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: 12

Выполнил студент гр. 3530901/90002 _____Г.А. Сухоруков

(подпись)

Принял старший преподаватель _____Д. С. Степанов

(подпись)

“____” _____ 2021 г.

Санкт-Петербург

2021

Формулировка 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

По варианту номер 12 необходимо реализовать генерацию кода Грея заданной разрядности. Используется алгоритм зеркального двоичного кода Грея. Для получения кода разрядности n производится n шагов. На первом шаге код имеет разрядность 1 и состоит из двух векторов 0 и 1. На каждом следующем шаге в конец списка заносятся все уже имеющиеся вектора в обратном порядке, и затем к первой половине получившихся векторов дописывается 0, а ко второй 1.

Initial Orders 1

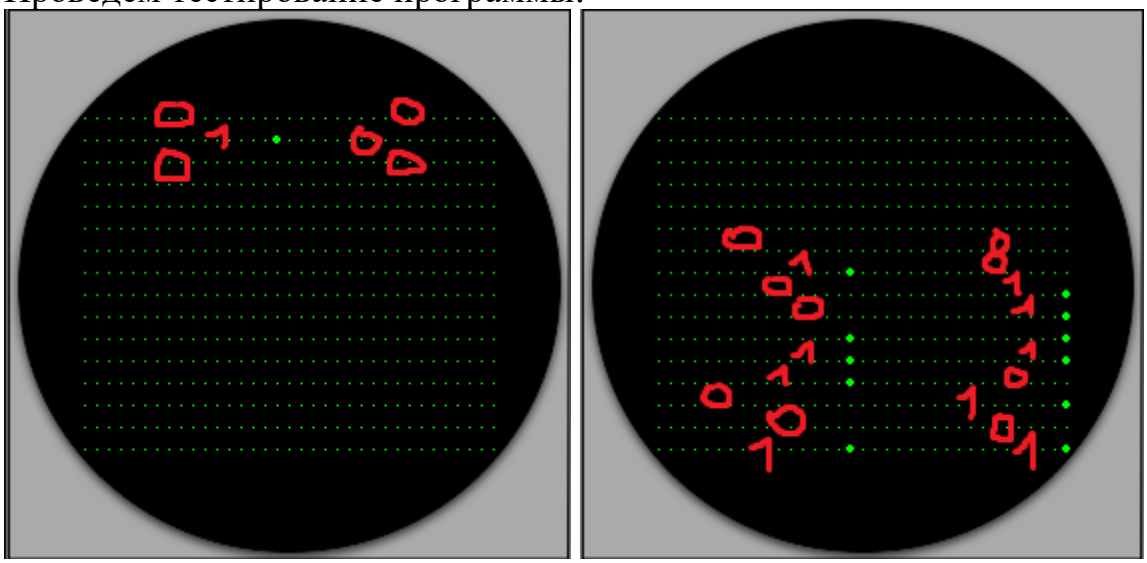
Алгоритм следующий:

- Сначала записываем код Грея разрядности 1.
- Затем “клонируем” в реверсивном порядке и приписываем слева в первую половину 0, а во вторую 1. Повторяем данный шаг до тех пор, пока не получим код Грея заданной разрядности

В начале программы записаны константы числа один, два, начальная ячейка вывода (250), заданная разрядность. И меняющиеся переменные: счетчик количества уже имеющихся кодов, количество кодов в (a – 1) коде Грея, счётчик по n для цикла отзеркаливания, счётчик по количеству кодов, счетчик по n для главного цикла.

Код с комментариями представлен в приложении 1.

Проведём тестирование программы:



Результаты начинаются с 250 ячейки, цифры читаются справа налево.

Initial Orders 2 и руководство программиста

Руководство программиста:

[44-47] Вызов подпрограммы.

[47-48] Переход в старт и конец программы.

[sub] - подпрограмма генерации кода грея заданной разрядности. Далее приведу пояснения по коду:

[0-13] Объявление констант и шаблонов инструкций.

[14-42] Клонирование.

[43-56] Старт программы, установка начального нуля, установка начальной единицы и запись статистики кодов в p.

[57-71] Главный цикл программы, после его завершения завершается программа.

[72-77] Цикл клонирования.

[78-88] Конец клонирования, переход в запись нулей и единиц.

[89-107] Запись нулей и единиц.

[108-111] Конец цикла клонирования, переход в главный цикл.

[112-115] Формирование инструкции возврата.

[116-118] Выход из подпрограммы.

