Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчет по лабораторной работе №2**

**Дисциплина:** Низкоуровневое программирование

**Тема:** EDSAC

Выполнил

студент гр. 5130901/20001 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жуков И.Д.

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Коренев Д.А.

(подпись)

« 27 » октября 2023 г.

Санкт-Петербург   
2023

**Оглавление**

[**ПОСТАНОВКА ЗАДАЧИ** 3](#_Toc64232757)

[**ПРЕДЛОЖЕННОЕ РЕШЕНИЕ** 4](#_Toc64232758)

**ПОСТАНОВКА ЗАДАЧИ**

1. Разработать программу для EDSAC, которая реализует определение k-й порядковой статистики, и предполагающую загрузчик Initial Orders 1.
2. Выделить разработанную в первой задаче функциональность в замкнутую (closed) подпрограмму, разработать вызывающую ее тестовую программу. Использовать возможности загрузчика Initial Orders 2.

**ПРЕДЛОЖЕННОЕ РЕШЕНИЕ**

**Задача 1**

Для решения поставленной задачи необходим массив чисел и натуральное число k, меньшее или равное длине массива. В результате выполнения программы ответ будет содержать в регистре аккумулятора (далее асс). Для определения k-й порядковой статистики нужно отсортировать все числа в массиве. Далее добавить в асс k элемент в отсортированном массиве.

В качестве сортировки будем использовать сортировку «пузырьком». Наша программа будет состоять из 10 основных блоков.

**Первый блок**: initially requirements.

[0 - ZERO FOREVER]

[1 - RIGHT INDEX]

[2 - CURRENT INDEX]

[3 - PLACE FOR I ITEM]

[4 - FOR RUBBISH]

[INITIALLY REQUIREMENTS]

[31] T 114 S [LAST COMMAND]

[32] X 0 S [останов для отладки]

[33] T 4 S [запись аккумулятора в ячейку 4, обнуление аккумулятор]

[34] A 101 S [LEN] [загрузка в аккумулятор длины обрабатываемого массива]

[35] T 1 S [запись этого значения в ячейку 1

В первом блоке устанавливаем начальные условия для работы программы, очищаем асс. В ячейку 1 записываем длину массива. Далее это значения будет уменьшаться по ходу выполнения программы.

**Второй блок**: loop 1

Сортировка пузырьком имеет внешний и внутренний цикл. Loop 1 – внешний цикл for. Здесь уменьшается переменная len (right index). 39 команда присваивает записывает в 2 ячейку переменную, которая будет итерироваться во внутреннем цикле. В начале программы очищается асс.

[LOOP 1]

[36] T 4 S

[37] A 1 S

[38] S 100 S [1 CONST]

[39] U 2 S

[40] G 93 S [FINISH STEP]

[41] T 1 S

**Третий блок**: read i item

[FOR R1]

[42] A 102 S [R-BLANK]

[43] A 106 S [ADDR] [READ I ITEM]

[44] T 61 S [R1] [OPERATION: READ I ITEM]

….

[102] A 0 S [R-BLANK]

В этом блоке мы формируем команду для чтения i элемента списка.

R-BLANK – это шаблон для формирования команды чтения. В 106 ячейке хранится адрес 0 элемента массива.

**Четвертый и пятый блоки**: read i + 1 item

Эти блоки нужны для формирования инструкций чтения i + 1 элемента. Эти 2 блока выполняют одну и ту же функцию, однако r2 инструкция находиться в 63 ячейке, а r3 в 88.

[FOR R2]

[45] A 103 S [R-BLANK-2]

[46] A 106 S [ADDR] [READ I ITEM]

[47] T 63 S [R2] [OPERATION: READ I + 1 ITEM]

[FOR R3]

[48] A 103 S [R-BLANK-2]

[49] A 106 S [ADDR] [READ I ITEM]

[50] T 88 S [R3] [OPERATION: READ I + 1 ITEM]

…

[103] A 1 S [R-BLANK-2]

.

