Санкт-Петербургский политехнический университет Петра Великого

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №3**

**Дисциплина**: Низкоуровневое программирование

**Тема**: Программирование на языке RiscV

Выполнил студент гр. 3530901/90003 Ремнева Е.Е.

(подпись)

Преподаватель Алексюк А.О.

(подпись)

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Санкт-Петербург

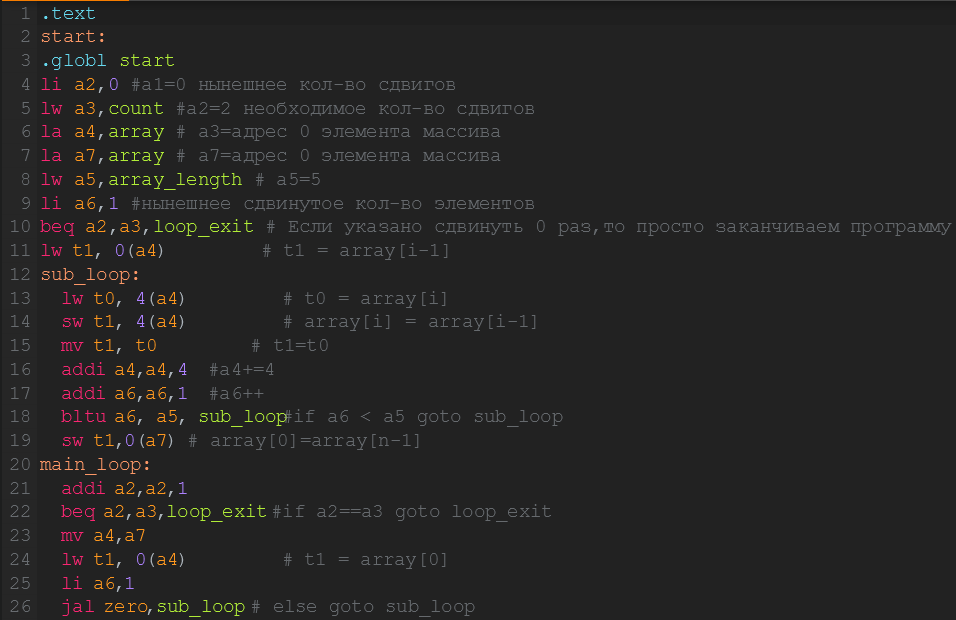
2021

**Вариант:** 17

**Постановка задания**: циклический сдвиг массива на заданное количество разрядов вправо.

**Метод решения:** Обход массива начинается с нулевого элемента, запоминается следующее значение ячейки массива в регистре общего назначения, а значение текущей ячейки записывается в следующую ячейку. Этот процесс продолжается до тех пор, пока не будет достигнута предпоследняя ячейка массива, тогда происходит запись последнего элемента массива в нулевую ячейку. И так N раз, пока весь массив не сдвинется на заданное количество разрядов.

**Программа на языке ассемблера Risc-V:**



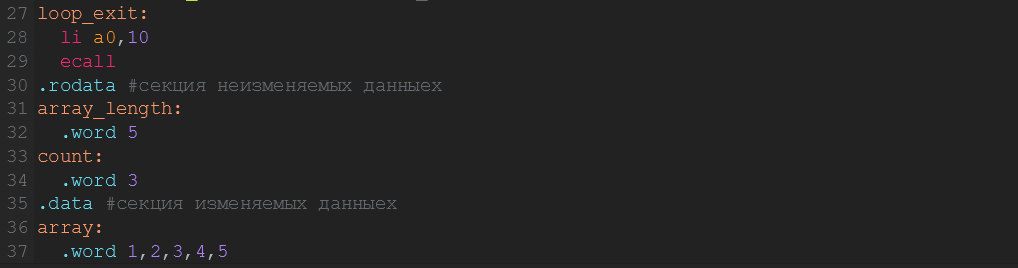


Рис.1.1. Листинг программы.

