

Санкт-Петербургский политехнический университет Петра Великого  
Кафедра компьютерных систем и программных технологий

## **Отчёт по лабораторной работе №3**

Дисциплина: Низкоуровневое программирование

Тема: Программирование RISC-V

Выполнил студент гр. 3530901/10005

\_\_\_\_\_ Воронов И. В.  
(подпись)

Преподаватель

\_\_\_\_\_ Коренев Д. А.  
(подпись)

“    ” \_\_\_\_\_ 2022 г.

Санкт-Петербург

2022

# 1.Техническое задание

Расчет заданного члена ряда Фибоначчи. Написать две программы: в Initial Orders 1 и Initial Orders 2.

## 2.Метод решения

Число ряда Фибоначчи вычисляется по формуле  $F_n = F_{n-1} + F_{n-2}$ , при этом  $F_0 = 0$ ,  $F_1 = 1$ .

Программа должна быть реализована в виде цикла, который проверяет достиг ли счётчик нужного порядкового номера числа  $n$ .

Для хранения чисел Фибоначчи используются две переменные, хранящие значения  $F_k$  и  $F_{k+1}$ . В теле цикла в большую переменную записывается сумма прежних значений, а в меньшую — бывшее значение  $F_{k+1}$ . После этого программа возвращается к проверке условия выхода из цикла.

Когда происходит выход из цикла, программа записывает значение из меньшей из двух переменных Фибоначчи в ячейку ответа.

## 3.Руководство программисту

Начальные данные к программе:порядковый номер искомого числа Фибоначчи. В реализации без подпрограммы  $n$  хранится в регистре  $a3$ , а адрес результата — в  $a6$  . В реализации через подпрограмму  $a0$  и  $a3$  соответственно.

## 4.Реализация программы 1

```
1 .text
2 __start:
3 .globl __start
4     li a2, 0 # задаётся первоначальное значение счётчику
5     la a3, number # считывается адрес порядкового номера
6     lw a3, 0(a3) # в переменную считывается порядковый номер
7     la a4, result # считывается адрес результата
8     li t0, 0
9     li t1, 1
10
11 loop:
12     bgeu a2, a3, loop_exit
13
14     addi t2, t1, 0
15
16     add t1, t1, t0
17     addi t0, t2, 0
18
19     addi a2, a2, 1
20     jal zero, loop
21
22 loop_exit:
23     sw t0, 0(a4)
24
25
26 finish:
27     li a0, 17
28     li a1, 0
29     ecall
30
31 .data
32 result:
33     .word 0
34
35 .rodata
36 number:
37     .word 10
```

## 5.Работа программы 1

При  $n = 10$  число Фибоначчи должно равняться 55.

0x00010054	0	0	0	55
0x00010050	0	0	0	10

data ▼

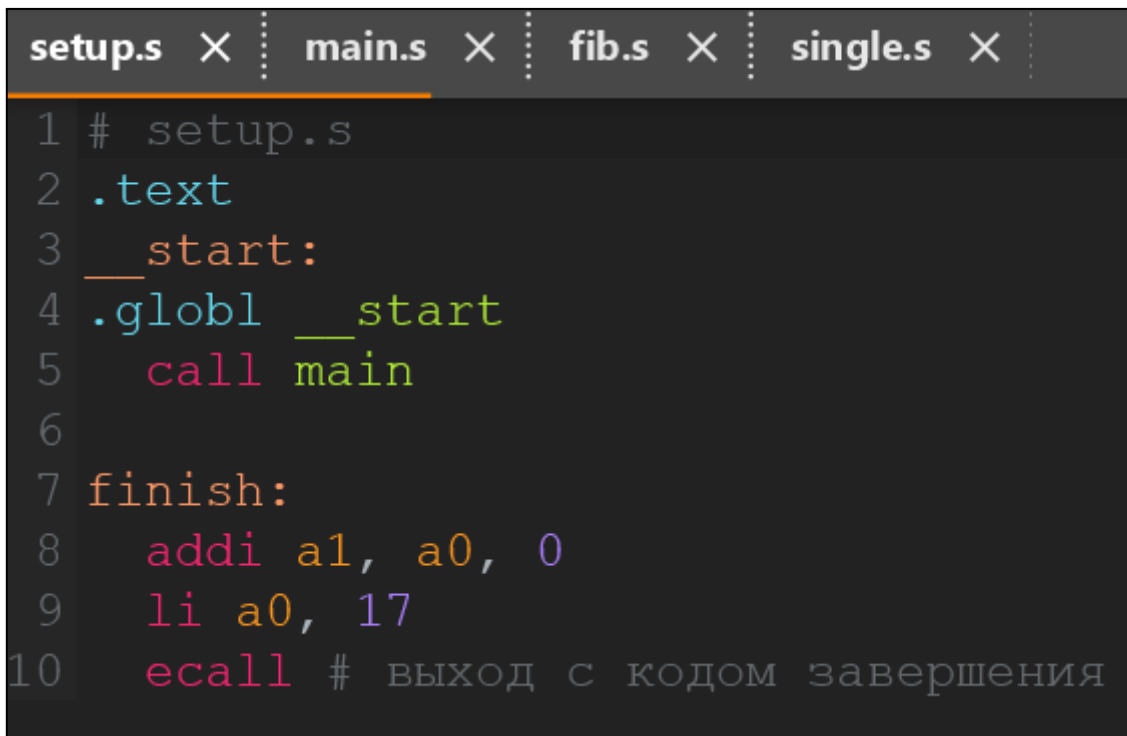
▲

▼

Результат.

Результат соответствовал ожидаемому.

## 6.Реализация программы 2 с подпрограммой



```
1 # setup.s
2 .text
3 __start:
4 .globl __start
5     call main
6
7 finish:
8     addi a1, a0, 0
9     li a0, 17
10    ecall # выход с кодом завершения
```

setup.s

```

1 # main.s
2 .text
3 main:
4 .globl main
5     addi sp, sp, -16 # выделение памяти в стеке
6     sw ra, 12(sp) # сохранение ra
7     la a0, number # считывается адрес порядкового номера
8     lw a0, 0(a0) # в переменную считывается порядковый номер
9
10    call fib # вызов подпрограммы fib
11
12    la t0, result # считывается адрес результата
13
14    sw a0, 0(t0)
15
16 finish:
17     lw ra, 12(sp) # восстановление ra
18     addi sp, sp, 16 # выделение памяти в стеке
19     li a0, 0
20     ret
21
22 .data
23 result:
24     .word 0 # результат
25
26 .rodata
27 number:
28     .word 10 # номер искомого числа

```

main.s

```

1 # fib.s
2 .text
3 fib:
4 .globl fib
5     li t3, 0
6     li t0, 0
7     li t1, 1
8
9 loop:
10    bgeu t3, a0, loop_exit
11
12    addi t2, t1, 0
13
14    add t1, t1, t0
15    addi t0, t2, 0
16
17    addi t3, t3, 1
18    jal zero, loop
19
20 loop_exit:
21    addi a0, t0, 0
22
23 finish:
24    ret

```

fib.s

## 7.Работа программы 2

При  $n = 10$  число Фибоначчи должно равняться 55.

0x00010084	0	0	0	55
0x00010080	0	0	0	10

Результат.

Полученный результат соответствовал ожидаемому.