Санкт-Петербургский политехнический университет Петра Великого

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе №2**

**Дисциплина**: Низкоуровневое программирование

**Тема:** Программирование на EDSAC

Выполнил студент гр. 3530901/90003 В.М. Ковалевский

(подпись)

Преподаватель А.О. Алексюк

(подпись)

“ ” 2021 г.

Санкт-Петербург

2021

**СОДЕРЖАНИЕ**

[1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 3](#_Toc373321930)

[2. МЕТОД РЕШЕНИЯ 4](#_Toc1389088563)

[3.1. ПРОГРАММА ORDERS 1 5](#_Toc396163071)

[3.2. РАБОТА ПРОГРАММЫ ORDERS 1 9](#_Toc325896655)

[4.1. ПРОГРАММА ORDERS 2 10](#_Toc464313017)

[4.2. РАБОТА ПРОГРАММЫ ORDERS 2 14](#_Toc1010059508)

[5. АДРЕСА И ПРАВИЛА КОДИРОВАНИЯ ДАННЫХ И РЕЗУЛЬТАТОВ. 15](#_Toc162156692)

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Написать программу, реализующую слияние двух отсортированных массивов, для EDSAC (initial orders 1 и initial orders 2) .

# МЕТОД РЕШЕНИЯ

Метод решения довольно прост. У нас есть 2 входных массива и один выходной, в который мы добавляем наименьший элемент из двух, находящихся на текущих индексах входных массивов (первый элемент на позиции N первого массива, второй элемент на позиции K второго массива). Для этого нам нужно 3 счетчика для подсчета количества оставшихся элементов в первом и во втором массиве, а также для подсчета общего числа необработанных элементов. Когда мы добавляем элемент из первого массива первый счетчик уменьшается. Аналогично для второго. Если один из них равен 0 (мы обработали все элементы массива) - добавляем элемент из другого массива. Третий счетчик мы уменьшаем на 1 каждый цикл, пока он не станет равен 0 - мы получили результат.

# 3.1. ПРОГРАММА ORDERS 1

[ ТЗ: написать программу, осуществляющую слияние двух отсортированных массивов ]

[ Длина выходного массива хранится в 98, 1 массива в 99, 2 массива в 100; ]

[ Адреса хранятся соответственно в 101, 102, 103 ]

[ Массив 1 хранится в 104 - 107, массив 2 в 108 - 114 ]

[ Ячейки 0 - 5 рабочие ]

[ Результат находится в 115 - 125 ]

T 115 S

[32] X 0 S [ для отладки ]

[33] T 0 S [ запись аккумулятора в рабочую ячейку 0, обнуление аккумулятора ]

[34] A 98 [<len>] S [ загрузка в аккумулятор длины выходного массива ]

[35] T 2 S [ запись этого значения в ячейку 2 ]

[36] A 99 [<len1>] S [ загрузка в аккумулятор длины 1 обрабатываемого массива ]

[37] T 3 S [ запись этого значения в ячейку 3 ]

[38] A 100 [<len2>] S [ загрузка в аккумулятор длины 2 обрабатываемого массива ]

[39] T 4 S [ запись этого значения в ячейку 4 ]

[40] A 101 [<addr>] S [ загрузка в аккумулятор адреса 0-го элемента выходного массива ]

[41] L 0 L [ сдвиг аккумулятора на 1 разряд влево ]

[42] A 90 [<w>] S [ прибавляем код инструкции с нулевым полем адреса ]

[43] T 90 [<w>] S [ запись сформированной инструкции, обнуление аккумулятора ]

[44] A 102 [<addr1>] S [ загрузка в аккумулятор адреса 0-го элемента массива 1 ]

[45] L 0 L [ сдвиг аккумулятора на 1 разряд влево ]

[46] A 56 [<r1>] S [ прибавляем код инструкции с нулевым полем адреса ]

