Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчет по лабораторной работе №2**

**Дисциплина:** Низкоуровневое программирование

**Тема:** EDSAC.

Вариант №11

Выполнил

студентка гр. 5130901/20004 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.В. Обухова

(подпись)

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Целищева

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Санкт-Петербург   
2023

Задача:

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.

Задание по варианту:

Загрузка с перфоленты последовательности чисел. Числа записываются в десятичной форме (со знаком), заканчиваются символом «#».

Код для Initial Order 1:

[31] T 109 S [32] X 0 S [33] P 5 S [число, на которое умножается] [34] H 33 S [добавление в умножающий регистр] [35] P 0 S [константа 0] [36] A 35 S [добавляем константу в аккумулятор] [37] T 109 S [запись в ячейку для обнуления] [38] T 3 S [обнуление] [39] T 0 S [обнуление] [40] T 2 S [обнуление] [41] A 0 S [добавление следующей цифры в аккумулятор] [42] A 109 S [добавление предыдущего числа в аккумулятор] [43] T 109 S [запись числа в ячейку] [44] T 2 S [обнуление] [45] I 0 S [считывание числа с перфоленты] [46] A 0 S [считывание в аккумулятор, что считали с перфоленты] [47] S 108 S [вычитание константы для проверки цифра/знак] [48] E 64 S [если не цифра - переход на проверку] [49] T 5 S [очистка аккумулятора] [50] A 106 S [запись 20] [51] S 0 S [вычитание для проверки] [52] T 2 S [53] A 2 S [54] G 63 S [проверка входа в цикл] [55] T 2 S [56] V 109 S [умножение на 10] [57] L 1024 S [для выравнивания умножения] [58] L 4 S [для выравнивания умножения] [59] T 109 S [60] A 106 S [61] S 0 S [62] E 40[] S [возвращаемся, чтобы считать след. символ] [63] T 2 S

[64] S 107 S [проверка на +- или #] [65] E 49 S

[66] A 107 S [добавляем константу в аккумулятор] [67] L 0 L [сдвиг аккумулятора влево] [68] A 37 S [добавляем в аккумулятор код инструкции->код инструкции меняет n] [69] T 37 S [записываем новую инструкцию на место старой]

[70] A 107 S [71] L 0 L [72] A 42 S [73] T 42 S

[74] A 107 S [75] L 0 L [76] A 43 S [77] T 43 S

[78] A 107 S [79] L 0 L [80] A 56 S [81] T 56 S

[82] A 107 S [83] L 0 L [84] A 59 S [85] T 59 S

[86] A 107 S[проверка на + или -] [87] A 3 S [88] E 92 S [89] T 1 S [90] S 109 S [вычитаем из 0 наше число] [91] T 109 S [92] T 1 S

[93] A 107 S [94] L 0 L [95] A 90 S [96] T 90 S

[97] A 107 S [98] L 0 L [99] A 91 S [100] T 91 S

[101] A 2 S [запись нового знака] [102] T 3 S [103] A 3 S

[104] G 38 S [если новый символ знак-> все заново] [105] X 0 S [the end]

[106] P 10 S [20] [107] P 0 L [1] [108] P 5 L [11]

+134 -212 -53 +69 -45 -76#

Пример:

Входные данные:

+134 -212 -53 +69 -45 -76#

Выходные данные:













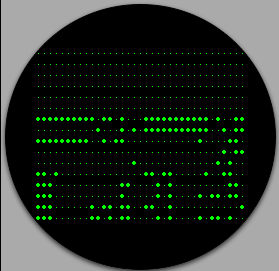


Рис. 1 Выходные данные программы. Initial order 1.

В результате числа записываются в десятичной форме со знаком в ячейках памяти.

