

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

Кафедра ИиСП

Отчет
по лабораторной работе № 4
по дисциплине «Машинно-зависимые языки программирования»
Вариант 1

Выполнил: ст. гр. ПС-11

Маркин И. А.

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

г. Йошкар-Ола
2025

Цель работы: Научиться переводить код с C на ASM.

Задания на лабораторную работу:

Исходный код со 2 лабораторной работы с языка C перевести на язык ASM и проверить его работоспособность

1. Теоретические сведения

<https://cxem.net/mc/book26.php>

https://ru.wikipedia.org/wiki/Intel_HEX

<http://www.gaw.ru/html.cgi/txt/doc/micros/avr/asm/start.htm>

<http://av-assembler.ru/mc/status-register.php>

<https://www.mcu4you.ru/tablicy-komand-assemblera-avr/>

<https://trolsoft.ru/ru/avr-assembler>

2. Практическая часть

Исходный код на C:

```
#include <avr/io.h>
#define F_CPU 16000000UL//16MHZ
#include <util/delay.h>
int main(void)
{
    DDRD |= (1 << 6);
    DDRD |= (1 << 1);

    while(1)
    {
        if ((PIND & (1 << 1)) == 0) PORTD |= (1 << PIND6);
        else PORTD &= ~(1 << PIND6);
        _delay_ms(674);
    }
}
```

Код на ASM:

```
main:
    sbi DDRD, 0x06
    sbi DDRD, 0x01
    rjmp check
check:
    sbic PIND, 0x01
    rjmp bit_empty
    rjmp bit_full
bit_empty:
    sbi PORTD, 0x06
    rjmp set_delay
bit_full:
    cbi PORTD, 0x06
    rjmp set_delay
set_delay:
    ldi r18, 0x62
    ldi r24, 0xf2
    ldi r25, 0x20
    rjmp delay_loop
set_delay_loop:
    subi r18, 0x01
    sbci r24, 0x00
    sbci r25, 0x00
    brne delay_loop

    rjmp check
```

Для того, чтобы получить исходный код, надо разбить код на C на части.

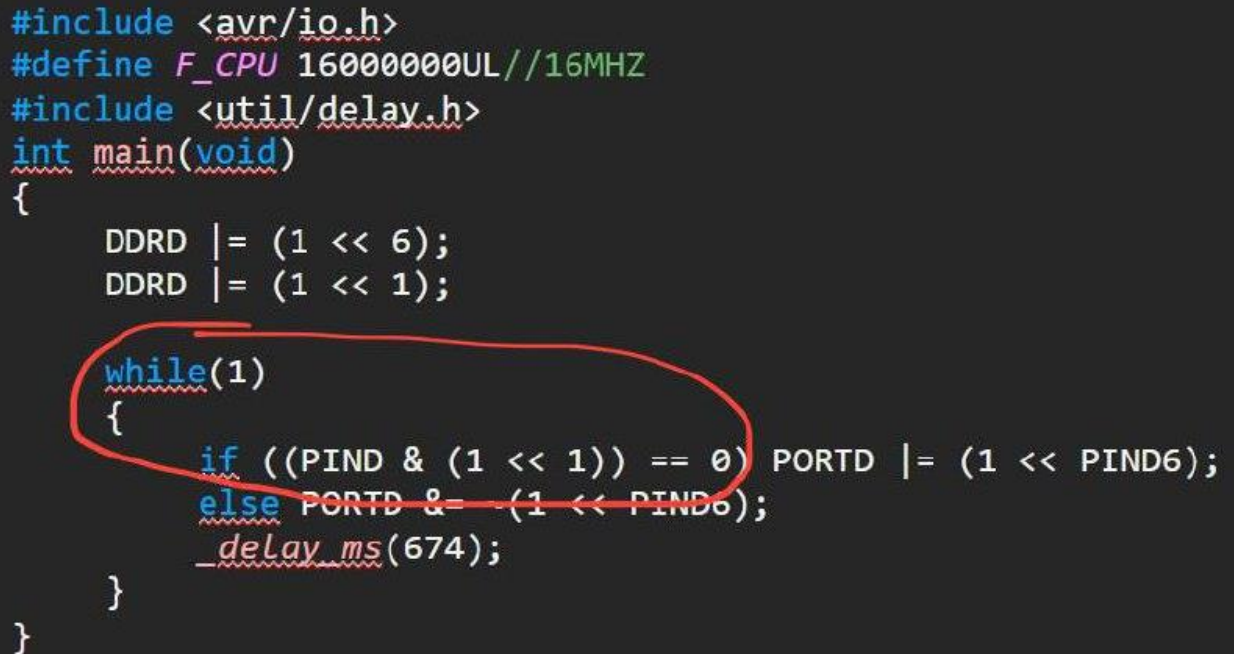
```
int main(void)
{
    DDRD |= (1 << 6);
    DDRD |= (1 << 1);
}
```

Из этого фрагмента кода получаем:

```
main:
    sbi DDRD, 0x06
    sbi DDRD, 0x01
    rjmp check
```

Команда `rjmp check` перенесет нас на следующий фрагмент код

В случае чего получим

A screenshot of C code on a dark background. The code includes headers for AVR I/O and delay functions, defines the CPU frequency, and sets up the main function. It configures DDRD and then enters a while(1) loop. Inside the loop, it checks if PIND bit 1 is low, and if so, sets PORTD bit 6. Otherwise, it clears PORTD bit 6. A red circle highlights the while(1) loop and its body. The code is as follows:

```
#include <avr/io.h>
#define F_CPU 16000000UL //16MHZ
#include <util/delay.h>
int main(void)
{
    DDRD |= (1 << 6);
    DDRD |= (1 << 1);

    while(1)
    {
        if ((PIND & (1 << 1)) == 0) PORTD |= (1 << PIND6);
        else PORTD &= ~(1 << PIND6);
        _delay_ms(674);
    }
}
```

check:

```
    sbic PIND, 0x01
    rjmp bit_empty
    rjmp bit_full
```

За `rjmp bit_empty` и `rjmp bit_full` отвечают следующие части кода, а именно:

```

#include <avr/io.h>
#define F_CPU 16000000UL//16MHZ
#include <util/delay.h>
int main(void)
{
    DDRD |= (1 << 6);
    DDRD |= (1 << 1);

    while(1)
    {
        if ((PIND & (1 << 1)) == 0) PORTD |= (1 << PIND6);
        else PORTD &= ~(1 << PIND6);
        _delay_ms(674);
    }
}

```

Чтобы создать нам задержку, мы должны вспомнить 1 и 2 лабораторную работу и фрагмент кода, который давал нам задержку в 674 мс:

```

ldi r18, 0x62
ldi r24, 0xf2
ldi r25, 0x20

```

В итоге получаем готовую задержку, а в конце делаем часть, которая будет отвечать за цикл данной программы.

Выводы: В данной лабораторной работе я научился переводить с языка C на ASM