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

Отчёт по лабораторной работе № 2

Дисциплина: Низкоуровневое программирование

Тема: программирование EDSAC

Вариант: 15

Выполнил студент гр. 3530901/90002 _____З.А. Фрид
(подпись)

Принял преподаватель _____ Д.С. Степанов
(подпись)

“____” _____ 2021 г.

Санкт-Петербург

2021

Постановка задачи

1. Разработать программу для EDSAC, реализующую определенную вариантом задания функциональность, и предполагающую загрузчик Initial Orders 1. Массив (массивы) данных и другие параметры (преобразуемое число, длина массива, параметр статистики и пр.) располагаются в памяти по фиксированным адресам.
2. Выделить определенную вариантом задания функциональность в замкнутую (closed) подпрограмму, разработать вызывающую ее тестовую программу. Использовать возможности загрузчика Initial Orders 2. Адрес обрабатываемого массива данных и другие параметры передавать через ячейки памяти с фиксированными адресами.

Вариант задания

Согласно варианту 15, необходимо реализовать слияние двух отсортированных массивов.

Initial Orders 1

Алгоритм реализован следующий:

1. Сравнение элемента A(i) и B(j);
2. Если B(j) > A(i), то в результирующий массив печатается A(i), а для следующего прохода в массиве A берется следующий элемент, B(j) остается прежним;
3. Если A(i) > B(j), то в результирующий массив печатается B(j), а для следующего прохода в массиве B берется следующий элемент, A(i) остается прежним;
4. Цикл 1-3 повторяется до тех пор, пока счетчик для массива R не будет равен 0.

