Санкт-Петербургский политехнический университет Петра Великого

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе № 3**

**Дисциплина:** Низкоуровневое программирование

**Тема:** RISC-V

Выполнил студент гр. 3530901/90003 Зотов М. С.

(подпись)

Преподаватель Алексюк А.О.

(подпись)

“ ” 2021 г.

Санкт-Петербург

2021

**Оглавление**

[**1. ТЗ** 3](#_Toc67397966)

[**2. Метод решения** 3](#_Toc67397967)

[**3. Обычная программа** 3](#_Toc67397968)

[**4. Программа с подпрограммой** 4](#_Toc67397969)

[**5. Результаты работы обычной программы** 5](#_Toc67397970)

[**6. Результаты программы с подпрограммой** 6](#_Toc67397971)

# **1. ТЗ**

Реверс массива чисел.

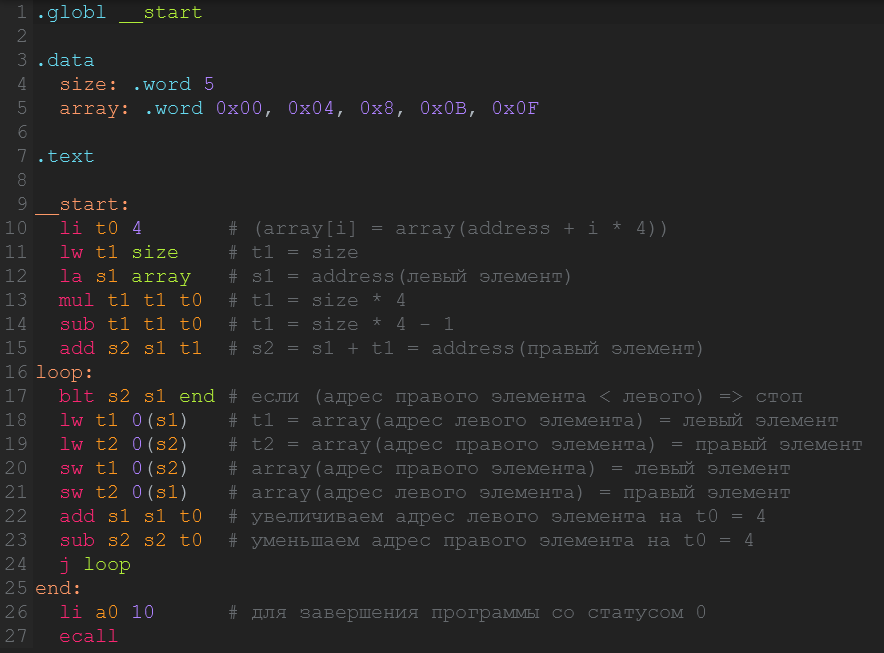
# **2. Метод решения**

Алгоритм решения:

1. В начальный момент у нас есть элементы массива **array**, расположенные в памяти друг за другом. Также есть **2** регистра, содержащие адреса крайнего левого непереставленного элемента (**address1**) и крайнего правого непереставленного элемента (**address2**). Кроме того, в определенный регистр **temp** мы заносим значение **4**, так как адрес каждого элемента массива кодируется **4** битами.
2. В ходе исполнения программы на каждой итерации цикла **loop** мы получаем левый элемент по адресу в **address1** и правый элемент по адресу **address2**, затем заносим их в специальные регистры **temp1** и **temp2** соответственно. После этого в память по адресу **address1**/**address2** заносим значение **temp2**/**temp1**. Далее увеличиваем/уменьшаем **address1**/ **address2** на величину **temp = 4**.
3. Цикл **loop** прекращает своё выполнение после того, как получается, что **address2** < **address1**.

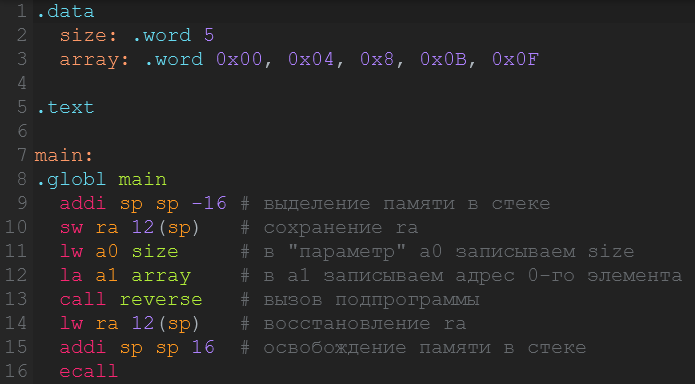
# **3. Обычная программа**

На следующей странице можно ознакомиться с кодом программы:

