

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.

Формулировка задачи

1. Разработать программу для EDSAC, реализующую определённую вариант задания функциональность, и предполагающую загрузчик Initial Orders

1. Массив данных и другие параметры располагаются в памяти по фиксированным адресам.

2. Выделить определённую вариант задания функциональность в замкнутую (closed) подпрограмму, разработать вызывающую её тестовую программу. Использовать 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 [END]104 S [Указатель на конец программы]
A 0 S [32] [Шаблон прибавления с 0 адресом]
S 1 S [33] [Шаблон вычитания с адресом 1]
T 0 S [34] [Шаблон записи с 0 адресом] [acc = 0]
T 4 S [F=0] [Условие выхода (массив отсортирован)]
A [ADR]88 S [acc = adr]
U 2 S [2 = acc = adr = j] [Добавляем счётчик в 2]
A [LEN]87 S [acc = adr + len]
S [TWO]89 S [acc = adr + len - 1] [До куда итерируемся вообще]
T 3 S [3 = adr + len - 1 = i]
[START] T 0 S [acc = 0]
A 32 S [acc = A 0 S]
A 2 S [acc = A j S]
U [Z1]56 S [z1 = acc = A j S] [Запись сформированной инструкции]
U [W1]62 S [w1 = acc = A j S] [Запись сформированной инструкции]
A [TWO]89 S [acc = A j+1 S] [Инструкция добавления j+1 элемента]
T [W2]64 S [w2 = A j+1 S, acc = 0] [Запись сформированной инструкции]
A 33 S [acc = S 1 S]
A 2 S [acc = S j+1 S] [Вычесть из аккума j+1-й элемент]

T [Z2]57 S [z2 = S j+1 S, асс = 0] [Запись сформированной инструкции]
 A 34 S [асс = T 0 S]
 A 2 S [асс = T j S] [Записать на j-е место массива]
 U [W3]65 S [w3 = асс = T j S] [Запись сформированной инструкции]
 A [TWO]89 S [асс = T j+1 S] [Записать на j+1-е место массива]
 T [W4]67 S [w4 = T j+1 S, асс = 0] [Запись сформированной инструкции]
 [Z1] A 0 S [асс = xj] [Начало цикла - Берём j-й элемент]
 [Z2] S 0 S [асс = x(j) - x(j+1)] [Вычитание, чтобы посмотреть знак результата]
 [Если отрицательный результат, то свап не нужен, иначе делаем]
 G [SKIP-SWAP]68 S [асс = x(j) - x(j+1)]
 T 0 S [асс = 0] [Обнулим, чтобы мусор не помешал безупречной работе программы]
 A [TWO]89 S [асс = 1]
 T 4 S [4 = 1, асс = 0] [Флаг]
 [W1] A 0 S [асс = x(j)] [Ранее сформированная инструкция]
 T 0 S [0 = x(j) , асс = 0]
 [W2] A 0 S [асс = x(j+1)] [Ранее сформированная инструкция]
 [W3] T 0 S [j = x(j+1) , асс = 0] [Ранее сформированная инструкция]
 A 0 S [асс = x(j)]
 [W4] T 0 S [j+1 =x(j), асс = 0] [Ранее сформированная инструкция]
 [SKIP] T 0 S [асс = 0] [Отчистка, если условие было выполнено]
 [INC J] A 2 S[асс = j] [Берём индекс элемента]
 A [TWO]89 S [асс = j + 1] [Инкрементируем индекс, чтобы идти дальше]
 U 2 S [5 = асс = j+1] [Запись нового индекса]
 S 3 S [асс = j+1 - (i-1)] [Вычтем и посмотрим знак]
 G [START]41 S[Знак отрицательный, можно сделать ещё проход]
 T 0 S [асс = 0] [Обнуление перед изменением]
 S 4 S [асс = 0 - C(4)]
 E [END]86 S [Не было свапов, значит отсортировано]
 T 0 S [асс = 0]

T 4 S [C(4) = acc = 0]
 A [ADR]88 S [acc = adr] [Адрес первого элемента массива]
 T 2 S [C(2) = adr , acc = 0] [Записываем на законное место]
 A 3 S [acc = i]
 S [TWO]89 S [acc = i--]
 U 3 S [acc = c(3) = i] [Уменьшаем на единицу]
 S 2 S [acc = i - j]
 E [START]41 S [Если не меньше, то надо идти дальше, иначе выход из программы]
 [END] Z 0 S
 [LEN] P 8 S [Длина массива + 1]
 [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
 [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

P 7 S [Сортированный Массив]

