER R24       Set Register
00000090 87.bb                  OUT 0x17,R24      Out to I/O location
    85:     PORTB = 0x00;   //ustawianie wszystkich diod na 0
00000091 18.ba                  OUT 0x18,R1       Out to I/O location
    89:         getkey();   // wywolanie funkcji getkey()
00000092 0e.94.36.00            CALL 0x00000036       Call subroutine
00000094 fd.cf                  RJMP PC-0x0002        Relative jump
```

Source code:

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

void getkey()
{
	uint8_t x, sk;			//inicjalizacja zmiennych x i sk
	uint8_t dt = 0;			//zmienna sluzy do obslugi nacisniec wiecej niz jeden
					przycisk
```

```
for (uint8_t i = 4; i < 8; i++)
{
	PORTA= ~_BV(i);		// wprowadz 0 na i-tej pozycji i 1 na innych pozycjach
	_delay_ms(1);		//opoznienie co 1 ms
	sk = PINA;		//odczyt stanu klawiatury
	if(sk == 0xFF)		//sprawdzanie stanu klawiatury
	{
		PORTB = 0xFF; //jesli nacisnieto 2 i wiecej przyciskow niech
				wszystkie diody zaswieca
		break;
	}
	_delay_ms(1);		//opoznienie co 1 ms
	x=PINA&0x0F;		// odczyt i zapamietowanie stanu klawiatury
	if (x == (PINA&0x0F))//petla if eliminujaca mozliwosc wystapienia drgania
				stykow
	{
		switch(x)	//petla switch zalezna od zmiennej x, w ktorej znajduje
				sie odczytany stan klawiatury
		{
			case 0b00001111:	//jesli zaden przycisk nie jest wcisniety
			{
				PORTB=0;		//zaden diod nie swieci
				break;
			}

			case 0b00001110:	//jesli wcisnieto przycisk z wiersza 1
			{
				dt++;			//inkrementacja zmiennej dt
				PORTB = i-3;	//zaswiecenie numeru przyciska w postaci
						binarnej (od 1 do 4)
				break;
			}

			case 0b00001101:	//jesli wcisnieto przycisk z wiersza 2
			{
				dt++;			//inkrementacja zmiennej dt
				PORTB = i+1;	//zaswiecenie numeru przyciska w postaci
						binarnej (od 5 do 8)
				break;
			}

			case 0b00001011:	//jesli wcisnieto przycisk z wiersza 3
			{
				dt++;			//inkrementacja zmiennej dt
				PORTB = i+5;	//zaswiecenie numeru przyciska w postaci
						binarnej (od 9 do 12)
				break;
			}

			case 0b00000111:	//jesli wcisnieto przycisk z wiersza 4
			{
				dt++;			//inkrementacja zmiennej dt
				PORTB = i+9;	//zaswiecenie numeru przyciska w postaci
						binarnej (od 13 do 16)
				break;
			}

			default:		//jesli zadna instrukcja nie byla
						wykonana, znaczy to, ze wcisnieto dwa i
						wiecej przyciski
			{
				PORTB = 0xFF; //zaswiecenie wszystkich diod
				break;
			}
		}
```

```
                    if (dt > 1)                 // jesli podczas wykonania instrukcji case
                                                wartosc dt jest wiecej niz 1, znaczy to,
                                                ze wcisnieto dwa i wiecej przyciski
                    {
                            PORTB = 0xFF;       //zaswiecenie wszystkich diod
                    }
            }
        }
}


int main(void)

{
        DDRA = 0xF0;            //ustawienie 4 najmlodszych bitow na wejscia i 4 najstarszych
                                na wyjscia
        PORTA = 0x0F;           //ustawienie 4 najmlodszych bitow na wejscia podciagniete i 4
                                najstarszych na 0

        DDRB = 0xFF;            //ustawianie kierunku danych dla diod na wyjscie
        PORTB = 0x00;           //ustawianie wszystkich diod na 0

    while (1)
    {
            getkey();       // wywolanie funkcji getkey()
    }
}
```