[47] T 56 [<r1>] S [ запись сформированной инструкции, обнуление аккумулятора ]

[48] A 103 [<addr2>] S [ загрузка в аккумулятор адреса 0-го элемента массива 2 ]

[49] L 0 L [ сдвиг аккумулятора на 1 разряд влево ]

[50] A 58 [<r2>] S [ прибавляем код инструкции с нулевым полем адреса ]

[51] T 58 [<r2>] S [ запись сформированной инструкции, обнуление аккумулятора ]

[loop:]

[52] A 2 S [ загружаем счетчик необработанных элементов массивов ]

[53] S 97 [<c1>] S [ уменьшаем на 1 ]

[54] G 96 [<exit>] S [ если результат меньше 0 завершаем работу ]

[55] T 2 S [ обновляем значение счетчика и обнуляем аккумулятор ]

[r1:]

[56] A 0 S [ загрузка в аккумулятор значения из ячейки N первого массива ]

[57] T 0 S [ запись этого значения в рабочую ячейку 0, обнуление аккумулятора ]

[r2:]

[58] A 0 S [ загрузка в аккумулятор значения из ячейки K второго массива ]

[59] T 1 S [ запись этого значения в рабочую ячейку 1, обнуление аккумулятора ]

[60] S 97 [<c1>] S [ уменьшаем на 1 ]

[61] A 3 S [ загружаем счетчик необработанных элементов массива 1]

[62] G 79 [<branch2>] S [ если результат меньше 0 добавляем элемент 2 массива ]

[63] S 3 S [ восстанавливаем -1 ]

[64] A 4 S [ загружаем счетчик необработанных элементов массива 2 ]

[65] G 70 [<branch1>] S [ если результат меньше 0 добавляем элемент 1 массива ]

[66] T 5 S [ обнуление аккумулятора ]

[67] A 1 S [ загружаем элемент второго массива ]

[68] S 0 S [ вычитаем из элемента второго массива элемент первого массива ]

[69] G 79 [<branch2>] S [ элемент второго массива меньше - добавляем его ]

[branch1:]

[70] T 5 S [ обнуление аккумулятора ]

[71] A 3 S [ загружаем счетчик необработанных элементов массива 1]

[72] S 97 [<c1>] S [ уменьшаем на 1 ]

[73] T 3 S [ обновляем значение счетчика и обнуляем аккумулятор ]

[74] A 97 [<c1>] S [ загрузка в аккумулятор константы 1 ]

[75] L 0 L [ сдвиг аккумулятора на 1 разряд влево ]

[76] A 56 [<r1>] S [ прибавляем код инструкции с нулевым полем адреса ]

[77] T 56 [<r1>] S [ запись сформированной инструкции, обнуление аккумулятора ]

[78] E 89 [<finally>] S [ не добавляем элемент 2 массива ]

[branch2:]

[79] T 5 S [ обнуление аккумулятора ]

[80] A 1 S [ считываем элемент 2 массива ]

[81] T 0 S [ записываем элемент 2 массива в ячейку 0 ]

[82] A 4 S [ загружаем счетчик необработанных элементов массива 1]

[83] S 97 [<c1>] S [ уменьшаем на 1 ]

[84] T 4 S [ обновляем значение счетчика и обнуляем аккумулятор ]

[85] A 97 [<c1>] S [ загрузка в аккумулятор константы 1 ]

[86] L 0 L [ сдвиг аккумулятора на 1 разряд влево ]

[87] A 58 [<r2>] S [ прибавляем код инструкции с нулевым полем адреса ]

[88] T 58 [<r2>] S [ запись сформированной инструкции, обнуление аккумулятора ]

[finally:]

[89] A 0 S [ загрузка в аккумулятор значения меньшего элемента ]

[w:]

[90] T 0 S [ запись этого значения в ячейку с адресом M, обнуление аккумулятора ]

[91] A 97 [<c1>] S [ загрузка в аккумулятор константы 1 ]