Текст программы

```
[31] T 125[<end>] S [Конец программы]
[32] X 0 S
[33] T 0 S [acc = 0]
[34] A 84 [<adrr> len A] S [acc = len A]
[35] T 1 S [1 = acc = len A, acc = 0]
[36] A 85 [<adrr> len B] S [acc = len B]
[37] T 2 S [2 = acc = len B, acc = 0]
[38] A 86 [<adrr> len R] S [acc = len R]
[39] T 3 S [3 = acc = len R, acc = 0]
[loop:]
[40] [r1] A  96 [<adrr> A(i)] S [ acc = A(i)]
[41] T 4 S [4 = acc = A(i), acc = 0]
[42] [r2] A 97 [adrr B(j)] S
[43] U 5 S [5 = acc = B(j)]
[44] S 4 [A(i)] S [ Acc = B(j) -  A(i)]
[45] E 59 [X] S [if B(j) > A(i) goto <X>]

[else:]
[46] T 0 S [acc = 0]
[47] A 5 S [acc = B(j)]
[48] T 6 S [6 = acc = B(j), acc = 0]

[49] A 2 S [загружаем счетчик необработанных элементов массива B]
[50] S 83 [two] S [уменьшаем на 2]
[51] G 57 S [если результат < 0, не меняем r2, но печатаем B(j)]
[52] T 2 S [j = j - 2]

[53] A 83 [two] S [загрузка в аккумулятор константы 2]
[54] L 0 L [сдвиг на 1 разряд влево]
[55] A 42 [r2] S [прибавляем код инструкции, исполненной на предыдущем шаге]
[56] T 42 [r2] S [записываем сформированную инструкцию в память]

[57] A 6 S [acc = 6 = B(j)]
[58] E 72 [Y] S [goto Y]

[X:]
[59] T 0 S [acc = 0]
[60] A 4 S [acc = A(i)]
[61] T 6 S [6 = acc = A(i), acc = 0]
[62] A 1 S [загружаем счетчик необработанных элементов массива A]
[63] S 83 [two] S [уменьшаем на 2]
[64] G 70 S [если результат < 0, не меняем r1, но печатаем A(j)]
[65] T 1 S [i = i - 2]

[66] A 83 [two] S [загрузка в аккумулятор константы 2]
[67] L 0 L [сдвиг на 1 разряд влево ]
[68] A 40 [r1] S [прибавляем код инструкции, исполненной на предыдущем шаге]
[69] T 40 [r1] S [записываем сформированную инструкцию в память]
```

```
[70] A 6 S [acc = 6 = A(j)]
[71] E 72 [Y] S [goto Y]

[Y:]
[72] [w3] T 109 [adrr R(k)] S [R(k) = acc, acc = 0]
[73] A 3 S [загружаем счетчик необработанных элементов массива R]
[74] S 83 [two] S [уменьшаем на 2]
[75] G 82 [<exit>] S [если результат < 0, завершаем работу]
[76] T 3 S [k = k - 2]

[77] A 83 [two] S [загрузка в аккумулятор константы 2]
[78] L 0 L [сдвиг на 1 разряд влево]
[79] A 72 [w3] S [прибавляем код инструкции, исполненной на предыдущем шаге]
[80] T 72 [w3] S [записываем сформированную инструкцию в память]

[81] E 40[<loop>] S