**Шестой и седьмой блоки**: write i/i + 1 item.

[FOR W1]

[51] A 104 S [W-BLANK]

[52] A 106 S [ADDR] [READ I ITEM]

[53] T 89 S [W1] [OPERATION: WRITE I ITEM]

[FOR W2]

[54] A 105 S [W-BLANK-2]

[55] A 106 S [ADDR] [READ I ITEM]

[56] T 91 S [W2] [OPERATION: WRITE I + 1 ITEM]

…

[104] T 0 S [W-BLANK]

[105] T 1 S [W-BLANK-2]

Эти блоки нужны для формирования инструкций записи i + 1 и i элементов. Эти 2 блока выполняют одну и ту же функцию, однако w1 инструкция находиться в 89 ячейке, а w2 в 91.

**Восьмой блок**: loop 2

[LOOP 2]

[57] A 2 S

[58] S 100 S [1 CONST]

[59] G 36 S [FIRST LOOP CONTINUATION]

[60] T 2 S

[61] A 0 S [R1 - DEFINITION] [OPERATION: READ I ITEM]

[62] T 3 S

[63] A 1 S [R2 - DEFINITION] [OPERATION: READ I + 1 ITEM]

[64] S 3 S

[65] G 87 S [SWAP]

Этот блок выступает в качестве внутреннего цикла for в сортировке. Сперва мы получаем переменную для итерирования (current index). Проверяем нужно ли продолжать цикл. Если да, продолжаем. Если не нужно продолжать, идет во внешний цикл. Current index мы записали в блоке loop 1 в 39 инструкции. Далее в 61 инструкции получаем i элемент. Эта инструкция сформирована в третьем блоке (r1). Записываем это значение в 3 ячейку (ячейка для временного хранения данных). Получаем i + 1 элемент в 63 инструкции и вычитаем из i + 1 элемента i элемент, находящийся в 3 ячейке. Если в асс отрицательное число, надо поменять i + 1 и i элементы местами. Иначе ничего не делать.

**Девятый блок**: swap

Это блок нужен для того, чтобы поменять местами i и i + 1 элементы местами. Сперва очищаем асс. Читаем i + 1 элемент. 88 инструкцию мы сформировали в 45-47 ячейках четвертого блока. Записываем это значение на место i элемента. Читаем из памяти i + 1 элемент. Этот элемент мы записали в 3 ячейку в восьмом блоке в 62 инструкции. Записываем i + 1 элемент на место i элемента. Возвращаем во внутренний цикл в десятый блок.

[SWAP]

[87] T 4 S

[88] A 1 S [R3 - DEFINITION] [OPERATION: READ I + 1 ITEM]

[89] T 0 S [W1 - DEFINITION] [WRITE I + 1 ITEM TO I INDEX]

[90] A 3 S [READ I ITEM]

[91] T 1 S [W2 - DEFINITION] [WRITE I ITEM TO I + 1 INDEX]

[92] E 66 S [LOOP 2 CONTINIATION]

**Десятый блок**: loop 2 continuation

Этот блок нужен для модификации инструкций чтения и записи i и i + 1 элементов. Инструкции r1, w1 будут теперь работать с i + 1 элементом, а инструкции r2, r3 и w2 будут работать с i + 2 элементом. В 86 инструкции идет безусловный переход на начало внутреннего цикла loop 2.

[LOOP 2 CONTINUATION]

[66] A 100 S [1 CONST]

[67] L 0 L

[68] A 61 S [R1]

[69] T 61 S [R1]

[70] A 100 S [1 CONST]

[71] L 0 L

[72] A 63 S [R2]

[73] T 63 S [R2]

[74] A 100 S [1 CONST]

[75] L 0 L

[76] A 88 S [R3]

[77] T 88 S [R3]

[78] A 100 S [1 CONST]

[79] L 0 L

[80] A 89 S [W1]

[81] T 89 S [W1]

[82] A 100 S [1 CONST]

[83] L 0 L

[84] A 91 S [W2]

[85] T 91 S [W2]

[86] E 57 S [LOOP 2]

**Одиннадцатый блок**: finish step

В этом блоке в асс записывает ответ. К моменту выполнения этого блока список отсортирован. Число k задается инструкцией 108. Натуральное число, находящееся между P и S, это и есть желаемое k. Ответ записывается в асс благодаря 107 инструкции.

