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: 5

Выполнил студент гр. 3530901/90002 _____Е. В. Бурков
(подпись)

Принял преподаватель _____ Д. С. Степанов
(подпись)

“_____” _____ 2021 г.

Санкт-Петербург
2021

Формулировка задачи

1. Разработать программу для EDSAC, реализующую определённую вариантом задания функциональность, и предполагающую загрузчик Initial Orders 1. Массив данных и другие параметры располагаются в памяти по фиксированным адресам.

2. Выделить определённую вариантом задания функциональность в замкнутую (closed) подпрограмму, разработать вызывающую её тестовую программу. Использоваться возможности загрузчика Initial Orders 2. Адрес обрабатываемого массива данных и другие параметры передавать через ячейку памяти с фиксированными адресами.

Вариант задания

По варианту номер 5 необходимо реализовать сортировку обменом чисел in-place. Сортировка обменом является простым алгоритмом. Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N - 1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции, как пузырёк в воде — отсюда и название алгоритма).

Initial Orders 1

В данном алгоритме нам нужно итерироваться по массиву, значит нам нужен счётчик. Также внешнему циклу так же нужен счётчик. Алгоритм будет следующий:

- Берём x[j], вычитаем из него x[j+1]. Если знак положительный, то меняем местами.
- Увеличиваем j на единицу. Если j дошло до i-1, то обнуляем j и уменьшаем на 1 i. Если i дальше не уменьшается, то выходим из программы, иначе идём в начало.

Так же использовались шаблоны для модификации кода.

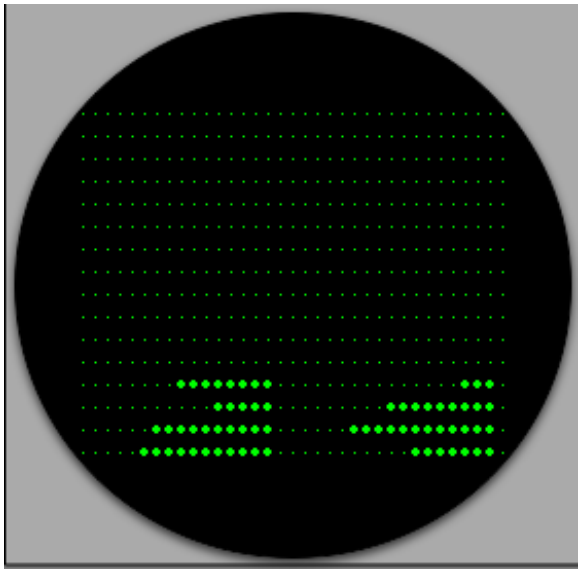
Код с комментариями представлен ниже:

```
T 104 S [Указатель на конец программы]
A 0 S     [Шаблон прибавления с 0 адресом]
S 1 S     [Шаблон вычитания с адресом 1]
T 0 S     [Шаблон записи с 0 адресом] [acc = 0]
A 32 S    [Добавляем шаблон] [acc = A 0 S]
T 2 S [2 = A 0 S , acc = 0]
A 33 S     [Добавляем шаблон] [acc = S 1 S]
T 3 S [3 = S 1 S, acc = 0]
A 34 S     [Добавляем шаблон] [acc = T 0 S]
T 4 S [4 = T 0 S , acc = 0]
A [ADR]87 S [acc = adr] [Адрес первого элемента массива]
U 5 S [5 = j = adr , acc = adr] [Записываем начало массива в 5 адрес]
A [LEN]86 S [acc = adr + len] [Определяем конец массива]
S [TWO]88 S [acc = adr + len - 1 ] [Условие завершение итерации цикла]
T 6 S [6 = adr + len - 1 , acc = 0] [Записываем условие]
[START] T 0 S [acc = 0] [Начинаем записывать инструкции с учётом адреса]
A 2 S    [acc = A 0 S]
A 5 S    [acc = A j S] [Взять j-й элемент массива]
U [Z1]61 S [z1 = acc = A j S] [Запись сформированной инструкции]
```