[92] L 0 L [ сдвиг аккумулятора на 1 разряд влево ]

[93] A 90 [<w>] S [ прибавляем код инструкции с нулевым полем адреса ]

[94] T 90 [<w>] S [ запись сформированной инструкции, обнуление аккумулятора ]

[95] E 52 [<loop>] S [ повторяем все операции ]

[exit:]

[96] Z 0 S [ останов ]

[c1:]

[97] P 0 L [ константа 1 ]

[len]

[98] P 5 L [ 11 ]

[len1:]

[99] P 2 S [ 4 ]

[len2:]

[100] P 3 L [ 7 ]

[addr:]

[101] P 57 L [115]

[addr1:]

[102] P 52 S [104]

[addr2:]

[103] P 54 S [108]

[array1:]

[104] P 1 S [ 2 ]

[105] P 2 L [ 5 ]

[106] P 3 S [ 6 ]

[107] P 4 S [ 8 ]

[array2:]

[108] P 0 S [ 0 ]

[109] P 0 L [ 1 ]

[110] P 1 L [ 3 ]

[111] P 2 S [ 4 ]

[112] P 3 L [ 7 ]

[113] P 4 L [ 9 ]

[114] P 5 S [ 10 ]

[115]

Все данные хранятся в конце, чтобы можно было менять длину массивов. Номера ячеек подписаны, кроме того в начале описано что где хранится.

Собственно о самой программе. Сначала мы записываем в ячейки **2 - 4** длины массивов. Длина выходного массива прописывается вручную, хотя можно было сделать так, чтобы она рассчитывалась автоматически. Эти ячейки используются как счетчики числа необработанных элементов.

Затем мы считываем адреса массивов и “переопределяем” инструкции **r1 (чтение из первого массива), r2 (чтение из второго массива), w (запись в выходной массив).** Можно увидеть что на позициях по этим метка лежат “пустые” инструкции чтения из ячейки с суммированием. Если прибавить к их коду адрес массива (предварительно сдвинув влево на **1 разряд**, так как адрес должен находиться **в позициях 10-1**), то эти инструкции будут “указывать” на начало массива. После чтения/записи нам нужно модифицировать инструкцию снова - добавив 1 (то есть мы перешли к следующей ячейке).

Теперь о том, как мы непосредственно решаем задачу. Сначала мы загружаем элемент первого массива, записываем в **ячейку 0**, потом загружаем элемент второго массива и записываем в **ячейку 1**. После этого, как ни странно, мы проверяем, не вышли ли мы за “пределы” массива. То есть одно из значений может быть “ошибочным”. В любом случае оно отбрасывается.

Есть 2 “ветки”: **branch1** - мы добавляем элемент первого массива, **branch2** - мы добавляем элемент второго массива. Если один из массивов “закончился” выбираем альтернативную ветку. Если нет, то сравниваем числа - вычитаем одно из другого и используем инструкцию перехода на нужную ветку. Я условился, что число, которое нужно будет записать находится в **ячейке 0**. То есть если мы записываем число из первого массива - оно уже находится в этой ячейке. Если из второго - копируем число из **ячейки 1** в **ячейку 0**.

После того, как мы записали новый элемент в выходной массив переходим в начало цикла.

# 3.2. РАБОТА ПРОГРАММЫ ORDERS 1



*Константа 1*







*Длины массивов*







*Адреса массивов*









*Первый массив*















*Второй массив*























*Результат*

# 4.1. ПРОГРАММА ORDERS 2

[ ТЗ: написать программу, осуществляющую слияние двух отсортированных массивов ]

[ Длины массивов записываются в ячейки 2, 3, 4; их адреса - в ячейки 5, 6, 7 ]

[ Ячейки 0, 1 и 8 - рабочие ]

[ Подпрограмма находится в ячейках 56 - 113 (56 + 57) ]

[ Тогда 0 элемент выходного массива находится в точке (начало основной программы) 114 + 33 = 147 ]

