Санкт-Петербургский политехнический университет Петра Великого

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе №3**

**Дисциплина**: Низкоуровневое программирование

**Тема:** Программирование на RISC-V

Выполнил студент гр. 3530901/90003 В.М. Ковалевский

(подпись)

Преподаватель А.О. Алексюк

(подпись)

“ ” 2021 г.

Санкт-Петербург

2021

**СОДЕРЖАНИЕ**

[1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 3](#_Toc1562003708)

[2. МЕТОД РЕШЕНИЯ 4](#_Toc119605367)

[3.1. РЕШЕНИЕ В ВИДЕ ПРОГРАММЫ 5](#_Toc1353215138)

[3.2. РАБОТА ПРОГРАММЫ 8](#_Toc1883074628)

[4.1. РЕШЕНИЕ В ВИДЕ ПОДПРОГРАММЫ 10](#_Toc206388284)

[4.2. РАБОТА ПОДПРОГРАММЫ 13](#_Toc411927887)

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Написать программу, реализующую слияние двух отсортированных массивов, для RISC-V (в виде программы и подпрограммы) .

# МЕТОД РЕШЕНИЯ

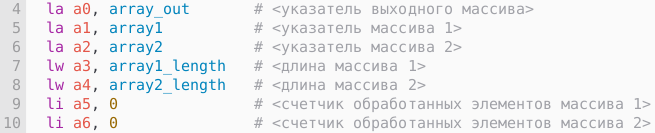
У нас есть два входных массива и один выходной массив. Изначально указатели установлены на их начала. Затем в цикле мы сравниваем очередные элементы массива 1 и массива 2. Если элемент массива 1 меньше элемента массива 2 - записываем элемент массива 1, увеличиваем указатель массива 1 и указатель выходного массива. И наоборот. Повторяем в цикле, считая количество обработанных элементов, пока не останется необработанных элементов.

# 3.1. РЕШЕНИЕ В ВИДЕ ПРОГРАММЫ

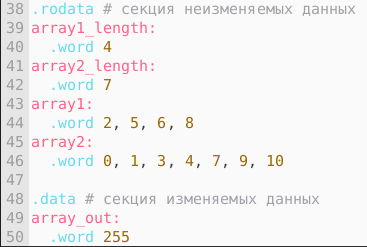


**.text** - указание ассемблеру размещать последующие инструкции в секции кода.

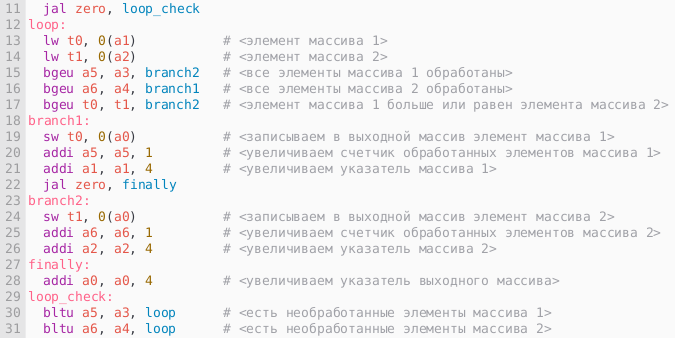
В следующей строке определяется метка “**start:**” - отсюда начинает выполняться наша программа.



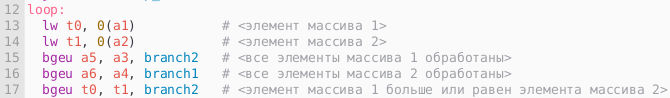
Далее следуют псевдоинструкции установки значений регистров a0 - a6. Псевдоинструкция транслируется ассемблером в последовательность инструкций (instruction) системы команд RISC-V, обеспечивающую выполнение требуемого действия.



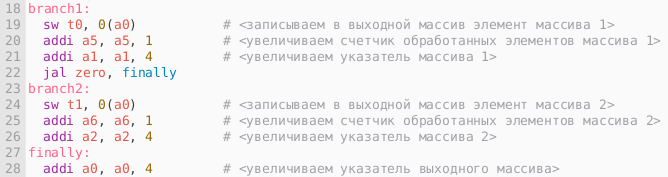
Сами данные определяются в конце программы. **.word** означает, что мы используем 32-битные слова (4 байта). То есть они занимают **4** восьмибитных секций. Это факт будет отражен далее.



Оператор **jal zero, loop\_check** осуществляет безусловный переход к метке **loop\_check**, где мы проверяем, остались ли у нас необработанные элементы в массивах. Цикл организован через оператор **bltu aX, aY, loop** который осуществляет переход, если **aX** меньше **aY**, в нашем случае мы проверяем, не вышли ли мы за границу массива.