[FINISH STEP]

[93] T 4 S

[94] A 108 S [K]

[95] L 0 L

[96] A 106 S [ADDR]

[97] A 102 S [R-BLANK]

[98] T 107 S [LAST - RESULT]

[99] E 107 S [LAST - RESULT]

[100] P 0 L [1 CONST]

[101] P 2 L [LEN]

[102] A 0 S [R-BLANK]

[103] A 1 S [R-BLANK-2]

[104] T 0 S [W-BLANK]

[105] T 1 S [W-BLANK-2]

[106] P 109 S [0'S] [ADDR]

[107] A 0 S [LAST - RESULT]

[108] P 2 S [K = 4] [I.E. 5 ITEM]

[109] P 5 S [0'S] [10]

[110] P 4 S [1'S] [8]

[111] P 3 S [2'S] [6]

[112] P 2 S [3'S] [4]

[113] P 1 S [4'S] [2]

**Задача 2**

Для решений задачи необходимо разработать программу, которая запускает в качестве подпрограммы программу, разработанную ранее.

G K [ директива IO2, фиксация начального адреса программы ]

[0] X 0 F [ для пошаговой отладки использовать Z 0 F ]

[1:] A 12 @ [<addr>] [ адрес массива (в начале работы обнуление аккумулятора не требуется) ]

[2:] T 6 F [ запись адреса массива в ячейку 6, обнуление аккумулятора ]

[3:] A 13 @ [<len>] [ длина массива ]

[4:] T 1 F [ запись длины массива в ячейку 1, обнуление аккумулятора ]

[5:] A 11 @ [<k>]

[6:] T 5 F [ запись значения k в ячейку 5, обнуление аккумулятора ]

[7:] T 0 F [0 ADDRESS USE AS ZERO]

[8:] A 8 @ [call subroute]

[9:] G 56 F

[10:] Z 0 F [ останов ]

[k:]

[11:] P 2 F [4] [С 0 ЭЛЕМЕНТА]

[addr:]

[12:] P 14 @ [<array>] [ адрес массива = <Начало программы>+10 (см. ниже) ]

[len:]

[13:] P 2 D [ длина массива - 5 ] [146]

[array:]

[14:] P 12 F

[15:] P 11 F

[16:] P 2 F [ 4 ]

[17:] P 1 F [ 2 ]

[18:] P 33 F

EZ PF [ директива IO2, переход к исполнению ]

Каркас основной программы:

Здесь мы в 1 инструкции относительно начала основной программы добавляем в асс адрес 0 элемента массива и записываем его в 6 ячейку памяти. Длина массива - в 1 ячейку памяти. Значение k – в 5 ячейку. Начало новой программы мы фиксируем директивой GK. Далее в 9 инструкции мы переходим в нашу подпрограмму.

Подпрограмма:

T 56 K [директива IO2, установка адреса загрузки]

G K [директива IO2, фиксация начального адреса подпрограммы]

[0] A 3 F [пролог: формирование кода инструкции возврата в Acc]

[1] T 75 @ [RETURN] [пролог: запись инструкции возврата]

[2] X 0 F [останов для отладки]

[0 - ZERO FOREVER]

[1 - RIGHT INDEX, INITIALY - LEN]

[2 - CURRENT INDEX]

[4 - FOR RUBBISH]

[5 - K VALUE]

[6 - ADDRESS FOR 0 ITEM]

[7 - TEMPORARY]

[3] X 0 F [останов для отладки]

[LOOP 1]

[4] T 4 F

[5] A 1 F

[6] S 68 @ [1 CONST]

[7] U 2 F

[8] G 61 @ [FINISH STEP]

[9] T 1 F

[FOR R1]