2. Zadanie 2

   Disassembly

```
       --- C:\Users\Maksym\OneDrive\[illegible]\Techniki
microprocesorowe\Lab4\Zadanie2\Zadanie2\Debug/.././main.c
     7: uint8_t przycisk(uint8_t port, uint8_t pin) {
     8:       uint8_t i = 1;
     9:       if (port == PINA) {i=0;}
0000003E 99.b3               IN R25,0x19             In from I/O location
0000003F 98.17               CP R25,R24        Compare
00000040 11.f0               BREQ PC+0x03            Branch if equal
--- C:\Users\Maksym\OneDrive\[illegible]\Techniki
microprocesorowe\Lab4\Zadanie2\Zadanie2\Debug/.././main.c
     8:       uint8_t i = 1;
00000041 e1.e0               LDI R30,0x01            Load immediate
00000042 01.c0               RJMP PC+0x0002          Relative jump
     9:       if (port == PINA) {i=0;}
00000043 e0.e0               LDI R30,0x00            Load immediate
    10:       if (port & (1 << pin)) {
00000044 90.e0               LDI R25,0x00            Load immediate
00000045 06.2e               MOV R0,R22        Copy register
00000046 02.c0               RJMP PC+0x0003          Relative jump
00000047 95.95               ASR R25           Arithmetic shift right
00000048 87.95               ROR R24           Rotate right through carry
00000049 0a.94               DEC R0            Decrement
0000004A e2.f7               BRPL PC-0x03            Branch if plus
0000004B 80.ff               SBRS R24,0        Skip if bit in register set
0000004C 15.c0               RJMP PC+0x0016          Relative jump
    11:             if (!(state[i] & (1 << pin))) {
0000004D f0.e0               LDI R31,0x00            Load immediate
0000004E e0.5a               SUBI R30,0xA0           Subtract immediate
0000004F ff.4f               SBCI R31,0xFF           Subtract immediate with carry
00000050 80.81               LDD R24,Z+0             Load indirect with displacement
00000051 90.e0               LDI R25,0x00            Load immediate
00000052 02.c0               RJMP PC+0x0003          Relative jump
00000053 95.95               ASR R25           Arithmetic shift right
00000054 87.95               ROR R24           Rotate right through carry
00000055 6a.95               DEC R22           Decrement
00000056 e2.f7               BRPL PC-0x03            Branch if plus
00000057 80.fd               SBRC R24,0        Skip if bit in register cleared
00000058 08.c0               RJMP PC+0x0009          Relative jump
--- 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);
00000059 83.ec               LDI R24,0xC3            Load immediate
0000005A 99.e0               LDI R25,0x09            Load immediate
0000005B 01.97               SBIW R24,0x01           Subtract immediate from word
0000005C f1.f7               BRNE PC-0x01            Branch if not equal
0000005D 00.c0               RJMP PC+0x0001          Relative jump
0000005E 00.00               NOP               No operation
--- C:\Users\Maksym\OneDrive\[illegible]\Techniki
microprocesorowe\Lab4\Zadanie2\Zadanie2\Debug/.././main.c
    14:                     return 1;
0000005F 81.e0               LDI R24,0x01            Load immediate
00000060 08.95               RET               Subroutine return
    21: }
00000061 08.95               RET               Subroutine return
    19:             return 0;
00000062 80.e0               LDI R24,0x00            Load immediate
    21: }
00000063 08.95               RET               Subroutine return
--- C:\Users\Maksym\OneDrive\[illegible]\Techniki
microprocesorowe\Lab4\Zadanie2\Zadanie2\Debug/.././main.c
    49: int main(void) {
```

```
00000064 ef.92                PUSH R14          Push register on stack
00000065 ff.92                PUSH R15          Push register on stack
00000066 1f.93                PUSH R17          Push register on stack
00000067 cf.93                PUSH R28          Push register on stack
00000068 df.93                PUSH R29          Push register on stack
    51:       DDRA = 0x00;
00000069 1a.ba                OUT 0x1A,R1             Out to I/O location
    52:       DDRB = 0xff;
0000006A 8f.ef                SER R24           Set Register
0000006B 87.bb                OUT 0x17,R24            Out to I/O location
    53:       DDRC = 0xff;
0000006C 84.bb                OUT 0x14,R24            Out to I/O location
    54:       DDRD = 0x00;
0000006D 11.ba                OUT 0x11,R1             Out to I/O location
    55:       PORTA = 0xff;
0000006E 8b.bb                OUT 0x1B,R24            Out to I/O location
    56:       PORTD = 0xff;
0000006F 82.bb                OUT 0x12,R24            Out to I/O location
    57:       PORTB = 0;
00000070 18.ba                OUT 0x18,R1             Out to I/O location
    58:       PORTC = 0;
00000071 15.ba                OUT 0x15,R1             Out to I/O location
    60:       j=0;k=0;state[0]=0;state[1]=0;
