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

Низкоуровневое программирование

Отчет по лабораторной работе №2

Программирование EDSAC

Работу выполнил:
Аникин Д.А.
Группа:
3530901/90004
Преподаватель:
Алексюк А.О.

Санкт-Петербург
2021

Содержание

1. Формулировка задания

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.

Формулировка варианта задания

Интегрирование табличной функции методом трапеций с «длинным» результатом. *Примечание: переполнение разрядной сетки предотвращается пользователем масштабированием параметра шага сетки.*

$$\int_a^b f(x)\,dx = \sum_{i=0}^{n-1} \frac{f(x_i) + f(x_{i+1})}{2}(x_{i+1} - x_i)$$

Условные обозначения

В работе используются следующие обозначения:

1. C(n) - значение n-й ячейки памяти

2. C(Acc) - значение в аккумуляторе

3. C(R) - значение в регистре умножения

2. Программа для Initial Orders 1

Входные и выходные данные

Параметрами программы служат: C(101) - длина массивов *коротких* чисел X и Y, C(102) и C(103) - адреса массивов X и Y соответственно. Элементы массива X хранятся в C(104) - C(111), а элементы массива Y с C(112) по C(119).

Регулировка шага сетки предполагается модификацией значений X.

Результатом работы программы является значение определенного интеграла, полученное относительно X и Y, которое записывается в C(120)

Описание работы

При старте программы формируется значение $n-1$ (n - длина массива), использующееся в качестве счетчика итерации и записывается в рабочую ячейку C(1). Далее происходит инициализация адресных полей инструкций преобразования коротких чисел в длинные (необходимо для формирования длинного результата).

Листинг 1: Фрагмент IO1.txt

```
[START:]
    [31] T 120 S
    [32] X 0 S

    [Counter]
    [33] A 101 S
    [34] S 100 S
    [35] T 1 S

    [Adress fields]
    [36] A 102 S
    [37] L 0 L
    [38] A 57 S
    [39] T 57 S

    [40] A 102 S
    [41] L 0 L
    [42] A 60 S
    [43] T 60 S

    [44] A 103 S
    [45] L 0 L
    [46] A 63 S
    [47] T 63 S

    [48] A 103 S
    [49] L 0 L
    [50] A 66 S
    [51] T 66 S
```

Рабочий цикл программы можно разделить на три части: формирование длинных чисел, вычисление слагаемого по формуле и модификация адресных полей.

Формирование длинных чисел для x_{i+1}, x_i, y_{i+1} и y_i происходит путем загрузки соответствующих значений в аккумулятор, домножения на C(R) = 1, предварительно загруженную из C(100). После этого происходит коррекция полученных результатов путем сдвига на 2 разряда право. Результаты записываются в рабочие ячейки C(2), C(4), C(6) и C(8) для x_{i+1}, x_i, y_{i+1} и y_i соответственно.

Листинг 2: Фрагмент IO1.txt

```
30 [LOOP:]
31     [52] A 1 S
32     [53] S 100 S
33     [54] G 99 S
34     [55] T 1 S
36
36     [Load 1]
37     [56] H 100 S
39
39     [Long numbers]
40     [57] V 1 S [X_i+1]
41     [58] R 1 S
42     [59] T 2 L
44
44     [60] V 0 S [X_i]
45     [61] R 1 S
46     [62] T 4 L
48
48     [63] V 1 S [Y_i+1]
49     [64] R 1 S
50     [65] T 6 L
52
52     [66] V 0 S [Y_i]
53     [67] R 1 S
54     [68] T 8 L
```

При вычислении слагаемого сначала вычисляется значение шага ($x_{i+1}-x_i$). Результат записывается в рабочую ячейку C(10) и загружается в C(R). Далее вычисляется значение $f(x_{i+1}) + f(x_i)$, результат записывается в рабочую ячейку C(12). Затем происходит умножение C(12) на C(R) ($f(x_{i+1}) + f(x_i) \cdot (x_{i+1} - x_i)$), после этого производится коррекция умножения сдвигом числа в аккумуляторе влево, при этом деление на 2 здесь уже учтено (т.к деление на 2 осществляется сдвигом результата на 1 разряд вправо) для корректной записи числа в выходную ячейку.

Полученный результат суммируется с предыдущим из C(120) и записывается туда же.

Листинг 3: Фрагмент IO1.txt

```
56
56     [X_i+1 - X_i]
57     [69] A 2 L
58     [70] S 4 L
59     [71] T 10 L
60     [72] H 10 L
62
62     [Yi+1 + Y_i]
63     [73] A 6 L
```

```
64      [74] A 8 L
65      [75] T 12 L
67
67      [(Yi+1 + Y_i)/2 * (X_i+1 - X_i)]
68      [76] V 12 L
69      [77] L 1024 S
70      [78] L 1024 S
71      [79] L 128 S
73
73      [80] A 120 L
74      [81] T 120 L
```

Модификация адресных полей происходит путем прибавления единицы (C(100)) к адресу предыдущего элемента в инструкциях преобразования коротких чисел в длинные(C(57), C(60), C(63), C(66)) для перехода к следующим элементам массива. Цикл повторяется, пока не обнулится счетчик.