[82] [exit] Z 0 S [останов]

[83] [two] P 1 S [const 2]
[84] [len A] P 5 S [= len A * 2 = 10, т.к. через 2]
[85] [len B] P 4 S  [= len B * 2 = 8, т.к. через 2]
[86] [len R] P 8 S  [= (len A + len B - 1) * 2 = 16, т.к. через 2]

[87] P 0 S
[88] P 0 S
[89] P 0 S
[90] P 0 S
[91] P 0 S [Для того, чтобы все массивы были на Long Tank 3]
[92] P 0 S
[93] P 0 S
[94] P 0 S
[95] P 0 S

[Array A]  [Через один для наглядности]
[Array B]

[96]  [A] P 7 S    [14]
[97]  [B] P 31 S   [62]
[98]  [A] P 127 S  [254]
[99]  [B] P 127 S  [254]
[100] [A] P 255 S  [510]
[101] [B] P 511 S  [1022]
[102] [A] P 1023 S [2046]
[103] [B] P 2047 S [4094]
[104] [A] P 4095 S [8190]

[105] P 0 S
[106] P 0 S  [Пробел между входными массивами и результатом]
[107] P 0 S

[Merged Array => M] [Через один для наглядности]
[Result Array => R]
[108] [M] P 7 S [14]
[109] [R] P 0 S
```

```
[110] [M] P 31 S [62]
[111] [R] P 0 S
[112] [M] P 127 S [254]
[113] [R] P 0 S
[114] [M] P 127 S [254]
[115] [R] P 0 S
[116] [M] P 255 S [510]
[117] [R] P 0 S
[118] [M] P 511 S [1022]
[119] [R] P 0 S
[120] [M] P 1023 S [2046]
[121] [R] P 0 S
[122] [M] P 2047 S [4094]
[123] [R] P 0 S
[124] [M] P 4095 S [8190]
```

Руководство

Так как мы проходим по массивам, нам необходимо иметь счетчики. Их задание происходит в ячейках [34] – [39], а сами счетчики хранятся в ячейках [1] – [3]. Так же дополнительно используются ячейки [4] и [5] для хранения текущих значений A(i) и B(j) соответственно. В ячейке [6] хранится значение A(i) или B(j), которое необходимо записать в результирующий массив.

Цикл начинается с ячейки [40]. Берется первый элемент массива A, записывается в ячейку [4], затем берется первый элемент массива B и записывается в ячейку [5], затем из B вычитается A ([40] - [45]).

Если результат получился положительным, то есть B(j) > A(i), тогда идем в ячейку [59]. Записываем A(i) в ячейку [6] для дальнейшей печати ([59] – [61]). В ячейках [62] – [65] уменьшаем счетчик для массива A, если это число равно 0, то сразу переходим к добавлению в массив R ([70] – [71]), иначе, перед печатью, меняем инструкцию r1, так, чтобы в следующем проходе на строке [40] в аккумулятор добавился следующий элемент массива A.