00000072 e0.e6                LDI R30,0x60            Load immediate
00000073 f0.e0                LDI R31,0x00            Load immediate
00000074 10.82                STD Z+0,R1        Store indirect with displacement
00000075 11.82                STD Z+1,R1        Store indirect with displacement
00000076 10.e0                LDI R17,0x00            Load immediate
00000077 d0.e0                LDI R29,0x00            Load immediate
    61:       while (j<2) {
00000078 2c.c0                RJMP PC+0x002D          Relative jump
    62:             wyswietl(count[j]); //pokaz wprowadzana liczbe
00000079 ed.2e                MOV R14,R29             Copy register
0000007A f1.2c                MOV R15,R1        Copy register
    63:             for (i=0;i<8;i++) // odczytanie liczby
0000007B c0.e0                LDI R28,0x00            Load immediate
0000007C 15.c0                RJMP PC+0x0016          Relative jump
    64:             if ( (przycisk( &PINA, i)) && k<4 ) {
0000007D 6c.2f                MOV R22,R28             Copy register
0000007E 89.e3                LDI R24,0x39            Load immediate
0000007F 0e.94.3e.00          CALL 0x0000003E         Call subroutine
00000081 88.23                TST R24           Test for Zero or Minus
00000082 71.f0                BREQ PC+0x0F            Branch if equal
--- No source file -----------------------------------------------------------
00000083 14.30                CPI R17,0x04            Compare with immediate
00000084 60.f4                BRCC PC+0x0D            Branch if carry cleared
--- C:\Users\Maksym\OneDrive\[illegible]\Techniki
microprocesorowe\Lab4\Zadanie2\Zadanie2\Debug/.././main.c
    65:                   count[j]= (count[j] * 10) + i;
00000085 f7.01                MOVW R30,R14            Copy register pair
00000086 ee.59                SUBI R30,0x9E           Subtract immediate
00000087 ff.4f                SBCI R31,0xFF           Subtract immediate with carry
00000088 80.81                LDD R24,Z+0             Load indirect with displacement
00000089 88.0f                LSL R24           Logical Shift Left
0000008A 98.2f                MOV R25,R24             Copy register
0000008B 99.0f                LSL R25           Logical Shift Left
0000008C 99.0f                LSL R25           Logical Shift Left
0000008D 89.0f                ADD R24,R25             Add without carry
0000008E 8c.0f                ADD R24,R28             Add without carry
0000008F 80.83                STD Z+0,R24             Store indirect with displacement
    66:                   k++;
00000090 1f.5f                SUBI R17,0xFF           Subtract immediate
    63:             for (i=0;i<8;i++) // odczytanie liczby
00000091 cf.5f                SUBI R28,0xFF           Subtract immediate
--- No source file -----------------------------------------------------------
```

```
00000092 c8.30               CPI R28,0x08            Compare with immediate
00000093 48.f3               BRCS PC-0x16            Branch if carry set
00000094 c0.e0               LDI R28,0x00            Load immediate
00000095 0d.c0               RJMP PC+0x000E          Relative jump
00000096 6c.2f               MOV R22,R28             Copy register
00000097 80.e3               LDI R24,0x30            Load immediate
00000098 0e.94.3e.00         CALL 0x0000003E         Call subroutine
0000009A 88.23               TST R24          Test for Zero or Minus
0000009B 31.f0               BREQ PC+0x07            Branch if equal
0000009C d1.11               CPSE R29,R1             Compare, skip if equal
0000009D 02.c0               RJMP PC+0x0003          Relative jump
0000009E c0.93.64.00         STS 0x0064,R28          Store direct to data space
000000A0 df.5f               SUBI R29,0xFF           Subtract immediate
000000A1 10.e0               LDI R17,0x00            Load immediate
000000A2 cf.5f               SUBI R28,0xFF           Subtract immediate
000000A3 c8.30               CPI R28,0x08            Compare with immediate
000000A4 88.f3               BRCS PC-0x0E            Branch if carry set
000000A5 d2.30               CPI R29,0x02            Compare with immediate
000000A6 90.f2               BRCS PC-0x2D            Branch if carry set
000000A7 80.91.64.00         LDS R24,0x0064          Load direct from data space
--- C:\Users\Maksym\OneDrive\[illegible]\Techniki
microprocesorowe\Lab4\Zadanie2\Zadanie2\Debug/.././main.c
    75:       switch(j) {
000000A9 82.30               CPI R24,0x02            Compare with immediate
000000AA c9.f0               BREQ PC+0x1A            Branch if equal
000000AB 28.f4               BRCC PC+0x06            Branch if carry cleared
000000AC 88.23               TST R24          Test for Zero or Minus
000000AD 41.f0               BREQ PC+0x09            Branch if equal