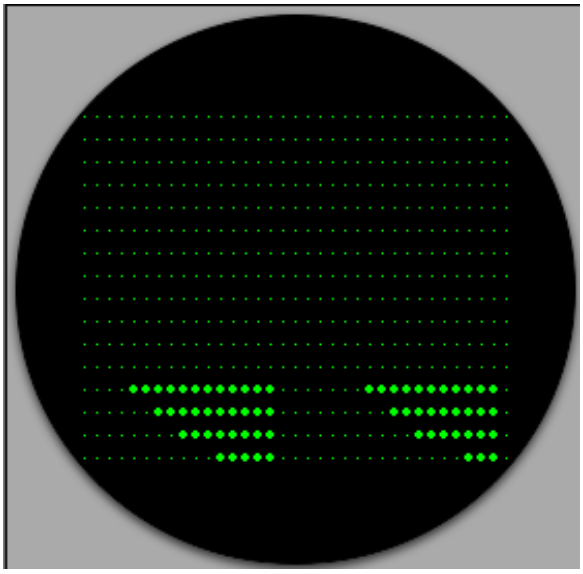
```
U [W1]65 S [w1 = acc = A j S] [Запись сформированной инструкции]
A [TWO]88 S [acc = A j+1 S] [Инструкция добавления j+1 элемента]
T [W2]67 S   [w1 = A j+1 S, acc = 0] [Запись сформированной инструкции]
A 3 S   [acc =   S 1 S]
A 5 S   [acc = S j+1 S] [Вычесть из аккума j+1-й элемент]
T [Z2]62 S   [z2 = S j+1 S, acc = 0] [Запись сформированной инструкции]
A 4 S   [acc = T 0 S]
A 5 S   [acc = T j S] [Записать на j-е место массива]
U [W3]68 S   [w3 = acc = T j S] [Запись сформированной инструкции]
A [TWO]88 S   [acc =   T j+1 S]    [Записать на j+1-е место массива]
T [W4]70 S    [w4 =   T j+1 S, acc = 0] [Запись сформированной инструкции]
[Z1] A 0 S   [ acc = xj ] [Начало цикла - Берём j-й элемент]
[Z2] S 0 S [acc = x(j) - x(j+1)] [Вычитание, чтобы посмотреть знак результата]
[Если отрицательный результат, то свап не нужен, иначе делаем]
G [SKIP-SWAP]71 S   [acc = x(j) - x(j+1)]
T 0 S [acc = 0] [Обнулим, чтобы мусор не помешал безупречной работе программы]
[W1] A 0 S [acc = x(j)] [Ранее сформированная инструкция]
T 0 S   [0 = x(j) , acc = 0]
[W2]A 0 S [acc = x(j+1)] [Ранее сформированная инструкция]
[W3]   T 0 S [j = x(j+1) , acc = 0] [Ранее сформированная инструкция]
A 0 S [acc = x(j)]
[W4]T 0 S [j+1 =x(j), acc = 0] [Ранее сформированная инструкция]
[SKIP-SWAP] T 0 S [acc = 0] [Отчистка, если условие было выполнено]
[INCREMENT J] A 5 S [acc = j] [Берём индекс элемента]
A [TWO]88 S   [acc = j + 1] [Инкрементируем индекс, чтобы идти дальше]
U 5 S   [5 = acc = j+1] [Запись нового индекса]
S 6 S [acc = j+1 - (i-1)] [Вычтем и посмотрим знак]
G [START]46 S [Знак отрицательный, можно сделать ещё проход]
T 0 S   [acc = 0] [Обнуление перед изменением]
```

```
A [ADR]87 S [acc = adr] [Адрес первого элемента массива]
T 5 S     [5 = adr , acc = 0] [Записываем на законное место]
A 6 S     [acc = i]
S [TWO]88 S    [acc = i--]
U 6 S         [acc = 6 = i] [Уменьшаем единицу]
S 5 S         [acc = i - j]
E [START]46 S [Если не меньше, то надо идти дальше, иначе выход из программы]
Z 0 S    [Выход!]
[LEN] P 8 S    [Длина массива]
[ADR] P 96 S [Адрес первого элемента]
[TWO] P 1 S    [Const 10 = 2 ] [Для инкрементации адреса]
[SKIPS] P 0 S
P 0 S
P 0 S
P 0 S [Пропустил чтобы на 3 экране перфокарты были только элементы массива]
P 0 S
P 0 S
P 0 S
[ARRAY]P 127 S [Массив]
P 4095 S          [Выбраны числа степени двойки - 1 для более]
P 2047 S          [ удобного их распознавания]
P 1023 S
P 511 S
P 31 S
P 7 S
P 255 S
```

Проведём тестирование программы:



Массив изначально



Массив после сортировки

Initial Orders 2 и руководство программиста

С учётом возможностей загрузчика Initial Orders 2 была написана программа, листинг которой находится в приложении 1.

Для начала была написана программа-заглушка, которая вставляла в 0 и 1 ячейки памяти адрес и длину массива.

```
[sub]
G K   [ Директива, фиксация начального адреса]
[далее в квадратных скобах адреса @ ]
[0] A 3 F [Пролог: формирование кода инструкции возврата]
[1] T 3 [RET] @     [Пролог: запись инструкции возврата]
[3]   [RET] E 0 F [EPILOG, RETURN FROM FUNC]
[test routine]
G K
[0] X 0 F
[1] A [ADR] 8 @
[2] T 0 F [WRITE TO 1]
[3] A [LEN] 9 @ [LEN]
[4] T 1 F [WRITE TO m]
[5] A 5 @                [CALL]
[6] G [SUB] 56 F   [SUBPROGRAMM]
[7] Z 0 F [STOP]
[8] [ADR] P 10 @
[9] [LEN] P 10 F
[10] [ARRAY]P 127 F
P 15 F
P 31 F
P 255 F
P 511 F
P 2047 F
P 1023 F
```

```
P 31 F
P 255 F
P 7 F
EZ PF
```

После мной была переписана программа из Initial Orders 1 с учётом директив Initial Orders 2. Так же при помощи параметров я определил ячейки, где должны храниться длина и адрес массива, что позволяет сделать закрытую подпрограмму удобней и практичней для использования.