Листинг 4: Фрагмент IO1.txt

```
76
76      [82] A 100 S
77      [83] L 0 L
78      [84] A 57 S
79      [85] T 57 S
81
81      [86] A 100 S
82      [87] L 0 L
83      [88] A 60 S
84      [89] T 60 S
86
86      [90] A 100 S
87      [91] L 0 L
88      [92] A 63 S
89      [93] T 63 S
91
91      [94] A 100 S
92      [95] L 0 L
93      [96] A 66 S
94      [97] T 66 S
96
96      [98] E 52 S
```

Константы и входные данные

Листинг 5: Фрагмент IO1.txt

```
97  [EXIT:]
98      [99] Z 0 S
99  [CONST:]
100     [100] P 0 L [1]
```

```
101 [LEN:]
102     [101] P 4 S
103 [ADDR:]
104     [102] P 52 S
105     [103] P 56 S
106 [DATAX:]
107     [104] P 0 L [1]
108     [105] P 1 S [2]
109     [106] P 1 L [3]
110     [107] P 2 S [4]
111     [108] P 3 S [6]
112     [109] P 5 S [10]
113     [110] P 5 L [11]
114     [111] P 7 S [14]
115 [DATAY:]
116     [112] P 1 S [2]
117     [113] P 2 S [4]
118     [114] P 3 S [6]
119     [115] P 4 S [8]
120     [116] P 5 S [10]
121     [117] P 6 S [12]
122     [118] P 3 S [6]
123     [119] P 20 S [40]
```

3. Программа для Initial Orders 2

Программа для IO2 основана на программе для IO1. Функциональность варианта задания выделена в замкнутую подпрограмму, выходное значение записывается в C(200). Длина массивов хранится в C(10), адреса массивов X и Y в C(11) и C(12) соответственно. Элементы массивов X и Y хранятся в C(13) - C(20) и C(21) - C(28) соответственно.

«Каркас» замкнутой подпрограммы состоит из типовых директив, адресованных загрузчику Initial Orders 2, а также «пролога» (prologue) и «эпилога» (epilogue) – инструкций, исполняемых соответственно сразу после вызова и непосредственно перед возвратом из подпрограммы.

Листинг 6: IO2.txt

```
1  [START:]
2          T 56 K
3          G K
4      [0] A 3 F
5      [1] T 71 @
7
7      [2] A 1 F
8      [3] S 72 @
9      [4] T 1 F
11
```

```
11        [5] A 2 F
12    [6] X 0 F
13        [7] A 68 @
14        [8] T 26 @
16
16        [9] A 2 F
17    [10] X 0 F
18        [11] A 69 @
19        [12] T 29 @
21
21        [13] A 3 F
22    [14] X 0 F
23        [15] A 68 @
24        [16] T 32 @
26
26        [17] A 3 F
27    [18] X 0 F
28        [19] A 69 @
29        [20] T 35 @
30 [LOOP:]
31        [21] A 1 F
32        [22] S 72 @
33        [23] G 70 @
34        [24] T 1 F
36
36        [25] H 72 @
38
38        [26] V 1 F
39        [27] R 1 F
40        [28] T 4 D
42
42        [29] V 0 F
43        [30] R 1 F
44        [31] T 6 D
46
46        [32] V 1 F
47        [33] R 1 F
48        [34] T 8 D
50
50        [35] V 0 F
51        [36] R 1 F
52        [37] T 10 D
54
54        [38] A 4 D
55        [39] S 6 D
56        [40] T 12 D
57        [41] H 12 D
59
```

```
59      [42] A 8 D
60      [43] A 10 D
61      [44] T 14 D
62      [45] V 14 D
63      [46] L 1024 F
64      [47] L 1024 F
65      [48] L 128 F
67
67      [49] A 200 D
68      [50] T 200 D
70
70      [51] A 72 @
71      [52] L 0 D
72      [53] A 26 @
73      [54] T 26 @
75
75      [55] A 72 @
76      [56] L 0 D
77      [57] A 29 @
78      [58] T 29 @
80
80      [59] A 72 @
81      [60] L 0 D
82      [61] A 32 @
83      [62] T 32 @
85
85      [63] A 72 @
86      [64] L 0 D
87      [65] A 35 @
88      [66] T 35 @
90
90      [67] E 21 @
91  [INSTR:]
92      [68] V 1 F
93      [69] V 0 F
94  [EXIT:]
95      [70] T 0 F
96  [RET:]
97      [71] E 0 F
98  [CONST:]
99      [72] P 0 D [1]
102
102
102 [START:]
103         G K
104     [0] X 0 F
105     [1] A 10 @
106     [2] T 1 F
```

```
107       [3] A 11 @
108       [4] T 2 F
109       [5] A 12 @
110       [6] T 3 F
111       [7] A 7 @
112       [8] G 56 F
113       [9] Z 0 F
114 [LEN:]
115       [10] P 4 F
116 [ADDR:]
117       [11] P 13 @
118       [12] P 21 @
119 [DATAX]
120       [13] P 0 D [1]
121       [14] P 1 F [2]
122       [15] P 9 D [3]
123       [16] P 2 F [4]
124       [17] P 3 F [6]
125       [18] P 5 F [10]
126       [19] P 5 D [11]
127       [20] P 7 F [14]
128 [DATAY:]
129       [21] P 1 F [4]
130       [22] P 2 F [2]
131       [23] P 3 F [6]
132       [24] P 4 F [6]
133       [25] P 5 F [10]
134       [26] P 6 F [10]
135       [27] P 3 F [6]
136       [28] P 20 F [12]
137            EZ PF
```

4. Выводы

В ходе работы были разработаны программы для загрузчиков IO1 и IO2 EDSAC. OCновной проблемой при работе с IO1 является абсолютная адресация, из-за которой возникает необходимость пересчитывать адреса при любом изменении программы. Данный недостаток был исправлен в IO2 введением относительной адресации.