000000AE 81.30               CPI R24,0x01            Compare with immediate
000000AF 69.f0               BREQ PC+0x0E            Branch if equal
000000B0 30.c0               RJMP PC+0x0031          Relative jump
000000B1 83.30               CPI R24,0x03            Compare with immediate
000000B2 d1.f0               BREQ PC+0x1B            Branch if equal
000000B3 84.30               CPI R24,0x04            Compare with immediate
000000B4 29.f1               BREQ PC+0x26            Branch if equal
000000B5 2b.c0               RJMP PC+0x002C          Relative jump
    77:             count[2]=count[0]+count[1];
000000B6 e2.e6               LDI R30,0x62            Load immediate
000000B7 f0.e0               LDI R31,0x00            Load immediate
000000B8 90.81               LDD R25,Z+0             Load indirect with displacement
000000B9 81.81               LDD R24,Z+1             Load indirect with displacement
000000BA 89.0f               ADD R24,R25             Add without carry
000000BB 82.83               STD Z+2,R24             Store indirect with displacement
    78:             break;
000000BC 24.c0               RJMP PC+0x0025          Relative jump
    80:             count[2]=count[0]-count[1];
000000BD e2.e6               LDI R30,0x62            Load immediate
000000BE f0.e0               LDI R31,0x00            Load immediate
000000BF 80.81               LDD R24,Z+0             Load indirect with displacement
000000C0 91.81               LDD R25,Z+1             Load indirect with displacement
000000C1 89.1b               SUB R24,R25             Subtract without carry
000000C2 82.83               STD Z+2,R24             Store indirect with displacement
    81:             break;
000000C3 1d.c0               RJMP PC+0x001E          Relative jump
    83:             count[2]=count[0]*count[1];
000000C4 e2.e6               LDI R30,0x62            Load immediate
000000C5 f0.e0               LDI R31,0x00            Load immediate
000000C6 80.81               LDD R24,Z+0             Load indirect with displacement
000000C7 91.81               LDD R25,Z+1             Load indirect with displacement
000000C8 89.9f               MUL R24,R25             Multiply unsigned
000000C9 80.2d               MOV R24,R0       Copy register
000000CA 11.24               CLR R1           Clear Register
000000CB 82.83               STD Z+2,R24             Store indirect with displacement
--- C:\Users\Maksym\OneDrive\[illegible]\Techniki
microprocesorowe\Lab4\Zadanie2\Zadanie2\Debug/.././main.c
```

```
    84:             break;
000000CC 14.c0                  RJMP PC+0x0015          Relative jump
    86:             count[2]=(count[0]*10)/count[1];
000000CD e2.e6                  LDI R30,0x62            Load immediate
000000CE f0.e0                  LDI R31,0x00            Load immediate
000000CF 80.81                  LDD R24,Z+0             Load indirect with displacement
000000D0 2a.e0                  LDI R18,0x0A            Load immediate
000000D1 82.9f                  MUL R24,R18             Multiply unsigned
000000D2 c0.01                  MOVW R24,R0             Copy register pair
000000D3 11.24                  CLR R1            Clear Register
000000D4 61.81                  LDD R22,Z+1             Load indirect with displacement
000000D5 70.e0                  LDI R23,0x00            Load immediate
000000D6 0e.94.e9.00            CALL 0x000000E9         Call subroutine
000000D8 62.83                  STD Z+2,R22             Store indirect with displacement
    87:             break;
000000D9 07.c0                  RJMP PC+0x0008          Relative jump
    89:             if(count[1]=count[0]){
000000DA e2.e6                  LDI R30,0x62            Load immediate
000000DB f0.e0                  LDI R31,0x00            Load immediate
000000DC 80.81                  LDD R24,Z+0             Load indirect with displacement
000000DD 81.83                  STD Z+1,R24             Store indirect with displacement
000000DE 81.11                  CPSE R24,R1             Compare, skip if equal
    90:                     count[2]=count[1];
000000DF 80.93.64.00            STS 0x0064,R24          Store direct to data space
   102: }
000000E1 80.e0                  LDI R24,0x00            Load immediate
000000E2 90.e0                  LDI R25,0x00            Load immediate
000000E3 df.91                  POP R29           Pop register from stack
000000E4 cf.91                  POP R28           Pop register from stack
000000E5 1f.91                  POP R17           Pop register from stack
000000E6 ff.90                  POP R15           Pop register from stack
000000E7 ef.90                  POP R14           Pop register from stack
000000E8 08.95                  RET               Subroutine return
```