P 31 S

P 127 S

P 255 S

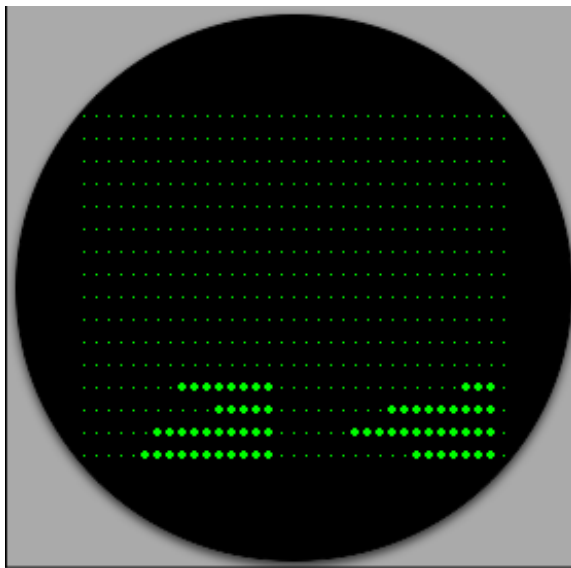
P 511 S

P 1023 S

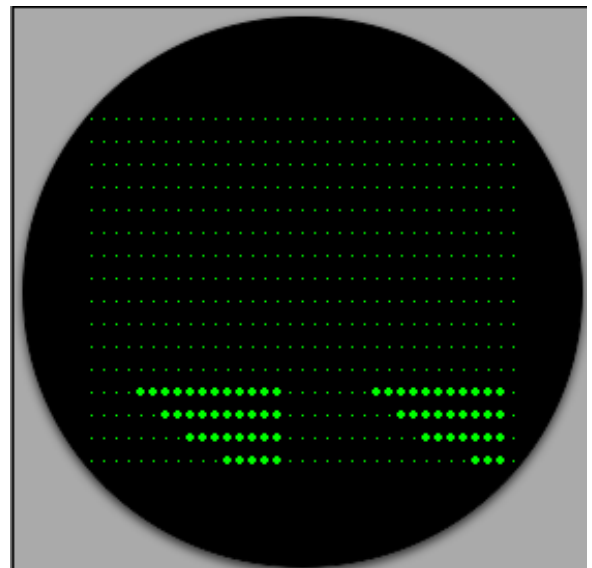
P 2047 S

P 4095 S

Проведём тестирование программы:



Массив изначально



Массив после сортировки

Так же было учтено то, что массив уже отсортирован, тогда программа сразу завершает работу после первого прохода.

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 [addresses] отвечает за установку адресов параметров для подпрограммы.

[sub] – подпрограмма сортировки обмёнами. 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]

[addresses]

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 [восстановления 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 54 [RET] @ [Пролог: запись инструкции возврата]

[2] A 0 H [adr = H] [set adr and len on 52,53]

[3] U 59 @

[4] A 0 N [len = N]

[5] S [TWO]55 @

[6] T 60 @

[7] [START] T 61 @ [self-modified code]

[8] A [A0]56 @

[9] A 59 @

[10] U [Z1]22 @

[11] U [W1] 28 @

[12] A [TWO]55 @

[13] T [W2]30 @

[14] A [S1]57 @

[15] A 59 @

[16] T [Z2]23 @
 [17] A [T0]58 @
 [18] A 59 @
 [19] U [W3]31 @
 [20] A [TWO]55 @
 [21] T [W4]33 @ [/self-modified code]
 [22] [Z1] A 0 F [swapping]
 [23] [Z2] S 0 F
 [24] G [SKIP-SWAP]34 @
 [25] T 61 @
 [26] A [TWO]55 @
 [27] T [F]62 @
 [28] [W1] A 0 F
 [29] T 61 @
 [30] [W2] A 0 F
 [31] [W3] T 0 F
 [32] A 61 @
 [33] [W4] T 0 F [/swapping]
 [34] [SKIP-SWAP] T 61 @
 [35] [INCREMENT J] A 59 @
 [36] A [TWO]55 @
 [37] U 59 @
 [38] S 60 @
 [39] G [START]7 @
 [40] T 61 @
 [41] S [F]62 @
 [42] E [END]53 @
 [43] T 61 @
 [44] T [F]62 @
 [45] T 61 @

```

[46]   A 0 H   [adr = H]
[47]   T 59 @
[48]   A 60 @
[49]   S [TWO]55 @
[50]   U 60 @
[51]   S 59 @
[52]   E [START]7 @
[53]   [EXIT] T 62 @
[54]   [RET] E 0 F [EPILOG, RETURN FROM FUNC]
[CONSTS]
[55]   [TWO] P 1 F
[56]   [A0] A 0 F
[57]   [S1] S 1 F
[58]   [T0] T 0 F
[59]   [J] P 0 F
[60]   [I]  P 0 F
[61]   [TMP] P 0 F
[62] [f] 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] E 23 @ [GO TO SECOND TEST]   [Z 0 F]
[8] [ADR] P 13 @

```

[9] [LEN] P 10 F
 [10] [SKIPS] P 0 F
 P 0 F
 P 0 F
 [13] [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
 [23] A [ADR2]30 @
 [24] T 256 F [WRITE ADR-2]
 [25] A [LEN-2]31 @
 [26] T 255 F [WRITE LEN-2]
 [27] A 27 @
 [28] G [SUB] 56 F
 [29] Z 0 F [STOP]
 [30] [ADR-2] P 32 @
 [31] [LEN-2] P 8 F
 P 7 F
 P 31 F
 P 1 F
 P 255 F [SORTED ARRAY] [LEN 8]
 P 156 F
 P 1023 F
 P 11 F

P 4095 F

EZ PF