Код для Initial Order 2:

[44] T 56 K [директива IO1, установка адреса загрузки] G K [директива IO2, фиксация начального адреса подпрограммы] [0] A 3 F [пролог, формирование кода инструкции возврата в аккумуляторе] [1] T 77 @ [пролог, запись инструкции возврата] [2] P 5 F [число, на которое умножается] [3] H 2 @ [добавление в умножающий регистр] [4] P 0 F [0] [5] T 5 F[обнуление] [6] A 4 @ [добавляем константу в аккумулятор] [7] T 145 F [запись в ячейку для обнуления] [8] T 3 F [обнуление] [9] T 0 F [обнуление] [10] T 2 F [обнуление] [11] A 0 F [добавление следующей цифры в аккумулятор] [12] A 145 F [добавление предыдущего числа в аккумулятор] [13] T 145 F [запись числа в ячейку] [14] T 2 F [обнуление] [15] I 0 F [считывание числа с перфоленты] [16] A 0 F [считывание в аккумулятор, что считали с перфоленты] [17] S 144 F [вычитание константы для проверки цифра/знак] [18] E 34 @ [если не цифра - переход на проверку] [19] T 5 F [очистка аккумулятора] [20] A 142 F [запись 20] [21] S 0 F [вычитание для проверки] [22] T 2 F [23] A 2 F [24] G 33 @ [проверка входа в цикл] [25] T 2 F [26] V 145 F [умножение на 10] [27] L 1024 F [для выравнивания умножения] [28] L 4 F [для выравнивания умножения] [29] T 145 F [30] A 142 F [31] S 0 F [32] E 10 @ [возвращаемся, чтобы считать след. символ] [33] T 2 F

[34] S 143 F [проверка на +- или #] [35] E 19 @ [36] T 5 F [обнуление] [37] A 143 F [добавляем константу в аккумулятор] [38] L 0 D [сдвиг аккумулятора влево] [39] A 7 @ [добавляем в аккумулятор код инструкции-> код инструкции меняет n] [40] T 7 @ [записываем новую инструкцию на место старой]

[41]A 143 F [42]L 0 D [43]A 12 @ [44]T 12 @

[45]A 143 F [46]L 0 D [47]A 13 @ [48]T 13 @

[49]A 143 F [50]L 0 D [51]A 26 @ [52]T 26 @

[53]A 143 F [54]L 0 D [55]A 29 @ [56]T 29 @

[57]A 143 F[проверка на + или -] [58]A 3 F [59]E 63 @ [60]T 1 F [61]S 145 F [вычитаем из 0 наше число] [62]T 145 F [63]T 1 F

[64]A 143 F [65]L 0 D [66]A 61 @ [67]T 61 @

[68]A 143 F [69]L 0 D [70]A 62 @ [71]T 62 @

[72]A 2 F[запись нового знака] [73]T 3 F [74]A 3 F

[75]G 8 @ [если новый символ знак->все заново] [76]X 0 F [the end] [77]E 0 F [эпилог, инструкция возврата из подпрограммы] G K [директива IO2 фиксация начального адреса подпрограммы] [0] Z 0 F [для Single E.P, -> X 0 S] [1] T 5 F [обнуление] [2] T 3 F [обнуление] [3] T 0 F [обнуление] [4] T 2 F [обнуление]

[5] A 5 @ [вызов подпрограммы] [6] G 56 F

[7] Z 0 F [для Single E.P, -> X 0 S] [8] P 10 F [20] [9] P 0 D [1] [10] P 5 D [11] [11]EZ PF [директива IO2, переход к исполнению]

+134 -212 -53 +69 -45 -76#

Пример:

Входные данные:

+134 -212 -53 +69 -45 -76#

Выходные данные:













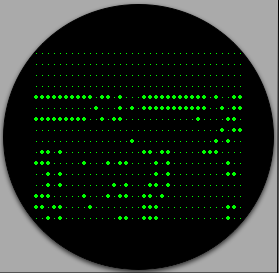


Рис. 2 Выходные данные программы. Initial order 2.

Для Initial Orders 1 максимальное число для записи 65535, минимальное - -65536, ячейки для записи чисел – [110: 1023], количество чисел не указывается, конец ряда обозначается символом «#».

Для Initial Orders 2 максимальное число для записи 65535, минимальное - -65536, ячейки для записи чисел – [146: 1023], количество чисел не указывается, конец ряда обозначается символом «#».

Вывод: В данной работе мы ознакомились с работой EDSAC и его двумя загрузчиками. Была разработана программа, использующая загрузчик Initial Orders 1, которая реализовала определенную функциональность. Кроме того, используя загрузчик Initial Orders 2, была выделена заданная функциональность в замкнутую подпрограмму и разработана вызывающая ее программа. Ожидаемые результаты совпали с результатами работы симулятора, значит программа работает корректно.