Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра Информатика и вычислительная техника

**Отчёт по лабораторной работе**

**Дисциплина**: Низкоуровневое программирование

**Тема**: RISC-V

Выполнила студентка гр. 3530901/10005

Тучков Д.А

(подпись)

Преподаватель Сидорина Т.Л.

(подпись)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Санкт-Петербург

2022

**1. Техническое задание**

Вариант 2   
Сортировка методом вставки

**2. Метод решения**

**Сортировка вставками** имеет большую вычислительную сложность. Поэтому она эффективна на небольших наборах данных. Рекомендуется использовать этот метод на наборах размером до десятков элементов. Сортировка вставками эффективна на последовательностях с данными, которые уже частично отсортированы.

Первый элемент в массиве образует уже отсортированную последовательность. Сравниваем второй элемент с первым. Если порядок между ними нарушен, то первый элемент передвигается на одну позицию вправо. Теперь отсортированный массив состоит из двух элементов.

Далее, в течении каждой итерации, берем следующий элемент (третий, четвертый и т.д) и сравниваем его поочередно с другими элементами в уже отсортированном списке, **начиная с конца** этого списка. Если порядок между сравниваемыми элементами нарушен, то меняем их местами, если нет, то «вставка» нового элемента закончена, переходим к следующему.

Реализуем описанный выше алгоритм сортировки с помощью языка программирования Си. Напишем функцию void InsertionSort(int n, int mass[]), которая в качестве аргументов принимает: число элементов в массиве и сам массив

|  |  |
| --- | --- |
|  | void InsertionSort(int n, int mass[])  {      int newElement, location;        for (int i = 1; i < n; i++)      {          newElement = mass[i];          location = i - 1;          while(location >= 0 && mass[location] > newElement)          {              mass[location+1] = mass[location];              location = location - 1;          }          mass[location+1] = newElement;     }} |

**3. Руководство программисту**

Длинна массива: 10 (a3)

Адрес первого элемента (а4)

**4. Реализация программы 1**

.text # указатель на основную часть кода

\_\_start: #начало выполнения программы

.globl \_\_start

la a3, array\_length #длинна массива

lw a3, 0(a3)

#входим в цикл (инструкция выхода из цикла

#i>= длинне массива, если да то переходим к 49 строке

# если нет идем дальше по программе

la a4, array #a4 адрес первого элемента массива

#инструкция вычисления адреса массива

#значение вставляется в a5

li a2, 1 #i

loop1:

#t0 - записываем массив

#t1 - массив i-1

#t2 - j=i

bgeu a2, a3, loop\_exit

slli a5, a2, 2

add a5, a4, a5

# если элементы вставки > массива j переход к строке 45

# и увеличиваем значение i

lw t1, -4(a5)

lw t0, 0(a5)

#получаем значение массива j и записываем в t6

mv t2, a2

bgeu t0, t1, loop\_exit3

#пока элементы вставки > массива j если условие верно переходим в строку 43

loop2:

slli t3, t2, 2

#записываем в массив j+1 значение массива j

#делаем j-1 и записываем в t5 ссылку на массив j+1 для дальнейшей вставки числа

#переход в строку 26

add t3, a4, t3

lw t6, -4(t3)

beqz t2, loop\_exit2

bgeu t0, t6, loop\_exit2

#элемент для вставки вставляем в ячейку массива j+1 делаем i+1

#переход в строку 11

sw t6, 0(t3)

addi t2, t2, -1

mv t5, t3

jal zero, loop2

loop\_exit2:

sw t0, -4(t5)

#завершение работы программы

loop\_exit3:

addi a2, a2, 1

jal zero, loop1

#данные

loop\_exit:

finish:

li a0, 10

li a1, 0

ecall

.rodata

#длина

array\_length:

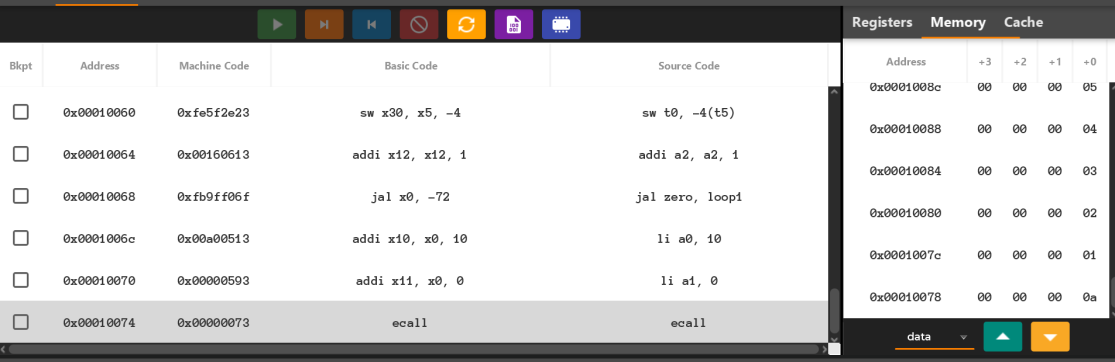
.word 10

.data

#неотсортированный массив

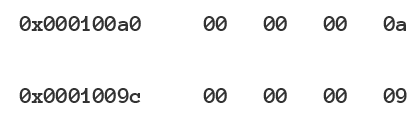
array:

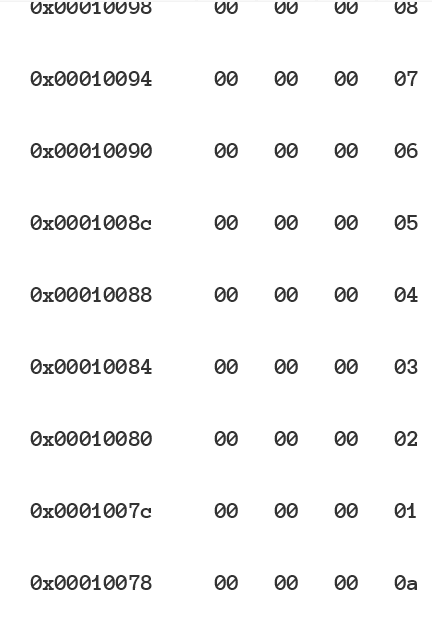
.word 5, 7, 8, 10, 4, 2, 3, 1, 6, 9



**5. Работа программы 1**

Массив = 5, 7, 8, 10, 4, 2, 3, 1, 6, 9





**6. Реализация программы 2 с подпрограммой**

#setup

.text

\_\_start:#начало выполнения программы

.globl \_\_start

call main

finish:

#a1=a0

mv a1, a0

#a0=17

li a0, 17

#завершение программы

Ecall

#main

.text# указатель на основную часть кода

main:

.globl main

#массив

la a0, array

la a1, array\_length

lw a1, 0(a1)

#выделение памяти в стеке

addi sp, sp, -16

#сохраняем

sw ra, 12(sp)

call vstavka

#восстанавливаем

lw ra, 12(sp)

#освобождаем память в стеке

addi sp, sp, 16

li a0, 0

ret

#длина массива

.rodata

array\_length:

.word 10

#массив

.data

array:

.word 5, 7, 8, 10, 4, 2, 3, 1, 6, 9

#vstavka

.text

vstavka:

.global vstavka

#a2=1

li a2, 1

#цикл

loop1:

#если i>длинны массива переход в строку 43

bgeu a2, a1, loop\_exit

slli a5, a2, 2

add a5, a0, a5

lw t1, -4(a5)

#элемент вставки

lw t0, 0(a5)

mv t2, a2

#если элемент вставки > массива j

bgeu t0, t1, loop\_exit3

loop2:

slli t3, t2, 2

add t3, a0, t3

#масссив j

lw t6, -4(t3)

beqz t2, loop\_exit2

bgeu t0, t6, loop\_exit2

#записываем массив j+1 в t6

sw t6, 0(t3)

