

Санкт-Петербургский политехнический университет Петра Великого
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе № 3

Дисциплина: Низкоуровневое программирование

Тема: Программирование RISC-V

Выполнил студент гр. 3530901/10005

Кузичева П. Д.

Преподаватель

Коренев Д. А.

“ ” _____ 2022 г.

Санкт-Петербург

Оглавление

Техническое задание	3
Метод решения.....	3
Руководство программисту	3
Реализация программы	4
Результат работы программы	4
Реализация подпрограммы и тестовой программы	5
Результат работы подпрограммы и тестовой программы.....	6
Вывод	7

Техническое задание

Написать программу на языке ассемблера RISC-V, которая реализует реверс исходного массива. Отладить ее в симуляторе Jupiter. Необходимо также выделить основной функционал в подпрограмму и написать для нее тестовую программу.

Метод решения

Реверс массива осуществляется посредством перестановки элементов. При реверсе первый элемент массива становится последним, а последний – первым, второй – предпоследним, а предпоследний – вторым и так далее. В случае, когда количество элементов массива нечетное число, средний элемент, то есть стоящий на середине массива, остается на своем месте. Перестановок надо сделать в два раза меньше, чем длина массива. Например, рассмотрим массив [5, 8, 13, 3, 59]. Сначала будут переставлены элементы 5 и 59. После этой итерации получится массив [59, 8, 13, 3, 5]. После будут переставлены на другие места элементы 8 и 3, и получится массив [59, 3, 13, 8, 5]. Элемент 13 игнорируется, так как не имеет пары и находится на середине массива.

Руководство программисту

Начальные данные к программе: адрес нулевого элемента массива (и соответственно сам массив) и его длина. В реализации без подпрограммы адрес и длина хранятся в регистрах a4 и a3 соответственно. В реализации через подпрограмму адреса остаются прежними.

Реализация программы

```
1 .text
2 __start:
3 .globl __start
4
5     la a3, array_length
6     lw a3, 0(a3) # загрузка длины массива
7     la a4, array # адрес нулевого элемента
8     slli a5, a3, 2 # адрес последнего элемента будет храниться в ячейке a5(a5 = a3*4)
9     add a5, a4, a5 # a5 = a4 + a3 * 4
10    addi a5, a5, -4 # коррекция адреса
11    # a5 - адрес последнего элемента
12    addi a2, a2, 1 # const = 1
13
14 loop:
15     lw t1, 0(a4) # меняем местами элементы
16     lw t0, 0(a5)
17     sw t1, 0(a5)
18     sw t0, 0(a4)
19     addi a5, a5, -4 # переходим на следующий элемент в конце
20     addi a4, a4, 4 # переходим на следующий элемент в начале
21     addi a3, a3, -2 # отнимаем два от длины массива
22     bgeu a2, a3, loop_exit #if( a2(1) >= a3(длина массива) goto loop_exit
23     jal zero, loop # goto loop
24 loop_exit:
25
26 finish:
27     li a0, 10 # x10 = 10
28     li a1, 0 # x11 = 0
29     ecall # ecall при значении x10 = 10 => останов симулятора
30
31 .rodata # неизменяемые данные
32     array_length: # длина массива
33     .word 5
34
35 .data # изменяемые данные
36     array: # массив
37     .word 1, 2, 3, 4, 5
```

Результат работы программы

Входные данные – массив [1, 2, 3, 4, 5]. Ожидаемый результат - [5, 4, 3, 2, 1].

0x00010070	00	00	00	05
0x0001006c	00	00	00	04
0x00010068	00	00	00	03
0x00010064	00	00	00	02
0x00010060	00	00	00	01

Исходные данные

0x00010070	00	00	00	01
0x0001006c	00	00	00	02
0x00010068	00	00	00	03
0x00010064	00	00	00	04
0x00010060	00	00	00	05

После запуска программы

Результат совпал с ожидаемым.

Реализация подпрограммы и тестовой программы

Тестовая программа

```

1 .text
2 __start:
3 .globl __start
4
5     la a3, array_length
6     lw a3, 0(a3) # загрузка длины массива
7
8     la a4, array # адрес нулевого элемента
9
10    call reverse # вызов подпрограммы
11
12    li a0, 10 # запись 10 в a0
13    ecall # остановка программы
14
15 .rodata # неизменяемые данные
16     array_length: # длина массива
17     .word 5
18
19 .data # изменяемые данные
20     array: # массив
21     .word 1, 2, 3, 4, 5

```

Подпрограмма

```
1 .text
2 reverse:
3 .globl reverse
4
5     slli a5, a3, 2 # адрес последнего элемента будет храниться в ячейке a5(a5 = a3*4)
6     add a5, a4, a5 # a5 = a4 + a3 * 4
7     addi a5, a5, -4 # коррекция адреса
8     # a5 - адрес последнего элемента
9     addi a2, a2, 1 # const = 1
10
11 loop:
12     lw t1, 0(a4) # меняем местами элементы
13     lw t0, 0(a5)
14     sw t1, 0(a5)
15     sw t0, 0(a4)
16
17     addi a5, a5, -4 # переходим на следующий элемент в конце
18     addi a4, a4, 4 # переходим на следующий элемент в начале
19
20     addi a3, a3, -2 # отнимаем два от длины массива
21     bgeu a2, a3, loop_exit #if( a2(1) >= a3(длина массива) goto loop_exit
22     jal zero, loop # goto loop
23 loop_exit:
24
25 finish:
26     ret
```

Результат работы подпрограммы и тестовой программы

Входные данные – массив [1, 2, 3, 4, 5]. Ожидаемый результат - [5, 4, 3, 2, 1].

0x00010078	00	00	00	05
0x00010074	00	00	00	04
0x00010070	00	00	00	03
0x0001006c	00	00	00	02
0x00010068	00	00	00	01

Исходные данные

0x00010078	00	00	00	01
0x00010074	00	00	00	02
0x00010070	00	00	00	03
0x0001006c	00	00	00	04
0x00010068	00	00	00	05

После запуска программы

Результат совпал с ожидаемым.

Вывод

В ходе выполнения лабораторной работы была разработана программа на языке ассемблера RISC-V, реализующая реверс массива. Программа была отлажена в симуляторе Jupiter. Результаты работы программы совпали с ожидаемыми.