Source code:

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

uint8_t count[3];//tablica liczb do operacji
uint8_t state[2];
uint8_t przycisk(uint8_t port, uint8_t pin) {
	uint8_t i = 1;
	if (port == PINA) {i=0;}
	if (port & (1 << pin)) {
		if (!(state[i] & (1 << pin))) {

			_delay_ms(10);
			return 1;
		}
	}
	else
	{
		return 0;
	}
}
uint8_t znaki(uint8_t znaczek) {
	/* przetwarzanie uint8 do 7LED */
	switch(znaczek) {
		case 0 : return 0x3F;
		case 1 : return 0x06;
		case 2 : return 0x5B;
		case 3 : return 0x4F;
		case 4 : return 0x66;
		case 5 : return 0x6D;
```

```
                case 6 : return 0x7D;
                case 7 : return 0x07;
                case 8 : return 0x7F;
                case 9 : return 0x6F;
                default: return 0xAA;
        }
        return 0xAA;
}
void wyswietl(uint8_t num) {//fukcja do wyswietlanie na 7LED
        uint8_t led[4];
        if (num <= 9999) {
                led[3]=num/1000;//ustalanie na 1 kolumnie
                led[2]=(num%1000)/100;//ustalanie na 1,2 kolumnie
                led[1]=(num%100)/10;// ustalanie na 1,2,3 kolumnie
                led[0]=num%10;//ustalanie na 1,2,3,4 kolumnie
        }
}

int main(void) {
        //deklaracja portow A,B,C,D
        DDRA = 0x00;
        DDRB = 0xff;
        DDRC = 0xff;
        DDRD = 0x00;
        PORTA = 0xff;
        PORTD = 0xff;
        PORTB = 0;
        PORTC = 0;
        uint8_t i,j,k;
        j=0;k=0;state[0]=0;state[1]=0;
        while (j<2) {
                wyswietl(count[j]); //pokaz wprowadzana liczbe
                for (i=0;i<8;i++) // odczytanie liczby
                if ( (przycisk( &PINA, i)) && k<4 ) {
                        count[j]= (count[j] * 10) + i;
                        k++;
                }
                for (i=0;i<8;i++) //odczytujemy operacje dla liczb
                if (przycisk( &PIND, i)) {
                        if (!j) count[2]=i;
                        j++;k=0;
                }
        }
        j=count[2];
        switch(j) {
                case 0: // dodawanie
                count[2]=count[0]+count[1];
                break;
                case 1: // odejmowanie
                count[2]=count[0]-count[1];
                break;
                case 2: // mnozenie
                count[2]=count[0]*count[1];
                break;
                case 3: // dzielenie
                count[2]=(count[0]*10)/count[1];
                break;
                case 4: //czy jest rowne
                if(count[1]=count[0]){
                        count[2]=count[1];
                }
                else
                {
                        break;
                }
```

```
        }
        while(1)
        {
                wyswietl(count[2]);//wyswietlamy wynik
                return 0;
        }
}
```