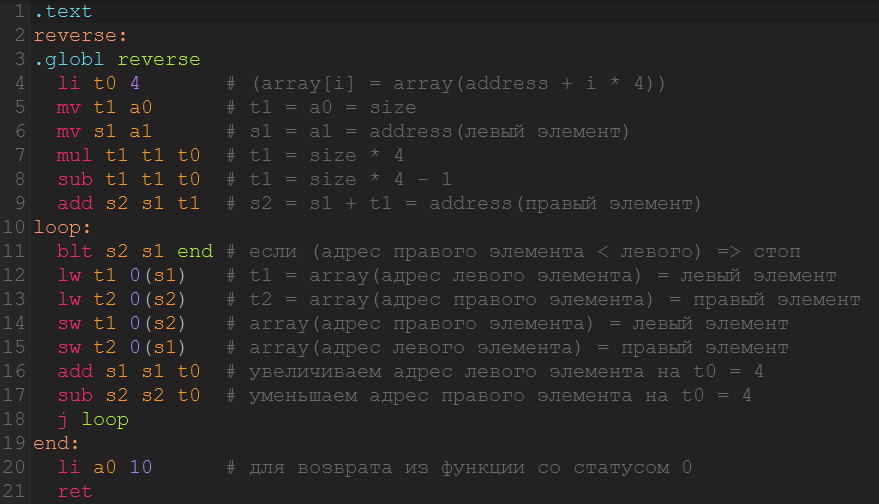
****

**Рис. 1** Листинг обычной программы

# **4. Программа с подпрограммой**



**Рис. 2** Основная программа



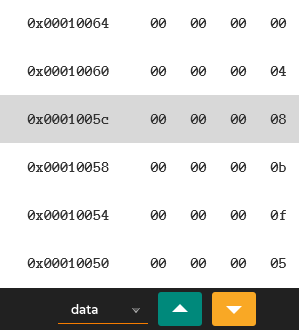
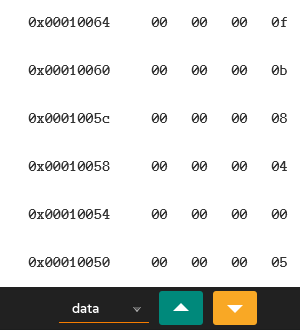
**Рис. 3** Подпрограмма

Отдельного внимания заслуживает **рис. 2**. На нем видно, что перед вызовом **reverse** мы выполняем сохранение текущего адреса возврата **ra**. Это связано с тем, что, находясь в теле **main**, у нас значение **ra** одно, а при вызове **reverse** мы получаем другой **ra**. После исполнения **reverse** и возвращения в **main** **pa** оказывается равным тому значению, которое **pa** принял в **reverse**. Для того, чтобы после выхода из **main** **pa** был корректным, мы выполняем сохранение адреса возврата и после исполнения подпрограммы восстанавливаем исходный адрес.

# **5. Результаты работы обычной программы**

С результатами можно ознакомиться на следующей странице.

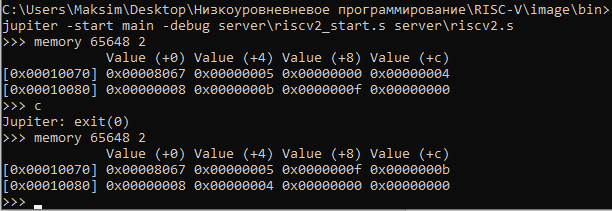
**Заметка**: по адресу **0x00010050** расположен размер массива, все, что выше – элементы массива

****

**Рис. 4** До **Рис. 5** После

# **6. Результаты программы с подпрограммой**

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



**Рис. 6** Результаты

На **рис. 6** можно увидеть **2** строчки: **0x00010070** и **0x00010080**.

В строке **0x00010070** в столбце **Value (+8)** расположен **0-ой** элемент массива, а в столбце **Value (+с)** – **1-ый** элемент.

В строке **0x00010080** в столбце **Value (+0)** расположен **2-ой** элемент массива, в столбце **Value (+4)** – **3-ый** элемент, в столбце **Value (+8)** – **4-ый** элемент.

Как видно по **рис. 6**, реверс массива был выполнен успешно.