Приложение 1

Листинг программы для загрузчика Initial Orders 1:

```
[Initial Orders 1]
[зеркальный двоичный код Грея]
 [31] T 145 F
 [32] E 46 S [переходим в начало программы]
[const1:]
 [33] P 0 L
[const2:]
 [34] P 1 S
[current:]
 [35] P 125 S [вывод с 250]
[n:]
 [36] P 1 L [разрядность кода грея]
[p:]
 [37] P 1 S [счетчик количества уже имеющихся кодов]
[t:]
 [38] P 0 S [показывает количество кодов в (a−1)-м коде Грея]
[j:]
 [39] P 0 S [счётчик j по n для цикла отзеркаливания]
[k:]
 [40] P 0 S [счётчик по p]
[clean:]
 [41] P 0 S [trash]
[ONE:]
 [42] P 1 S [1]
[i:]
 [43] P 0 L [счётчик по n]
[set:]
 [44] T 0 S
[get:]
 [45] A 0 S
[startC:]
 [46] A 35 S [взять current]
 [47] L 0 L [сдвиг на 1]
 [48] A 50 S [инициализация set1]
 [49] T 50 S [инициализация set1]
[set1:]
 [50] T 0 S [загузили в 250 ячейку 0]
 [51] A 35 S [взять current]
 [52] A 36 S [current + n]
 [53] L 0 L [сдвиг на 1]
```

```
 [54] A 57 S [инициализация set2]
 [55] T 57 S [инициализация set2]
 [56] A 33 S [взяли 1]
[set2:]
 [57] T 0 S [T (current + n)]
 [58] A 34 S [добавили 2]
 [59] T 37 S [записали в p]
[for:]
 [60] T 41 S [clena]
 [61] A 43 S [взяли i]
 [62] S 36 S [acc = i - n]
 [63] E 115 S [проверка если i - n >= 0 jump to endfor]
 [64] T 41 S [clena]
 [65] A 37 S [acc = p]
 [66] S 33 S [acc = p - 1]
 [67] T 38 S [t = p - 1]
 [68] A 37 S [acc = p]
 [69] T 40 S [k = p]
 [70] H 34 S [скопировали 2 в умножитель]
 [71] V 37 S [acc = 2 * p]
 [72] L 2 S [acc = 16 * (2 * p)]
 [73] L 0 S
 [74] T 37 S [p = acc]
[fork:]
 [75] T 41 S [clena]
 [76] A 40 S [acc = k]
 [77] S 37 S [acc = k - p]
 [78] E 111 S [проверка если k - p >= 0 конец цикла]
 [79] T 41 S [clena]
 [80] E 116 S [jump to clone]
 [endC:]
 [81] T 41 S [clean]
 [82] T 39 S [j = 0]
 [83] H 38 S [в умножитель t]
 [84] V 36 S [acc = t * n]
 [85] L 2 S [acc *= 16]
 [86] L 0 S
 [87] A 35 S [acc += 250]
 [88] A 43 S [acc += i]
 [89] L 0 L [acc *= 2]
 [90] A 44 S [acc += T0S]
 [91] T 92 S [init set3]
```

```
[set3:]
 [92] T 0 S [записали в нужную ячейку 0]
 [93] H 40 S [в умножитель k]
 [94] V 36 S [acc = k * n]
 [95] L 2 S [acc *= 16]
 [96] L 0 S
 [97] A 35 S [acc += 250]
 [98] A 43 S [acc += i]
 [99] L 0 L [acc *= 2]
 [100] A 44 S [acc += T0S]
 [101] T 103 S [init set4]
 [102] A 33 S [acc = 1]
[set4:]
 [103] T 0 S [записали в нужную ячейку 1]
 [104] A 38 S [acc = t]
 [105] S 33 S [acc = t - 1]
 [106] T 38 S [t = t - 1]
 [107] A 40 S [acc = k]
 [108] A 33 S [acc = k + 1]
 [109] T 40 S [k = k + 1]
 [110] E 75 S [jump to fork]
[endfork:]
 [111] A 43 S [acc = k - p + i]
 [112] A 33 S [acc += 1]
 [113] T 43 S [i = acc]
 [114] E 60 S [jump to for]
[endfor:]
 [115] Z 0 S [end of programm]
[clone:]
 [116] T 41 S [clean]
 [117] A 39 S [acc = j]
 [118] S 36 S [acc = j - n]
 [119] E 81 S [проверка если j = n переходим в endC]
 [120] T 41 S [clean]
 [121] H 40 S [скопировали k в умножитель]
 [122] V 36 S [acc = k * n]
 [123] L 2 S [сдвиг на четыре]
 [124] L 0 S
 [125] A 35 S [acc = 250 + acc]
 [126] A 39 S [acc = acc + j]
 [127] L 0 L [acc *= 2]
 [128] A 44 S [acc += T0S]
 [129] T 140 S [инит put1]
 [130] H 38 S [в умножитель t]
 [131] V 36 S [acc = n * t]
```

```
 [132] L 2 S [acc *= 16]
 [133] L 0 S
 [134] A 35 S [acc += current]
 [135] A 39 S [acc += j]
 [136] L 0 L [acc *= 2]
 [137] A 45 S [acc += A0S]
 [138] T 139 S [инит get1]
[get1:]
 [139] A 0 S
[put1:]
 [140] T 0 S [загрузили либо 1, либо 0]
 [141] A 33 S [acc = 1]
 [142] A 39 S [acc = 1 + j]
 [143] T 39 S [j = 1 + j]
 [144] E 116 S [jump to clone]
```

