Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Лабораторная Работа №3**

**Дисциплина:** Низкоуровневое программирование

**Тема:** RISC-V

Выполнил

студент гр. 3530901/90003 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Иванов Т.Р.

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алексюк А.О.

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Санкт-Петербург   
2021

**ОГЛАВЛЕНИЕ**

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 3](#_Toc58595498)

[МЕТОД РЕШЕНИЯ 3](#_Toc58595499)

[ОБЫЧНАЯ ПРОГРАММА 4](#_Toc58595499)

[РАБОТА ОБЫЧНОЙ ПРОГРАММЫ 5](#_Toc58595499)

[ПРОГРАММА С ПОДПРОГРАММОЙ 5](#_Toc58595499)

[РАБОТА ПРОГРАММЫ С ПОДПРОГРАММОЙ 7](#_Toc58595499)

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**Вариант 15**

В массиве поменять местами максимальный и минимальный элементы

**МЕТОД РЕШЕНИЯ**

Программа последовательно перебирает все элементы массива, сравнивая их с текущими максимальным и минимальным значениями. До работы с циклом как условный максимум и минимум принимается значение первого элемента массива. Дальше идет проверка, если текущий элемент массива меньше минимума, то присваиваем минимуму значение этого элемента и запоминаем его адрес. Аналогично с максимумом, если текущий элемент больше текущего максимума, то присваиваем максимуму значение этого элемента и запоминаем его адрес. После обработки всех элементов массива по адресу минимального элемента записываем значение максимального, а по адресу максимального – значение минимального.

Псевдокод:

int[] array;

int max, min, minAddr, maxAddr;

max = array[0];

min = array[0];

for (i = 0; i < array.size; i++){

if (array[i] <= min) {

min = array[i];

minAddr = i;

}

if (array[i] >= max) {

max = array[i];

maxAddr = i;

}

}

array[minAddr] = max;

array[maxAddr] = min;

}

# **ОБЫЧНАЯ ПРОГРАММА**

На следующем рисунке можно ознакомиться с кодом программы. Заранее отмечу, что в программе рассматривается массив array со значениями:

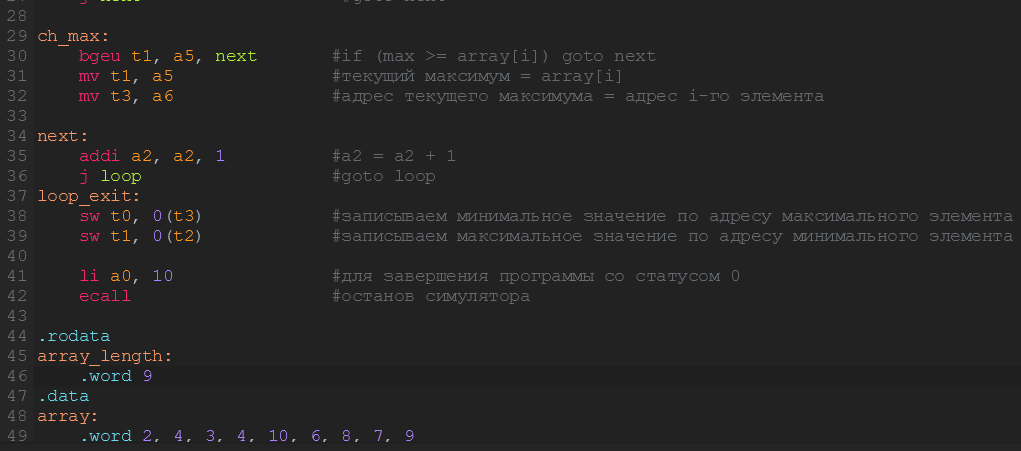
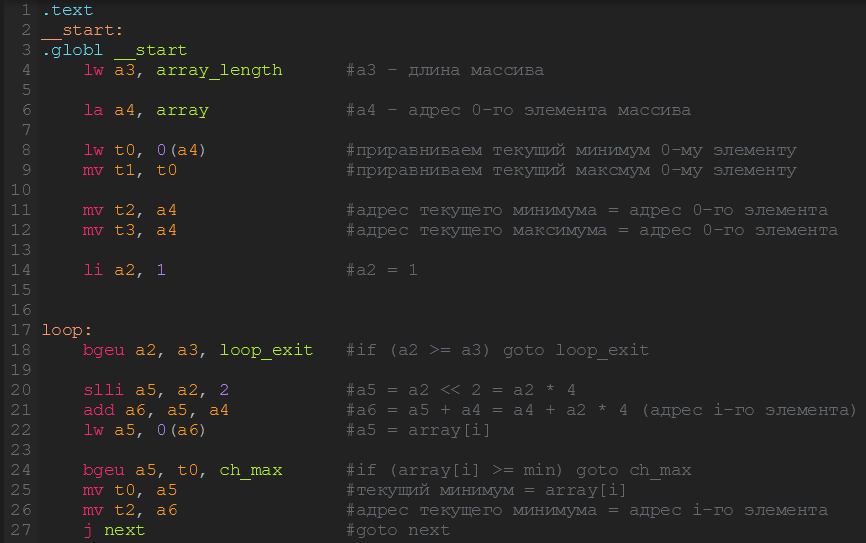
array[0] = 0x0216 = 210 array[5] = 0x0616 = 610