Аналогично, поступаем в случае, если B(j) < A(i). Записываем B(j) в ячейку [6] для дальнейшей печати ([46] – [48]). В ячейках [49] – [52] уменьшаем счетчик для массива B, если это число равно 0, то сразу переходим к добавлению в массив R ([57] – [58]), иначе, перед печатью, меняем инструкцию r2, так, чтобы в следующем проходе на строке [42] в аккумулятор добавился

следующий элемент массива B.

После любого из условий переходим к непосредственному добавлению элемента в массив R ([72]). В ячейках [73] – [76] уменьшаем счетчик для массива R, если это число равно 0, то завершаем программу, иначе переписываем инструкцию w3 так, чтобы в следующем проходе число записалось в следующую ячейку ([77] – [80]) и затем переходим к началу цикла ([81]).

Ячейки, начиная с [83] служат для задания констант, параметров и массивов.

Пример выполнения программы на симуляторе:

Для наглядности выведен Merged Array, который демонстрирует, каким должен быть Result Array.

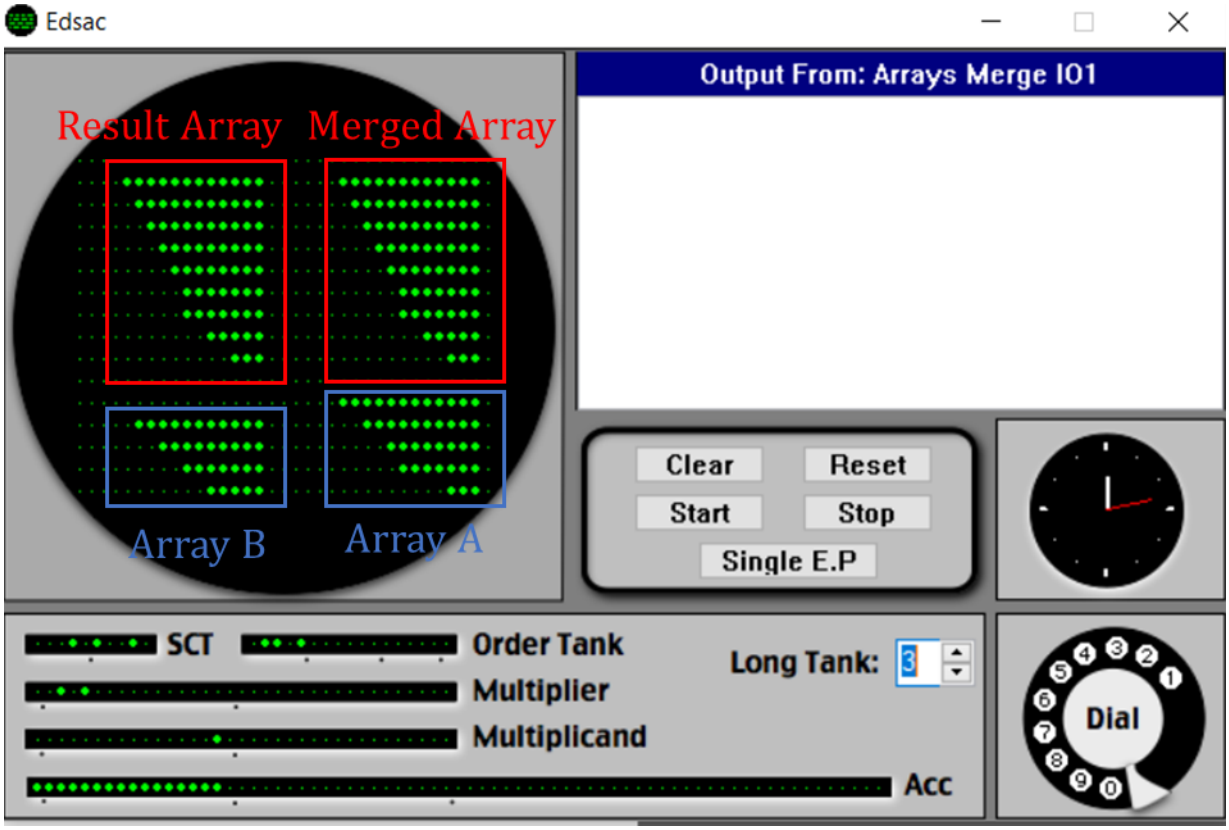

Рис. 1 Результат работы (с пометками)

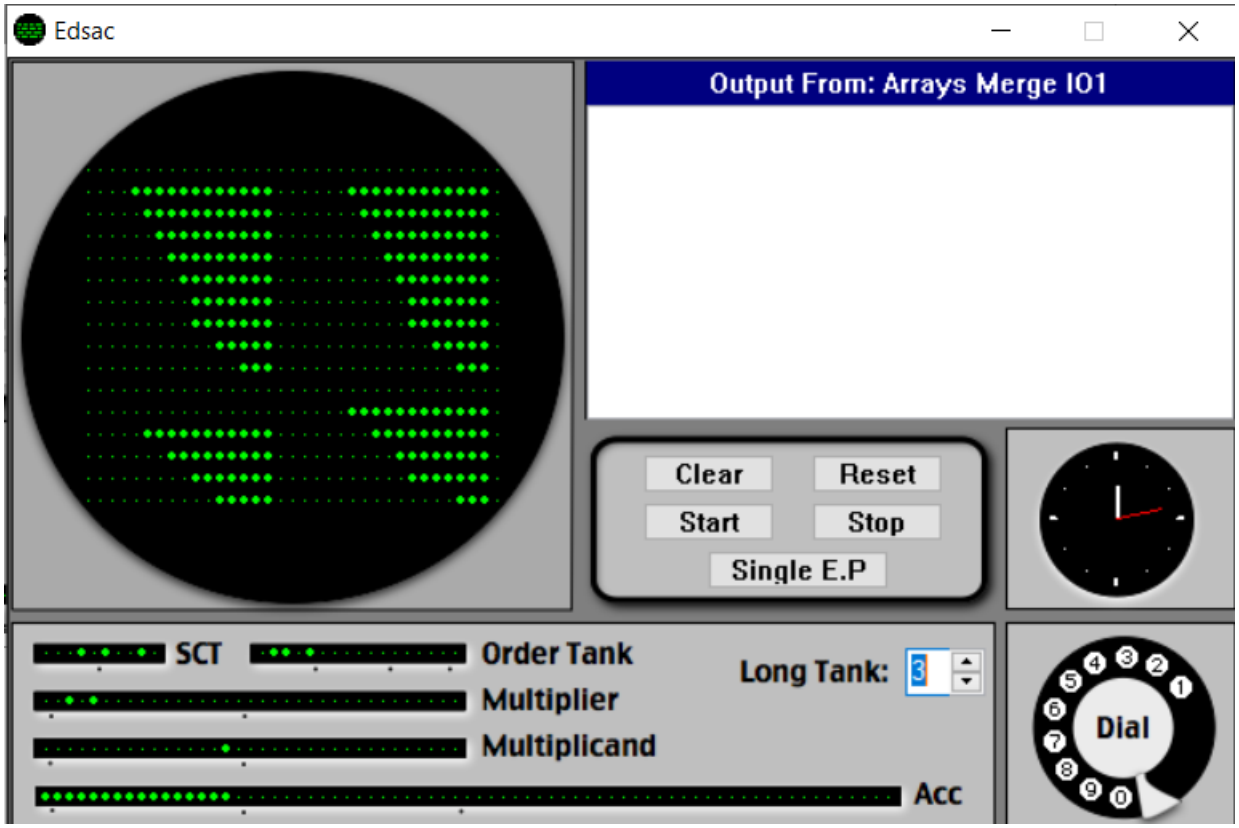

Рис. 2 Результат работы (без пометок)

Initial Orders 2

Функциональность программы, предполагающей загрузчик Initial Orders 1, была выделена в подпрограмму. Алгоритм не потерпел изменений.