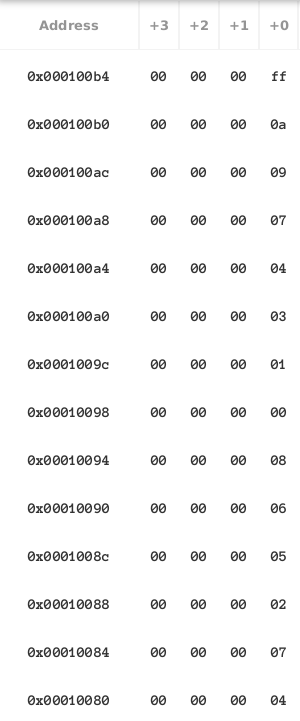
В метке **loop** командой **lw t0, 0(a1)** мы записываем значение ячейки по адресу **a1** в регистр **t1**. Далее идут операторы типа **bgeu aX, aY, branchZ** - если **aX** больше или равен **aY** переходим к метке **branchZ**. Собственно здесь мы проверяем, есть ли необработанные элементы в каждом из массивов и выбираем наименьший элемент.



Для каждой из веток: сохраняем элемент массива в следующую ячейку выходного массива, увеличиваем указатель и счетчик обработанных элементов. Стоит отметить, что указатель мы увеличиваем на 4, а не на 1, так как элементы занимают 4 байта (**.word**).

В **finally** мы увеличиваем указатель выходного массива - готовимся к записи следующего элемента.

# 3.2. РАБОТА ПРОГРАММЫ



Начало выходного массива

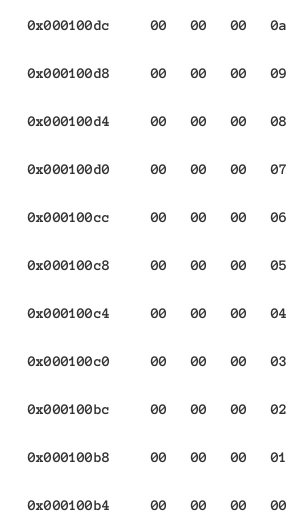
Длины массивов

Массив 2

Массив 1

*Значения ячеек до запуска программы (исходные данные)*

0x00010080 - длина массива 1, 0x00010084 - длина массива 2, 0х00010088 - начало массива 1, 0x0010098 - начало массива 2, 0x000100b4 - начало выходного массива.



Начало выходного массива

*Выходной массив*

Как видно - мы получили отсортированный массив. (0a это 10)

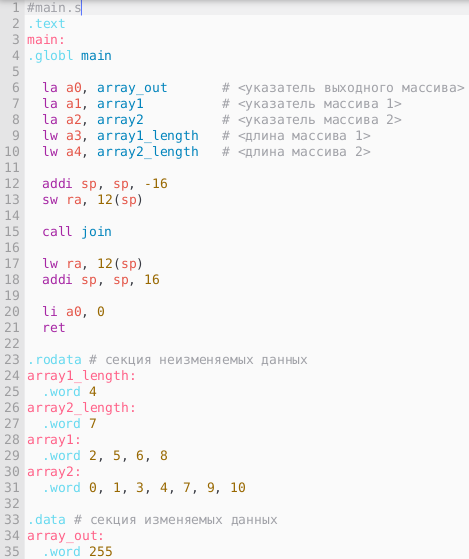
Данные располагаются с ячейки, следующей за 0x001007c, сразу за последней инструкцией программы.

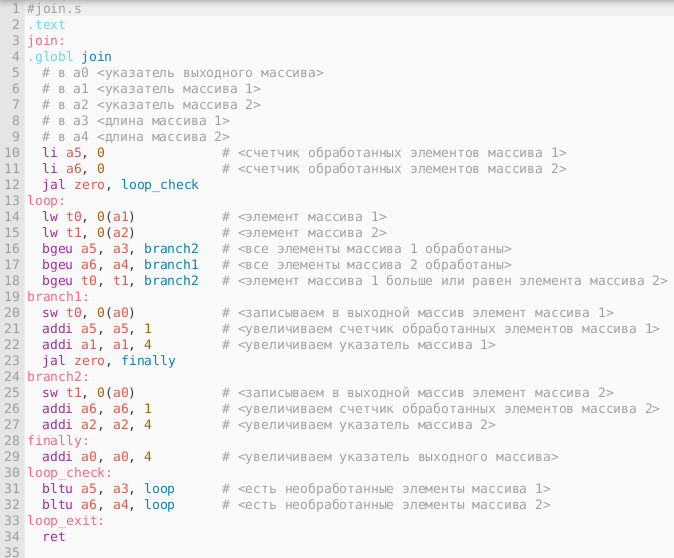


Сама программа начинается с адреса 0х0010000.