[10] A 69 @ [R-BLANK]

[11] A 6 F [ADDR] [READ I ITEM]

[12] T 29 @ [R1] [OPERATION: READ I ITEM]

[FOR R2]

[13] A 70 @ [R-BLANK-2]

[14] A 6 F [ADDR] [READ I ITEM]

[15] T 31 @ [R2] [OPERATION: READ I + 1 ITEM]

[FOR R3]

[16] A 70 @ [R-BLANK-2]

[17] A 6 F [ADDR] [READ I ITEM]

[18] T 56 @ [R3] [OPERATION: READ I + 1 ITEM]

[FOR W1]

[19] A 71 @ [W-BLANK]

[20] A 6 F [ADDR] [READ I ITEM]

[21] T 57 @ [W1] [OPERATION: WRITE I ITEM]

[FOR W2]

[22] A 72 @ [W-BLANK-2]

[23] A 6 F [ADDR] [READ I ITEM]

[24] T 59 @ [OPERATION: WRITE I + 1 ITEM]

[LOOP 2]

[25] A 2 F

[26] S 68 @ [1 CONST]

[27] G 4 @ [FIRST LOOP CONTINUATION]

[28] T 2 F

[29] A 0 F [R1 - DEFINITION] [OPERATION: READ I ITEM]

[30] T 7 F

[31] A 1 F [R2 - DEFINITION] [OPERATION: READ I + 1 ITEM]

[32] S 7 F

[33] G 55 @ [SWAP]

**Как изменилась подпрограмма?**

[CONTINIATION]

[34] A 68 @ [1 CONST]

[35] L 0 D

[36] A 29 @ [R1]

[37] T 29 @ [R1]

[38] A 68 @ [1 CONST]

[39] L 0 D

[40] A 31 @ [R2]

[41] T 31 @ [R2]

[42] A 68 @ [1 CONST]

[43] L 0 D

[44] A 56 @ [R3]

[45] T 56 @ [R3]

[46] A 68 @ [1 CONST]

[47] L 0 D

[48] A 57 @ [W1]

[49] T 57 @ [W1]

[50] A 68 @ [1 CONST]

[51] L 0 D

[52] A 59 @ [W2]

[53] T 59 @ [W2]

[54] E 25 @ [LOOP 2]

[SWAP]

[55] T 4 F

[56] A 1 F [R3 - DEFINITION] [OPERATION: READ I + 1 ITEM]

[57] T 0 F [W1 - DEFINITION] [WRITE I + 1 ITEM TO I INDEX]

[58] A 7 F [READ I ITEM]

[59] T 1 F [W2 - DEFINITION] [WRITE I ITEM TO I + 1 INDEX]

[60] E 34 @ [LOOP 2 CONTINIATION]

[FINISH STEP]

[61] T 4 F

[62] A 5 F [K]

[63] L 0 D

[64] A 6 F [ADDR]

[65] A 69 @ [R-BLANK]

[66] T 74 @ [LAST - RESULT]

[67] E 74 @ [LAST - RESULT]

[68] P 0 D [1 CONST]

[69] A 0 F [R-BLANK]

[70] A 1 F [R-BLANK-2]

[71] T 0 F [W-BLANK]

[72] T 1 F [W-BLANK-2]

[73] X 0 F

[74] A 0 F [LAST - RESULT]

[ret:]

[75] X 0 F [ эпилог: инструкция возврата из подпрограммы ]

[76] E 0 F [ эпилог: инструкция возврата из подпрограммы ]

* В первом блоке не надо загружать длину массива. Длина, значение k и длина массива уже хранится памяти.
* В 1 инструкции относительно начала подпрограммы мы формируем инструкцию возврата из подпрограммы. Т.е. в 76 инструкции относительно начала подпрограммы идет переход на то место, откуда мы вызвали подпрограмму.
* Все обращения к памяти, в которой хранятся инструкции подпрограммы, стали относительными (добавилась @ в конце инструкции)

Ответ все так же хранится в асс, после выхода из подпрограммы.