[sub:]

T 56 K [ директива IO2, установка адреса загрузки ]

G K [ директива IO2, фиксация начального адреса программы ]

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

[1] T 56 [<ret>] @ [ пролог: запись инструкции возврата ]

[2] A 5 [<addr>] F [ загрузка в аккумулятор адреса 0-го элемента выходного массива ]

[3] A 49 [<w>] @ [ прибавляем код инструкции с нулевым полем адреса ]

[4] T 49 [<w>] @ [ запись сформированной инструкции, обнуление аккумулятора ]

[5] A 6 [<addr1>] F [ загрузка в аккумулятор адреса 0-го элемента массива 1 ]

[6] A 15 [<r1>] @ [ прибавляем код инструкции с нулевым полем адреса ]

[7] T 15 [<r1>] @ [ запись сформированной инструкции, обнуление аккумулятора ]

[8] A 7 [<addr2>] F [ загрузка в аккумулятор адреса 0-го элемента массива 2 ]

[9] A 17 [<r2>] @ [ прибавляем код инструкции с нулевым полем адреса ]

[10] T 17 [<r2>] @ [ запись сформированной инструкции, обнуление аккумулятора ]

[loop:]

[11] A 2 F [ загружаем счетчик необработанных элементов массивов ]

[12] S 57 [<c1>] @ [ уменьшаем на 1 ]

[13] G 55 [<exit>] @ [ если результат меньше 0 завершаем работу ]

[14] T 2 F [ обновляем значение счетчика и обнуляем аккумулятор ]

[r1:]

[15] A 0 F [ загрузка в аккумулятор значения из ячейки N первого массива ]

[16] T 0 F [ запись этого значения в рабочую ячейку 0, обнуление аккумулятора ]

[r2:]

[17] A 0 F [ загрузка в аккумулятор значения из ячейки K второго массива ]

[18] T 1 F [ запись этого значения в рабочую ячейку 1, обнуление аккумулятора ]

[19] S 57 [<c1>] @ [ уменьшаем на 1 ]

[20] A 3 F [ загружаем счетчик необработанных элементов массива 1]

[21] G 38 [<branch2>] @ [ если результат меньше 0 добавляем элемент 2 массива ]

[22] S 3 F [ восстанавливаем -1 ]

[23] A 4 F [ загружаем счетчик необработанных элементов массива 2 ]

[24] G 29 [<branch1>] @ [ если результат меньше 0 добавляем элемент 1 массива ]

[25] T 8 F [ обнуление аккумулятора ]

[26] A 1 F [ загружаем элемент второго массива ]

[27] S 0 F [ вычитаем из элемента второго массива элемент первого массива ]

[28] G 38 [<branch2>] @ [ элемент второго массива меньше - добавляем его ]

[branch1:]

[29] T 8 F [ обнуление аккумулятора ]

[30] A 3 F [ загружаем счетчик необработанных элементов массива 1]

[31] S 57 [<c1>] @ [ уменьшаем на 1 ]

[32] T 3 F [ обновляем значение счетчика и обнуляем аккумулятор ]

[33] A 57 [<c1>] @ [ загрузка в аккумулятор константы 1 ]

[34] L 0 D [ сдвиг аккумулятора на 1 разряд влево ]

[35] A 15 [<r1>] @ [ прибавляем код инструкции с нулевым полем адреса ]

[36] T 15 [<r1>] @ [ запись сформированной инструкции, обнуление аккумулятора ]

[37] E 48 [<finally>] @ [ не добавляем элемент 2 массива ]

[branch2:]

[38] T 8 F [ обнуление аккумулятора ]

[39] A 1 F [ считываем элемент 2 массива ]

[40] T 0 F [ записываем элемент 2 массива в ячейку 0 ]

[41] A 4 F [ загружаем счетчик необработанных элементов массива 2]