Сперва идет обозначение директивы .text, которая обозначает, что последующие инструкции будут размещаться в секции кода “text”, то есть начиная с адреса 0x00010008. Метка start является меткой начала программы.

Дальше следует директива. globl, которая указывает, что последующая последовательность символов является экспортируемой.

* С помощью псевдоинструкции lw (записать данные из одного адреса памяти в другой) записываем в регистр a5 длину массива.
* Вызывая инструкцию la (записать адрес одной ячейки в другой адрес памяти), мы записываем в a4 адрес 0 по счету элемента массива. В данном случае 0x00010048.

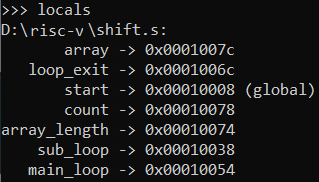


Рис.1.2. Адреса всех меток.

* а1 -нынешнее количество сдвигов
* а2 – необходимое количество сдвигов
* а3 – адрес нулевого элемента массива
* а7 адрес нулевого элемента массива
* а6 – нынешнее сдвинутое количество элементов, отсчет начинается с 1 по причине того, что за цикл должны будут сдвинуться на 1 разряд вправо все элементы, кроме последнего.
* beq-проверка: если указано сдвинуть массив на 0, то мы не выполняем работу, иначе-начинаем
* Первый цикл направлен на выполнение сдвига на 1 разряд массива, выполняется как раз-таки запись следующего и нынешнего элементов в ячейки памяти, сначала запоминается нынешний, далее запоминается последующий, и на место последующего записывается предыдущий элемент массива, переход в следующую ячейку
* Далее прибавляется 1 к количеству сдвинутых разрядов, и проверка: если количество сдвинутых разрядов еще не равно длине массива, то заново выполняем цикл sub\_loop, иначе в нулевую ячейку массива кладем последний элемент
* После следует цикл main\_loop, нужный для того, чтобы контролировать, необходимо ли еще раз сдвинуть массив на один элемент или же массив сдвинут на заданное количество разрядов
* Увеличиваем на 1 количество выполненных сдвигов и сравниваем, достигло ли оно необходимого значения, если да, выходим из цикла и заканчиваем работу, в противном случае снова переходим к циклу sub\_loop

**Проверка работы:**

****

Рис.1.3. Запуск

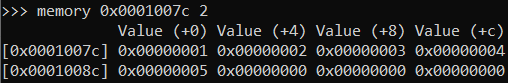


Рис.1.4. Начальные значения ячеек.

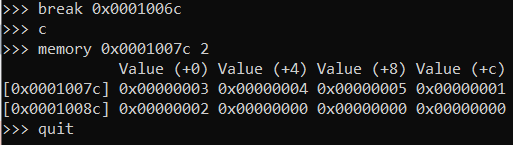


Рис.1.5. Значения после выполнения программы.

На рис.1.5. видно, что был установлен breakpoint на адрес инструкции завершения программы. Убеждаемся, что программа работает верно.

**Подпрограмма с вызывающей программой в соответствие с ABI:**

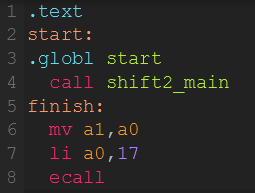


Рис.1.6. Листинг программы(shift2\_setup), вызывающей основную программу (shift2\_main).