Текст программы

```
[ПОДПРОГРАММА: ]
T 56 K [целевой адрес = 56]
G    K [ директива IO2, фиксация начального адреса подпрограммы ]

[ 0:] A 3 F [ пролог: формирование кода инструкции возврата в acc ]
[ 1:] T 44 [<ret>] @ [ пролог: запись инструкции возврата, acc = 0 ]

[loop:]
[ 2:] [r1] A 54 [<adrr> A(i)] @ [ acc = A(i)]
[ 3:] T 49 @ [4 = acc = A(i), acc = 0]
[ 4:] [r2] A 55 [adrr B(j)] @
[ 5:] U 50 @ [5 = acc = B(j)]
[ 6:] S 49 [A(i)] @ [ Acc = B(j) -  A(i)]
[ 7:] E 21 [X] @ [if B(j) > A(i) goto <X>]
```

```
[else:]
[ 8:] T 0 F [acc = 0]
[ 9:] A 50 @ [acc = B(j)]
[ 10:] T 51 @ [6 = acc = B(j), acc = 0]

[ 11:] A 47 @ [загружаем счетчик необработанных элементов массива B]
[ 12:] S 45 [two] @ [уменьшаем на 2]
[ 13:] G 19 @ [если результат < 0, не меняем r2, но печатаем B(j)]
[ 14:] T 47 @ [j = j - 2]

[ 15:] A 45 [two] @ [загрузка в аккумулятор константы 2]
[ 16:] L 0 D [сдвиг на 1 разряд влево]
[ 17:] A 4 [r2] @ [прибавляем код инструкции, исполненной на предыдущем
[ 18:] T 4 [r2] @ [записываем сформированную инструкцию в память]

[ 19:] A 51 @ [acc = 6 = B(j)]
[ 20:] E 34 [Y] @ [goto Y]

[X:]
[ 21:] T 0 F [acc = 0]
[ 22:] A 49 @ [acc = A(i)]
[ 23:] T 51 @ [6 = acc = A(i), acc = 0]

[ 24:] A 46 @ [загружаем счетчик необработанных элементов массива A]
[ 25:] S 45 [two] @ [уменьшаем на 2]
[ 26:] G 32 @ [если результат < 0, не меняем r1, но печатаем A(j)]
[ 27:] T 46 @ [i = i - 2]

[ 28:] A 45 [two] @ [загрузка в аккумулятор константы 2]
[ 29:] L 0 D [сдвиг на 1 разряд влево ]
[ 30:] A 2 [r1] @ [прибавляем код инструкции, исполненной на предыдущем шаге]
[ 31:] T 2 [r1] @ [записываем сформированную инструкцию в память]

[ 32:] A 51 @ [acc = 6 = A(j)]
[ 33:] E 34 [Y] @ [goto Y]
[Y:]
[ 34:] [w3] T 73 [adrr R(k)] @ [R(k) = acc, acc = 0]
[ 35:] A 48 @ [загружаем счетчик необработанных элементов массива R]
[ 36:] S 45 [two] @ [уменьшаем на 2]
[ 37:] G 44 [<exit>] @ [если результат < 0, завершаем работу]
[ 38:] T 48 @ [k = k - 2]

[ 39:] A 45 [two] @ [загрузка в аккумулятор константы 2]
[ 40:] L 0 D [сдвиг на 1 разряд влево]
[ 41:] A 34 [w3] @ [прибавляем код инструкции, исполненной на предыдущем шаге]
[ 42:] T 34 [w3] @ [записываем сформированную инструкцию в память]

[ 43:] E 2 [<loop>] @
[ 44:] [ret] E 0 F [эпилог: ипструкция возарата из подрограммы]

[ 45:] [two] P 1 F [const 2]
[ 46:] [len A] P 5 F [= len A * 2 = 10, т.к. через 2]
[ 47:] [len B] P 4 F  [= len B * 2 = 8, т.к. через 2]
[ 48:] [len R] P 8 F  [= (len A + len B - 1) * 2 = 16, т.к. через 2]
```

```
[ 49:] P 0 F [рабочая ячейка A]
[ 50:] P 0 F [рабочая ячейка B]
[ 51:] P 0 F [рабочая ячейка A или B]
[ 52:] P 0 F
[ 53:] P 0 F

[Array A]  [Через один для наглядности]
[Array B]
[ 54:] [A] P 7 F    [14]
[ 55:] [B] P 31 F   [62]
[ 56:] [A] P 127 F  [254]
[ 57:] [B] P 127 F  [254]
[ 58:] [A] P 255 F  [510]
[ 59:] [B] P 511 F  [1022]
[ 60:] [A] P 1023 F [2046]
[ 61:] [B] P 2047 F [4094]
[ 62:] [A] P 4095 F [8190]

[ 63:] P 0 F
[ 64:] P 0 F
[ 65:] P 0 F
[ 66:] P 0 F
[ 67:] P 0 F  [Для того, чтобы массивы]
[ 68:] P 0 F  [начинались на Long Tank 4]
[ 69:] P 0 F
[ 70:] P 0 F
[ 71:] P 0 F
[Merged Array => M] [Через один для наглядности]
[Result Array => R]
[ 72:] [M] P 7 F [14]
[ 73:] [R] P 0 F
[ 74:] [M] P 31 F [62]
[ 75:] [R] P 0 F
[ 76:] [M] P 127 F [254]
[ 77:] [R] P 0 F
[ 78:] [M] P 127 F [254]
[ 79:] [R] P 0 F
[ 80:] [M] P 255 F [510]
[ 81:] [R] P 0 F
[ 82:] [M] P 511 F [1022]
[ 83:] [R] P 0 F
[ 84:] [M] P 1023 F [2046]
[ 85:] [R] P 0 F
[ 86:] [M] P 2047 F [4094]
[ 87:] [R] P 0 F
[ 88:] [M] P 4095 F [8190]

[ТЕСТОВАЯ ПРОГРАММА: ]
      G   K   [Фиксация начального адреса программы]

[ 0:] A 0 @  [Вызов]
[ 1:] G 56 F [     подпрограммы]
[ 2:] Z 0 F  [Останов]

      EZ PF  [Переход к исполнению]
```