[42] S 57 [<c1>] @ [ уменьшаем на 1 ]

[43] T 4 F [ обновляем значение счетчика и обнуляем аккумулятор ]

[44] A 57 [<c1>] @ [ загрузка в аккумулятор константы 1 ]

[45] L 0 D [ сдвиг аккумулятора на 1 разряд влево ]

[46] A 17 [<r2>] @ [ прибавляем код инструкции с нулевым полем адреса ]

[47] T 17 [<r2>] @ [ запись сформированной инструкции, обнуление аккумулятора ]

[finally:]

[48] A 0 F [ загрузка в аккумулятор значения меньшего элемента ]

[w:]

[49] T 0 F [ запись этого значения в ячейку с адресом M, обнуление аккумулятора ]

[50] A 57 [<c1>] @ [ загрузка в аккумулятор константы 1 ]

[51] L 0 D [ сдвиг аккумулятора на 1 разряд влево ]

[52] A 49 [<w>] @ [ прибавляем код инструкции с нулевым полем адреса ]

[53] T 49 [<w>] @ [ запись сформированной инструкции, обнуление аккумулятора ]

[54] E 11 [<loop>] @ [ повторяем все операции ]

[exit:]

[55] T 0 F [ обнуление аккумулятора ]

[ret:]

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

[c1:]

[57] P 0 D [ константа 1 ]

[:main]

G K [ директива IO2, фиксация начального адреса программы ]

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

[1] A 16 [<len>] @ [ длина выходного массива ]

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

[3] A 17 [<len1>] @ [ длина массива 1 ]

[4] T 3 F [ запись адреса массива в ячейку 3, обнуление аккумулятора ]

[5] A 18 [<len2>] @ [ длина массива 2 ]

[6] T 4 F [ запись адреса массива в ячейку 4, обнуление аккумулятора ]

[7] A 19 [<addr>] @ [ адрес выходного массива ]

[8] T 5 F [ запись адреса выходного массива в ячейку 5, обнуление аккумулятора ]

[9] A 20 [<addr1>] @ [ адрес массива 1]

[10] T 6 F [ запись адреса массива 1 в ячейку 6, обнуление аккумулятора ]

[11] A 21 [<addr2>] @ [ адрес массива 2]

[12] T 7 F [ запись адреса массива 2 в ячейку 7, обнуление аккумулятора ]

[13] A 13 @ [\ вызов ]

[14] G 56 F [<sub>] [/ подпрограммы ]

[15] Z 0 F [ останов ]

[len:]

[16] P 5 D [ 11 ]

[len1:]

[17] P 2 F [ 4 ]

[len2:]

[18] P 3 D [ 7 ]

[addr:]

[19] P 33 [<array>] @ [ адрес выходного массива = <Начало программы> + 33 ]

[addr1:]

[20] P 22 [<array1>] @ [ адрес массива 1 = <Начало программы> + 22 ]

[addr2:]

[21] P 26 [<array2>] @ [ адрес массива 2 = <Начало программы> + 26 ]

[array1:]

[22] P 1 F [ 2 ]

[23] P 2 D [ 5 ]

[24] P 3 F [ 6 ]

[25] P 4 F [ 8 ]

[array2:]

[26] P 0 F [ 0 ]

[27] P 0 D [ 1 ]

[28] P 1 D [ 3 ]

[29] P 2 F [ 4 ]

[30] P 3 D [ 7 ]

[31] P 4 D [ 9 ]

[32] P 5 F [ 10 ]

[array]

[33]

EZ PF [ директива IO2, переход к исполнению ]

Основное отличие от IO1 здесь в том, что мы оформляем наше решение в виде подпрограмм. Первая из них - наше решение IO1 с некоторыми модификациями. Во-первых мы используем индексацию относительно “якоря” (номер ячейки за командой G K). Во вторых здесь немного другой формат команд, вместо S - F, вместо L - D. Также, так как мы используем подпрограммы нам требуются возвращать управление программе вызвавшей ее. Для этого перед вызовом подпрограммы помещаем в аккумулятор адрес текущей ячейки, прибавляем к ней 3 и только потом переходим к подпрограмме. Собственно подпрограмма должна в конечную инструкцию записать этот адрес, который переведет нас прямо к инструкции, следующей за вызовом подпрограммы.