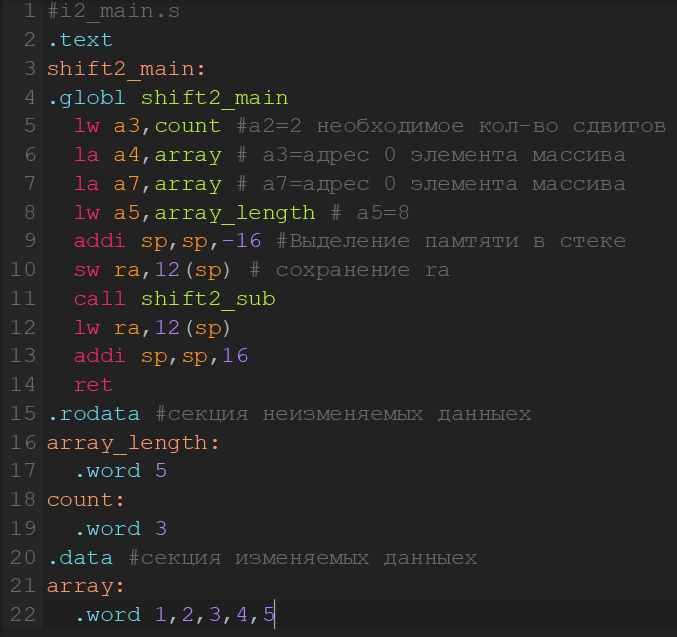


Рис.1.7. Листинг программы shift 2\_main.

В данному случае длина массива и адрес i -ого элемента хранятся в регистрах a1, a0 соответственно. Также до перехода на метку shift2\_sub нужно сохранить значение регистра ra (адреса, куда требуется вернуться после выполнения подпрограммы), если этого не сделать, то возвращаемый адрес будет адресом инструкции ret и программа просто зациклиться. Чтобы этого избежать мы сохраняем значение регистра ra в стэк (регистр sp), вместимость стэка устанавливаем-16 байт, в инструкции указано отрицательное число, так как стэк растет вниз. После возвращения из подпрограммы мы достаем значение регистра ra и удаляем стэк.

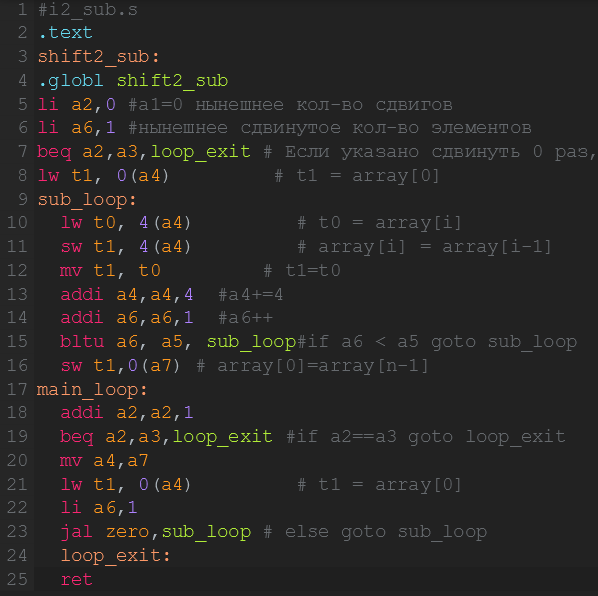


Рис.1.8. Листинг подпрограммы shift2\_sub.s.

Здесь в отличие от shift.s изменены только регистры под длину массива и адрес его i-ого элемента.

**Проверка правильности работы:**

****

Рис.2.1. Запуск программ.

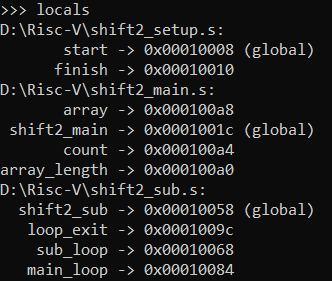


Рис.2.2. Адреса всех меток.

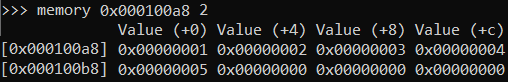


Рис.2.3. Значения до выполнения.

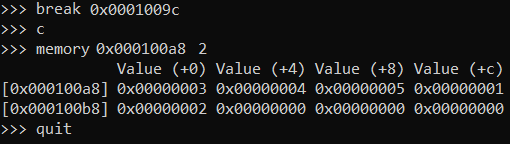


Рис.2.4. Значения после и успешное выполнение программы.