Руководство

Никаких существенных изменений нет. Основной рабочий цикл остался тем же, принцип работы тот же. Главное отличие – в относительной адресации.

Пример выполнения программы на симуляторе:

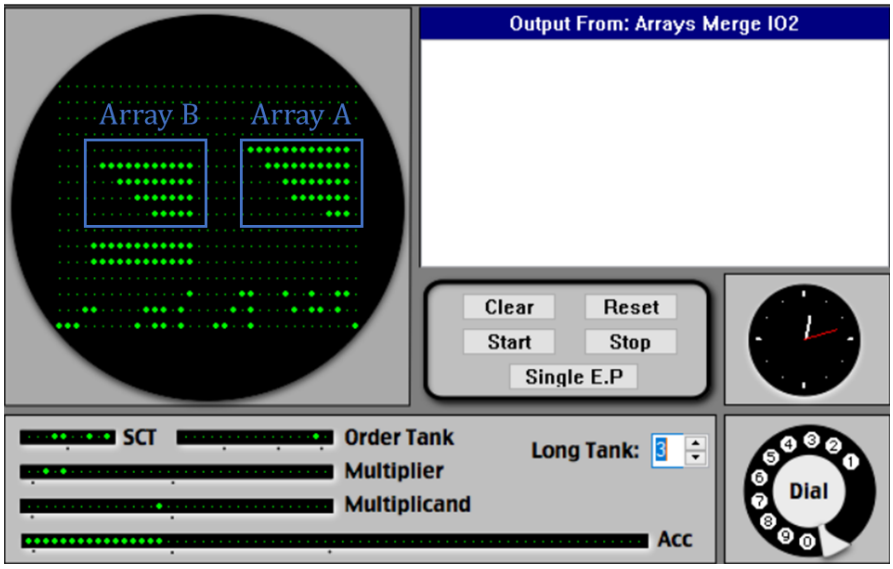

Рис. 3 Входные массивы

Для наглядности выведен Merged Array, который демонстрирует, каким должен быть Result Array.

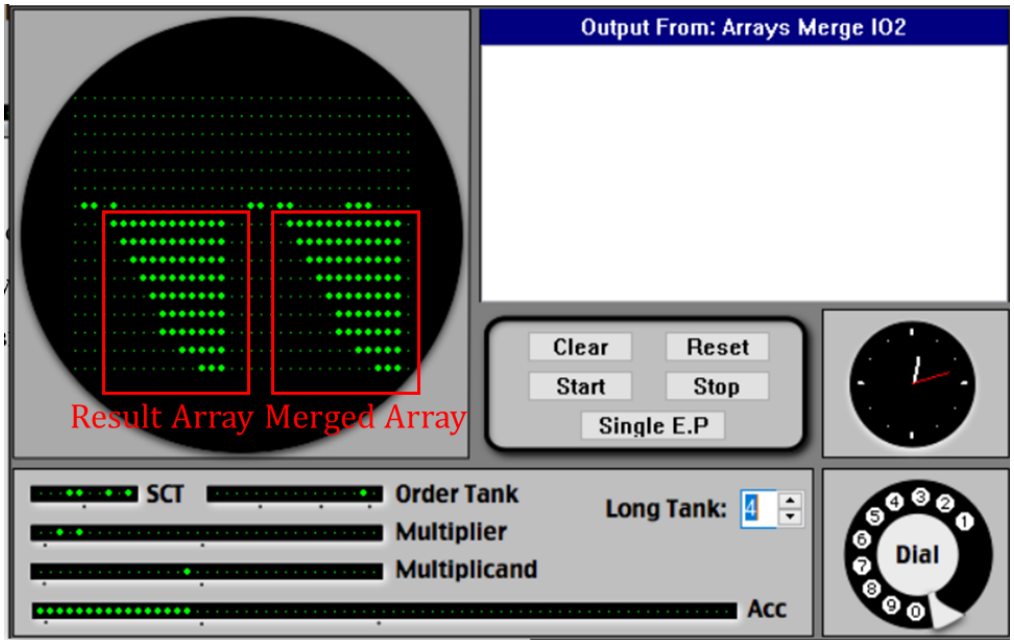

Рис. 4 Результат работы

Вывод

В ходе выполнения лабораторной работы была реализована программа для EDSAC, реализующая слияние двух отсортированных массивов для двух загрузчиков – Initial Orders 1 и Initial Orders 2; получены навыки по созданию алгоритмов для машины EDSAC, а также навыки работы с ее симулятором.

Список использованных источников

https://www.dcs.warwick.ac.uk/~edsac/Software/EdsacTG.pdf

http://kspt.icc.spbstu.ru/media/files/2019/lowlevelprog/edsac_io.pdf

http://kspt.icc.spbstu.ru/media/files/2019/lowlevelprog/edsac_prg.pdf

http://kspt.icc.spbstu.ru/media/files/2020/lowlevelprog/edsac_sub.pdf

http://kspt.icc.spbstu.ru/media/files/2019/lowlevelprog/edsac_subprg_c.pdf