Отличие еще и в том, как мы задаем параметры. Теперь мы уже в главной программе записываем длины массивов в ячейки 2, 3, 4, а также адреса массивов в ячейки 5, 6, 7. Сами массивы хранятся в конце основной программы.

О том как вычислять номера ячеек с данными: первая команда основной программы находится в ячейке 114, тогда len будет в ячейке 114 + 16 и т.д.

# 4.2. РАБОТА ПРОГРАММЫ ORDERS 2



*Константа 1*







*Длины массивов*







*Адреса массивов (нужно смотреть на order, а не на Integer)*









*Первый массив*















*Второй массив*























*Выходной массив*

# АДРЕСА И ПРАВИЛА КОДИРОВАНИЯ ДАННЫХ И РЕЗУЛЬТАТОВ.

* Initial Orders 1:

Чтобы воспользоваться программой нужно:

1. В ячейках 98, 99, 100 указать длины массивов (выходной, первый, второй).
2. В ячейках 101, 102, 103 указать их адреса в таком же порядке.
3. Разместить сами массивы по указанным адресам. Для выходного массива просто нужно свободное место указанной длины, начиная с адреса.

Результат выполнения программы будет записан в ячейки с адресами, соответствующими адресам элементов выходного массива.

* Initial Orders 2:

Чтобы воспользоваться программой нужно:

1. В ячейки 2, 3, 4 записать длины массивов (выходной, первый, второй).
2. В ячейки 5, 6, 7 записать их адреса в таком же порядке.
3. Разместить сами массивы по указанным адресам. Для выходного массива просто нужно свободное место указанной длины, начиная с адреса.

Результат выполнения программы будет записан в ячейки с адресами, соответствующими адресам элементов выходного массива.

Массивы могут находиться где угодно, но нужно убедиться, что они не пересекаются.

Результат выполнения программы будет записан в ячейки со 147 строки. Это связано со смещением: запись подпрограммы (sub) в память EDSAC начинается с 57 ячейки (что указано в первой строке текста подпрограммы), подпрограмма занимает 57 строк, плюс длина основной программы в 21 строку (до массива). Таким образом, в сумме получается, что первая строка массива находится в памяти по адресу 57 + 57 + 33 = 157, а не 33, как это записано в основной программе.

В первой программе:

Массив 1 начинается с 104 **(addr1 = 104)** ячейки и заканчивается на 107 (**len1 = 4, addr1 + [len1 - 1] = 107**) => 104, 105, 106, 107;

Массив 2 начинается с 108 **(addr2 = 108)** ячейки и заканчивается на 114 (**len2 = 7, addr2 + [len2 - 1] = 114**) => 108, 109, 110, 111, 112, 113, 114;

Выходной массив начинается с 115 ячейки **(addr = 115)** и заканчивается на 125 **(len = 11, addr + [len - 1] = 125)**.

Во второй программе (**114** *- начало основной программы*):

Массив 1 начинается с 136 **(addr1 = 114 + 22 = 136)** ячейки и заканчивается на 139 (**len1 = 4, addr1 + [len1 - 1] = 139**) => 136, 137, 138, 139;

Массив 2 начинается с 140 **(addr2 = 114 + 26 = 140)** ячейки и заканчивается на 146 (**len2 = 7, addr2 + [len2 - 1] = 146**) => 140, 141, 142, 143, 144, 145, 146;

Выходной массив начинается с 147 ячейки **(addr = 114 + 33 = 147)** и заканчивается на 157 **(len = 11, addr + [len - 1] = 157)**.