МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ»

Кафедра ИиСП

Отчет

по лабораторной работе № 2

по дисциплине «Машинно-зависимые языки программирования»

Вариант 20

Выполнил: ст. гр. ПС-11

Ложкин С.А.

Проверил: доцент, доцент кафедры ИиСП Баев А.А.

г. Йошкар-Ола

2023

Цель работы: перевести ассемблерный код в аналогичный код на C

Задания на лабораторную работу:

1. По документации найти точку входа программы
2. Внутри неё сопоставить каждую команду с аналогичной на C
3. Все полученные команды записать в конечную программу

1. Теоретические сведения

Начнём с 1 команды программы, переходим на 0x68, доходим до call 0x80. Значит, мы вызвали программу, следовательно, точкой входа программы будет код с 80 ячейки. Получается, что это функция main.

От неё разберём все элементы, опираясь на документацию.

2. Практическая часть

```
80: 3D 9A        sbi 0x07,5      ;    DDRC |= (1 << DDC5)
```

адрес регистра 0x07 это DDRC, его 5 бит соответсвует DDC5

```
82: 81 E0        ldi r24,0x01    ; 1  uint8_t dir = 1
84: 91 E0        ldi r25,0x01    ; 1
86: 89 27        eor r24,r25          dir ^= 1
```

Загружаем значение в регистр и применяем а нём исключающее или. Получается, r24 это переменная, а значение в r25 - просто константа.

```
88: 11 F0        breq .+4        ; 0x8e    if(dir)
8a: 45 9A        sbi 0x08,5      ; 8       PORTC |= (1 << PINC5);
8c: 01 C0        rjmp .+2        ; 0x90else
8e: 45 98        cbi 0x08,5      ; 8       PORTC &= ~(1 << PINC5);
```

Здесь условный оператор, который смотрит на флаг нуля. Если r24 = 1, то ставим 5 бит в PORTC, иначе очищаем его

```
90: 20 E8        ldi r18,0x80 ; 128
92: 3A E6        ldi r19,0x6A    ; 106
94: 4D E1        ldi r20,0x1D    ; 29
96: 21 50        subi r18, 0x01  ; 1
98: 30 40        sbci r19, 0x00  ; 0
9a: 40 40        sbci r20, 0x00  ; 0
9c: E1 F7        brne .-8        ; 0x96
```

Это задержка программы

```
9e: 00 00        nop
a0: F2 CF        rjmp .-28       ; 0x86
```

Переход на 0x86. Переход абсолютный, следовательно, с 0x86 по 9e идёт бесконечный цикл.

```
a2: F8 94          cli
a4: FF CF          rjmp .-2        ; 0xA4
```

Надо посчитать задержку в миллисекундах.

Три регистра (18, 19, 20) представляют число N = 0x1D6A80 = 1927808

Столько раз пройдёт цикл для задержки.

За итерацию проходит 5 тактов: subi - 1, sbci - 1, brne - 2

Но на последнем сравнении brne - 1

Общее количество тактов во время задержки = 5 * N - 1 = 9.639.039

В конце вычислений прибавляем 4 такта: загрузка в регистры - 3, nop – 1

9.639.039 + 4 = 9.639.043

9.639.043 / 16.000.000 = 0,6024401875 секунд, округлим до 602мс

Получается следующее:

```
int main(void)
{
	DDRC |= (1 << DDC5)
	uint8_t dir = 1;
	while(1)
	{
		dir ^= 1;
		if(dir) PORTC |= (1 << PINC5);
```

```
            else PORTC &= ~(1 << PINC5);
            _delay_ms(602);
        }
}
```

Выводы

В ходе выполнения данной лабораторной работы я отлично понял работу компилятора C для микроконтроллера AVR , реализацию функции задержки в процессоре и теперь могу применять эти знания на практике.