Руководство программиста:

Раздел [adresses] отвечает за установку адресов параметров для подпрограммы.

[sub] – подпрограмма сортировки обменами. Далее приведу пояснения по коду:

[0-1] Запись адреса возврата для выхода из подпрограммы.

[2-6] Запись параметров в выделенные для этого ячейки.

[7-21] Изменяем код, вставляя туда нужный адрес.

[22-24] Вычитаем x[j+1] из x[j], если знак положительный, то надо поменять местами [25-31].

[33-35] Увеличиваем счётчик на 1.

[35-37] Проверяем, чтобы не уйти за границы неотсортированной части массива.

[38-43] Обнуляем j, уменьшаем i.

[44-45] Смотрим сколько уже отсортировали.

[48-51] Константы для подпрограммы.

[52-54] Переменные подпрограммы.

[test routine] – тестовая программа, которая вызывает замкнутую подпрограмму.

Вывод

В ходе выполнения данной лабораторной работы был получен опыт программирования на EDSAC и работы с двумя его загрузчиками. Несмотря на неудобство, можно выполнять весьма нетривиальные задачи. Также можно заметить, что многие принципы работы EDSAC имеют общие черты с современными компьютерами, что в свою очередь делает данную лабораторную работу очень поучительной.

Приложение 1

Листинг программы для загрузчика Initial Orders 2:

```
T 56 K [Адрес загрузки]


[adresses]
G K [ Директива, фиксация начального адреса]
T 45 K [ Установка адреса для параметров далее]
P 256 F [45 = H , тут будет адрес первого элемента массива]
P 255 F [46 = N , длинна массива для сортировки]
TZ   [ востановления адреса целевой ячейки]


[sub]
G K   [ Директива, фиксация начального адреса]
[далее в квадратных скобах адреса @ ]
[0] A 3 F [Пролог: формирование кода инструкции возврата]
[1] T 47 [RET] @     [Пролог: запись инструкции возврата]
[2]   A 0 H   [adr = H]     [set adr and len on 52,53]
[3]   U 52 @
[4]   A 0 N   [len = N]
[5]   S [TWO]48 @
[6]   T 53 @
[7] [START] T 54 @       [self-modified code]
[8]   A [A0]49 @
[9]   A 52 @
[10] U [Z1]22 @
[11] U [W1] 26 @
[12] A [TWO]48 @
[13] T [W2]28 @
[14] A [S1]50 @
[15] A 52 @
```

```
[16] T [Z2]23 @
[17] A [T0]51 @
[18] A 52 @
[19] U [W3]29 @
[20] A [TWO]48 @
[21] T [W4]31 @          [/self-modified code]
[22] [Z1] A 0 F          [swapping]
[23] [Z2] S 0 F
[24] G [SKIP-SWAP]32 @
[25] T 54 @
[26] [W1] A 0 F
[27] T 54 @
[28] [W2] A 0 F
[29] [W3] T 0 F
[30]   A 54 @
[31] [W4] T 0 F          [/swapping]
[32]   [SKIP-SWAP] T 54 @
[33]   [INCREMENT J] A 52 @
[34]    A [TWO]48 @
[35]    U 52 @
[36]    S 53 @
[37]    G [START]7 @
[38]    T 54 @
[39]     A 0 H   [adr = H]
[40]    T 52 @
[41]    A 53 @
[42]    S [TWO]48 @
[43]    U 53 @
[44]    S 52 @
[45]    E [START]7 @
```

```
[46]   [EXIT] T 54 @
[47]   [RET] E 0 F [EPILOG, RETURN FROM FUNC]
[CONSTS]
[48]   [TWO] P 1 F
[49]   [A0] A 0 F
[50]   [S1] S 1 F
[51]   [T0] T 0 F
[52]   [J] P 0 F
[53]   [I]   P 0 F
[54]   [TMP] P 0 F

[test routine]
G K
[0] X 0 F
[1] A [ADR] 8 @
[2] T 256 F [WRITE TO n]
[3] A [LEN] 9 @ [LEN]
[4] T 255 F [WRITE TO m]
[5] A 5 @                [CALL]
[6] G [SUB] 56 F   [SUBPROGRAMM]
[7] Z 0 F [STOP]
[8] [ADR] P 17 @
[9] [LEN] P 10 F
[10] [SKIPS] P 0 F
P 0 F
P 0 F
P 0 F
P 0 F
P 0 F
P 0 F
```

```
[17] [ARRAY]P 1023 F
P 511 F
P 255 F
P 127 F
P 63 F
P 31 F
P 15 F
P 7 F
P 3 F
P 1 F
EZ PF
```