#j-1

addi t2, t2, -1

#сохранение массива j+1

mv t5, t3

#переход в начало

jal zero, loop2

loop\_exit2:

sw t0, -4(t5)

loop\_exit3:

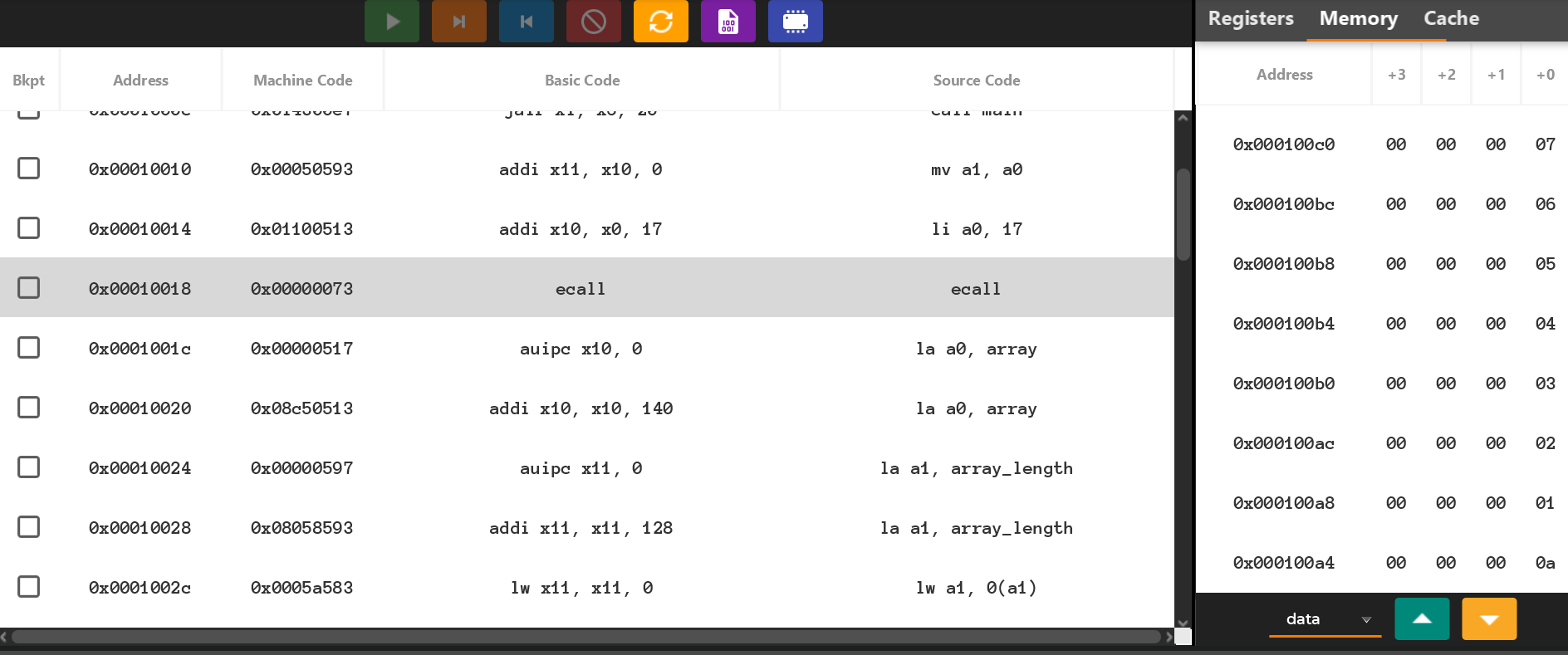
addi a2, a2, 1

#переход на строку 8

jal zero, loop1

loop\_exit:

ret



**7. Работа программы 2**

Массив = 5, 7, 8, 10, 4, 2, 3, 1, 6, 9