# 4.1. РЕШЕНИЕ В ВИДЕ ПОДПРОГРАММЫ







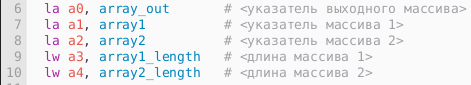
Сначала о тестирующей программе.



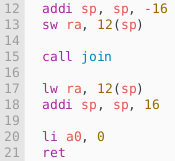
Выполнение начинается с **setup.s**. Здесь мы вызываем подпрограмму **main** используя команду **call**. После завершения подпрограммы исполняется код после метки **finish**, где мы завершаем работу программы.

**ret** - возврат из подпрограммы.

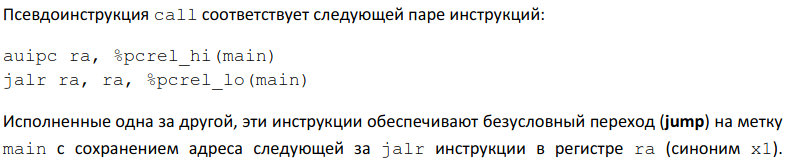
Теперь перейдем к подпрограмме **main**.

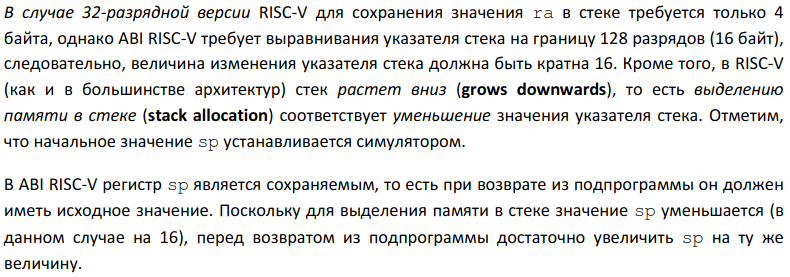


Здесь мы передаем аргументы перед вызовом подпрограммы.



Почему этот вызов подпрограммы отличается от вызова подпрограммы в **setup.h**? Потому что уже находимся в подпрограмме, а регистр с кодом возврата **(ra)** у нас один. Поэтому перед тем, как мы вызовем еще одну подпрограмму, нам нужно где-то сохранить этот код.





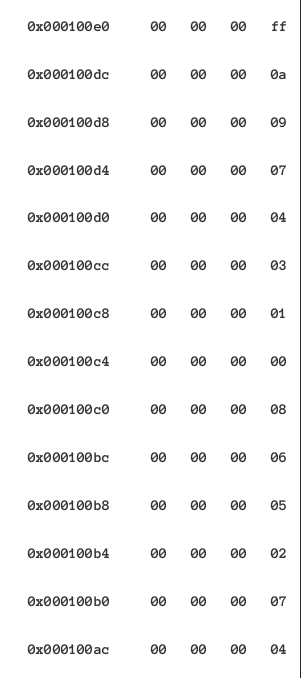
*Пояснение к псевдоинструкции* ***call***

Программу из предыдущего пункта мы оформили как подпрограмму. Разница в том, что все значения устанавливаются до ее вызова. Все остальное - без изменений.

# 4.2. РАБОТА ПОДПРОГРАММЫ

Все данные точно так же размещены в конце программы, начиная с ячейки 0x000100ac. Выходной массив начинается с ячейки 0x000100e0.

Начало выходного массива

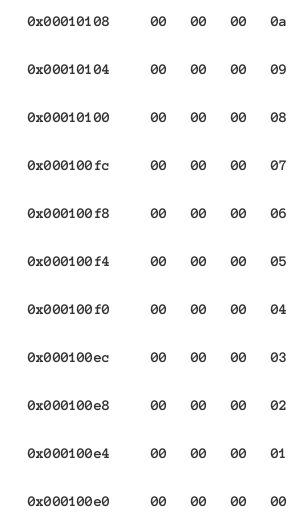


Массив 2

Длины массивов

Массив 1

*Значения ячеек до запуска программы (исходные данные)*



Начало выходного массива

*Значения ячеек после запуска программы (выходной массив)*