array[1] = 0x0416 = 410 array[6] = 0x0816 = 810

array[2] = 0x0316 = 310 array[7] = 0x0716 = 710

array[3] = 0x0416 = 410 array[8] = 0x0916 = 910

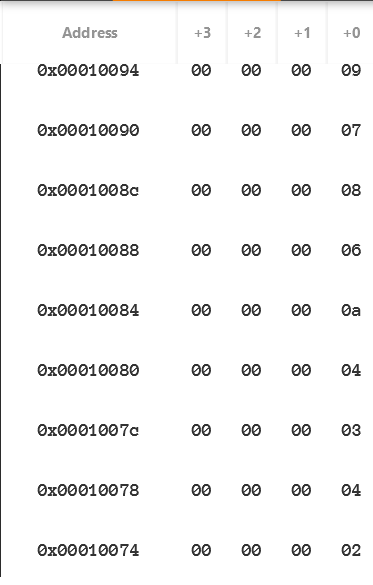
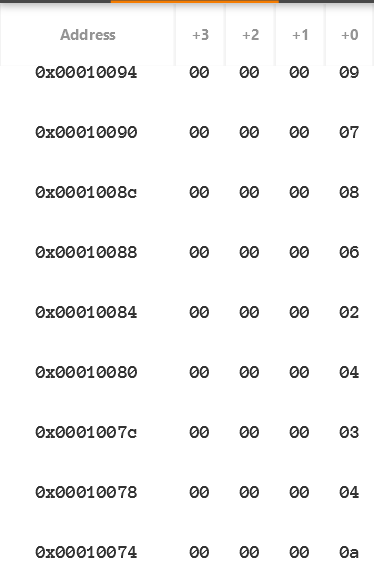
array[4] = 0x0A16 = 1010



**Рис. 1** Листинг обычной программы

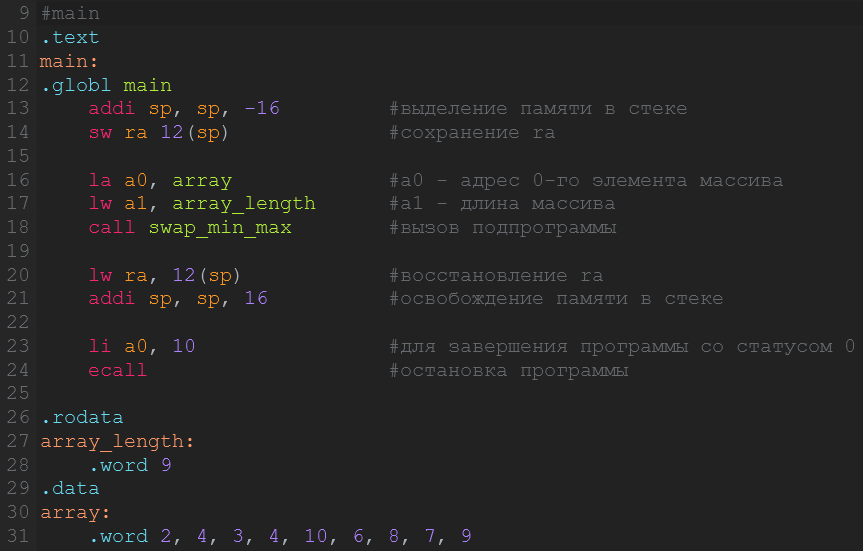
# **РАБОТА ОБЫЧНОЙ ПРОГРАММЫ**

**Заметка:** 0-й элемент массива расположен по адресу 0x00010074, остальные элементы - выше

**Рис. 2** До и после

# **ПРОГРАММА С ПОДПРОГРАММОЙ**



**Рис. 3** Основная программа

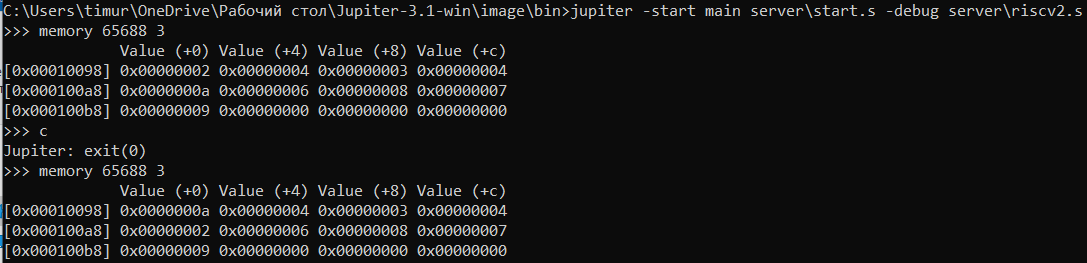
# 

**Рис. 4** Подпрограмма

Обратим внимание на **рисунок 3**. На нем видно, что перед вызовом подпрограммы swap\_min\_max мы сохраняем текущий адрес возврата **ra.** Это связано с тем, что **ra** подпрограммыотличается от **ra** основной программы. Из-за этого, после исполнения swap\_min\_max и возвращения в основное тело программы **ra** оказывается некорректным. Для того, чтобы устранить эту ошибку, мы выполняем предварительное сохранение адреса возврата и после исполнения подпрограммы восстанавливаем его.

# **РАБОТА ПРОГРАММЫ C ПОДПРОГРАММОЙ**

Для того, чтобы проверить работу двух программ, нужно запустить исполнение через консоль:



**Рис. 5** Результаты

На рис. 5 можно увидеть 3 строчки: 0x00010098, 0x000100a8 и 0x000100b8.

В строчке 0x00010098 в столбце Value(+0) расположен 0-ой элемент массива, на той же строчке в столбце Value(+4) – 1-ый элемент, соответственно в строчке 0x000100b8 в столбце Value(+0) – 9-ый .

Как видно по рисунку, перестановка максимального и минимального элементов массива была успешна.