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

Факультет информатики и
вычислительной техники

Отчет
по лабораторной работе № 1

по дисциплине «Машинно-зависимые языки программирования»
Вариант 2

Выполнил:
Сайфутдинов Э.Р.
Студент группы ПС-14
Проверил: доцент, доцент кафедры ИиСП Баев А.А.

г. Йошкар-Ола
2024

Цель работы:

Изучение основ ассемблера и hex кода

Задания на лабораторную работу:

Перевести hex код в ассемблерный

1. **Теоретические сведения**

Пример перевода первой строки:

:10 0000 00 0C94 3400 0C94 3E00 0C94 3E00 0C94 3E00 82

1. Сначала нужно поменять соседние байты в командах:
   :10 0000 00 940C 0034 940C 003E 940C 003E 940C 003E 82
2. Команды занимают 2 байта в памяти (Исключение - команды перехода jmp, call, rjmp, которые занимают 4 байта)
3. 940C. Чтобы узнать какая это команда переведём число из шестнадцатеричной системы исчисления в двоичную:
   940C = 1001 0100 0000 1100
4. Далее найдём маску, подходящую для этого числа
   Маска: 1001 010k kkkk 110k kkkk kkkk kkkk kkkk
   Число: 1001 0100 0000 1100
   Число соответствует маске => это команда jmp, которая занимает 4 байта вместо 2, поэтому нужно добавить следующую ячейку памяти к команде. 0034 = 0000 0000 0011 0100
   Маска: 1001 010k kkkk 110k kkkk kkkk kkkk kkkk
   Число: 1001 0100 0000 1100 0000 0000 0011 0100
   По маске k = 0000 0000 0000 0011 0100 = 34. В командах jmp, call, rjmp это число нужно умножить на 2: 34 * 2 = 68
   Итог: jmp 0x68
5. Проводим эти действия с остальными командами

2. Практическая часть

:00 0C94 3400 jmp 0x68 Переход на ячейку 68

:04 0C94 3E00 jmp 0x7C Переход на ячейку 7C

:08 0C94 3E00 jmp 0x7C

:0C 0C94 3E00 jmp 0x7C

:10 0C94 3E00 jmp 0x7C

:14 0C94 3E00 jmp 0x7C

:18 0C94 3E00 jmp 0x7C

:1C 0C94 3E00 jmp 0x7C

:20 0C94 3E00 jmp 0x7C

:24 0C94 3E00 jmp 0x7C

:28 0C94 3E00 jmp 0x7C

:2C 0C94 3E00 jmp 0x7C

:30 0C94 3E00 jmp 0x7C

:34 0C94 3E00 jmp 0x7C

:38 0C94 3E00 jmp 0x7C

:3C 0C94 3E00 jmp 0x7C

:40 0C94 3E00 jmp 0x7C

:44 0C94 3E00 jmp 0x7C

:48 0C94 3E00 jmp 0x7C

:4C 0C94 3E00 jmp 0x7C

:50 0C94 3E00 jmp 0x7C

:54 0C94 3E00 jmp 0x7C

:58 0C94 3E00 jmp 0x7C

:5C 0C94 3E00 jmp 0x7C

:60 0C94 3E00 jmp 0x7C

:64 0C94 3E00 jmp 0x7C

:68 1124 eor r1, r1 Исключающее или для r1 и r1 (Результат 0)

:6A 1FBE out 0x3F, r1 Установка 0 во флаге C

:6C CFEF ldi r28, 0xFF Установка числа FF в регистр 28

:6E D8E0 ldi r29, 0x08 Установка числа 8 в регистр 29

:70 DEBF out 0x3E, r29 Установка значения r29 в SPH

:72 CDBF out 0x3D, r28 Установка значения r28 в SPL

:74 0E94 4000 call 0x80 Вызов процедуры по адресу 80, помещение адреса возврата в стек

:78 0C94 5200 jmp 0xA4

:7C 0C94 0000 jmp 0x00

:80 519A sbi 0x0A, 1 Установка 1-го бита в регистре I/O DDRD

82 5098 cbi 0x0A, 0 Очистка 0-го бита в регистре I/O DDRD

84 589A sbi 0x0B, 0 Установка 0-го бита в регистре I/O PORTD

86 4899 sbic 0x09, 0 Проверяет состояние 0-го бита в регистре I/O PIND. Если этот бит очищен, то пропускает следующую команду

88 02C0 rjmp +4 Переход на 4 ячейки вперед

8A 599A sbi 0x0B, 1 Установка 1-го бита в регистре I/O PORTD

8C 01C0 rjmp +2 Переход на 2 ячейки вперед

8E 5998 cbi 0x0B, 1 Очистка 1-го бита в регистре I/O PORTD

90 28ED ldi r2, 0xD8 Установка числа 0xD8 в r2

92 8EE9 ldi r8, 0x9E Установка числа 0x9E в r8

94 92E1 ldi r9, 0x12 Установка числа 0x12 в r9

96 2150 subi r2, 0x01 Вычитание 1 из r2

98 8040 sbci r8, 0x00 Вычитание флага переноса из r8

9A 9040 sbci r9, 0x00 Вычитание флага переноса из r8

:9C E1F7 brne -8 Если флаг Z = 0, то переход на 8 ячеек назад

:9E 00C0 rjmp +0 Ничего не делает

:A0 0000 nop Ничего не делает

:A2 F1CF rjmp -30 Переход на 30 ячеек назад

:A4 F894 cli Очистка флага глобального прерывания

:A6 FFCF rjmp -2 Переход назад на 2 ячейки