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

Кафедра ИиСП

Отчет
по лабораторной работе № 1, 2

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
2024

Цель работы: Научиться восстанавливать по HEX коду ассемблерный код,строить алгоритмы и писать код на C

Задания на лабораторную работу:

1. Восстановить HEX код в ассемблерный код
2. Построить алгоритм и написать код на C

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**

   0: 0c 94 34 00 jmp 0x68 ; 0x68

   4: 0c 94 3e 00 jmp 0x7c ; 0x7c

   8: 0c 94 3e 00 jmp 0x7c ; 0x7c

   c: 0c 94 3e 00 jmp 0x7c ; 0x7c

   10: 0c 94 3e 00 jmp 0x7c ; 0x7c

   14: 0c 94 3e 00 jmp 0x7c ; 0x7c

   18: 0c 94 3e 00 jmp 0x7c ; 0x7c

   1c: 0c 94 3e 00 jmp 0x7c ; 0x7c

   20: 0c 94 3e 00 jmp 0x7c ; 0x7c

   24: 0c 94 3e 00 jmp 0x7c ; 0x7c

   28: 0c 94 3e 00 jmp 0x7c ; 0x7c

   2c: 0c 94 3e 00 jmp 0x7c ; 0x7c

   30: 0c 94 3e 00 jmp 0x7c ; 0x7c

34: 0c 94 3e 00 jmp 0x7c ; 0x7c

38: 0c 94 3e 00 jmp 0x7c ; 0x7c

3c: 0c 94 3e 00 jmp 0x7c ; 0x7c

40: 0c 94 3e 00 jmp 0x7c ; 0x7c

44: 0c 94 3e 00 jmp 0x7c ; 0x7c

48: 0c 94 3e 00 jmp 0x7c ; 0x7c

4c: 0c 94 3e 00 jmp 0x7c ; 0x7c

50: 0c 94 3e 00 jmp 0x7c ; 0x7c

54: 0c 94 3e 00 jmp 0x7c ; 0x7c

58: 0c 94 3e 00 jmp 0x7c ; 0x7c

5c: 0c 94 3e 00 jmp 0x7c ; 0x7c

60: 0c 94 3e 00 jmp 0x7c ; 0x7c

64: 0c 94 3e 00 jmp 0x7c ; 0x7c

68: 11 24 eor r1, r1

6a: 1f be out 0x3f, r1 ; 63

6c: cf ef ldi r28, 0xFF ; 255

6e: d8 e0 ldi r29, 0x08 ; 8

70: de bf out 0x3e, r29 ; 62

72: cd bf out 0x3d, r28 ; 61

74: 0e 94 40 00 call 0x80 ; 0x80

78: 0c 94 52 00 jmp 0xa4 ; 0xa4

7c: 0c 94 00 00 jmp 0 ; 0x0

80: 56 9a sbi 0x0a, 0x06

82: 51 98 cbi 0x0a, 0x01

84: 59 9a sbi 0x0a, 0x01

86: 49 99 sbic 0x09, 0x01

88: 02 c0 rjmp .+4

8a: 5e 9a sbi 0x0b, 0x06

8c: 01 c0 rjmp .+2

8e: 5e 98 cbi 0x0b, 0x06

90: 22 e6 ldi r18, 0x62 ; 98

92: 82 ef ldi r24, 0xf2 ; 242

94: 90 e2 ldi r25, 0x20 ; 32

96: 21 50 subi r18, 0x01 ; 1

98: 80 40 sbci r24, 0x00 ; 0

9a: 90 40 sbci r25, 0x00 ; 0

9c: e1 f7 brne .-8

9e: 00 c0 rjmp .+0

a0: 00 c0 rjmp .+0

a2: f1 cf rjmp .-30

a4: f8 94 cli

a6: ff cf rjmp .-2

2. Построить алгоритм и написать код на С

Алгоритм программы:

1) 0: 0c 94 34 00 jmp 0x68 ; 0x68 Перебрасывает на строку 68 занимает 4 такта

2) 68: 1124 eor r1, r1 Исключающее или для регистра 1 и регистра 1 => обнуление 1 регистра

3) 6a: 1FBE out 0x3f r1; 63 Записывает регистр 1 (0) в порт SREG

4) 6c: cf ef ldi r28, 0xFF ; 255 Записывает значение 255 в регистр 28

5) 6e: d8 e0 ldi r29, 0x08 ; 8 Записывает значение 8 в регистр 29

6) 70: de bf out 0x3e, r29 ; 62 Записывает значение 29 регистра (8) в SPH

7) 72: cd bf out 0x3d, r28 ; 61 Записывает значение 28 регистра (255) в SPL

8) 74: 0e 94 40 00 call 0x80 ; 0x80 Команда call вызывает подпрограмму по адресу 0x80

9) 80: 56 9a sbi 0x0a, 0x06 Устанавливается 6 бит в DDRD

10) 82: 51 98 cbi 0x0a, 0x01 Обнуляется 1 бит в DDRD

11) 84: 59 9a sbi 0x0a, 0x01 Устанавливается 1 бит в DDRD

12) 86: 49 99 sbic 0x09, 0x01 Если бит очищен, то пропускает команду

12) 88: 02 c0 rjmp .+4 Пропускается следующая команда (rjmp +4)

13) 8a: 5e 9a sbi 0x0b, 0x06 Устанавливается 6 бит в PORTD

14) 8c: 01 c0 rjmp .+2 Пропускается следующая команда (rjmp +2)

15) 8e: 5e 98 cbi 0x0b, 0x06 Обнуляется 6 бит PORTD

16) 90: 22 e6 ldi r18, 0x62 ; 98 Устанавливается значение 98 в регистр 18

17) 92: 82 ef ldi r24, 0xf2 ; 242 Устанавливается значение 242 в регистр 24

18) 94: 90 e2 ldi r25, 0x20 ; 32 Устанавливается значение 32 в регистр 25

19) 96: 21 50 subi r18, 0x01 ; 1 Вычетается 1 из 18 регистра
20) 98: 80 40 sbci r24, 0x0 Вычетается 1 из регистра 24 (если регистр 18 равен 0)
21) 9a: 90 40 sbci r25, 0x0 Вычетается 1 из регистра 25 (если предыдущие 2 регистра равны 0)
22) 9c: e1 f7 brne .-8 Переход к действию 19) (повторяются 19, 20, 21 пока все 3 регистра не будут равны 0)
23) 9e: 00 c0 rjmp .+0 Переход на 0 бит вперед (переход на строку 0xa0)
24) a0: 00 c0 rjmp .+0 Переход на 0 бит вперед (переход на строку 0xa2)
25) a2: f1 cf rjmp .-30 Переход на строку 0x86

Подсчет задержи:
Регистры 18-25 здесь представляют одно число N = 0x20f262 = 2 159 202,

Количество итераций = N = 2 159 202

Общее количество циклов составит 5 * N – 1 = 10 796 009

rjmp .+0 добавит 2 цикла

rjmp .+0 добавит 2 цикла

Загрузка в регистры ещё 3 такта

Итого: 10 796 016 циклов

При известном значении тактовой частоты = 16 МГц можно вычислить

длительность

задержки цикла.

t = 10 796 016 / 16000000 = 0,674751 сек или 674мс

Код на С:

```
#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);
        }
}
```

Выводы: в данной лабораторной работе я научился восстанавливать HEX код в ассемблерный и попробовал написать код на C.