Приложение 2

Листинг программы для загрузчика Initial Orders 2:

```
[Initial Orders 2]
[44] P 48 F [96]
[start:]
[45] A 44 F [acc = 96]
[46] T 200 F [в ячейку 200 = 96]
[47] E 92 F [jump to startC]
[return:]
[48] Z 0 F [Конец]
GK [sub]
[const1:]
[0] P 0 D [1]
[const2:]
[1] P 1 F [2]
[current:]
[2] P 125 F [250]
[n:]
[3] P 1 D [разрядность кода грея]
[p:]
[4] P 1 F [счетчик количества уже имеющихся кодов]
[t:]
[5] P 0 F [показывает количество кодов в (a−1)-м коде Грея]
[j:]
[6] P 0 F [счётчик по j по n для цикла отзеркаливания]
[k:]
[7] P 0 F [счётчик по p]
[clean:]
[8] P 0 F [trash]
[ONE:]
[9] P 1 F [1]
[i:]
[10] P 0 D [счётчик по n]
[set:]
[11] T 0 F [T0F]
[get:]
[12] A 0 F [A0F]
[exit:]
[13] E 0 F [E0F]
```

```
[clone:]
[14] T 8 @ [clean]
[15] A 6 @ [acc = j]
[16] S 3 @ [acc = j - n]
[17] E 78 @ [проверка если j = n переходим в endC]
[18] T 8 @ [clean]
[19] H 7 @ [скопировали k в умножитель]
[20] V 3 @ [acc = k * n]
[21] L 2 F [сдвиг на четыре]
[22] L 0 F
[23] A 2 @ [acc = 250 + acc]
[24] A 6 @ [acc = acc + j]
[25] L 0 D [acc *= 2]
[26] A 11 @ [acc += T0S]
[27] T 38 @ [инит put1]
[28] H 5 @ [в умножитель t]
[29] V 3 @ [acc = n * t]
[30] L 2 F [acc *= 16]
[31] L 0 F
[32] A 2 @ [acc += current]
[33] A 6 @ [acc += j]
[34] L 0 D [acc *= 2]
[35] A 12 @ [acc += A0S]
[36] T 37 @ [инит get1]
[get1:]
[37] A 0 F
[put1:]
[38] T 0 F [записываем 0 или 1]
[39] A @ [acc = 1]
[40] A 6 @ [acc = 1 + j]
[41] T 6 @ [j = 1 + j]
[42] E 14 @ [jump to clone]
[startC:]
[43] A 2 @ [acc = current]
[44] L 0 D [сдвиг на 1]
[45] A 47 @ [init set1]
[46] T 47 @ [init set1]
[set1:]
[47] T 0 F [загузили в 250 ячейку 0]
[48] A 2 @ [взять current]
[49] A 3 @ [current + n]
[50] L 0 D [сдвиг на 1]
[51] A 54 @ [инициализация set2]
[52] T 54 @ [инициализация set2]
[53] A 0 @ [acc = 1]
```

```
[set2:]
[54] T 0 F [T (current + n) F]
[55] A 1 @ [acc = 2]
[56] T 4 @ [записали в p]
[for:]
[57] T 8 @ [clena]
[58] A 10 @ [взяли i]
[59] S 3 @ [acc = i - n]
[60] E 112 @ [проверка если i - n >= 0 jump to endfor]
[61] T 8 @ [clena]
[62] A 4 @ [acc = p]
[63] S 0 @ [acc = p - 1]
[64] T 5 @ [t = p - 1]
[65] A 4 @ [acc = p]
[66] T 7 @ [k = p]
[67] H 1 @ [скопировали 2 в умножитель]
[68] V 4 @ [acc = 2 * p]
[69] L 2 F [acc = 16 * (2 * p)]
[70] L 0 F
[71] T 4 @ [p = acc]
[fork:]
[72] T 8 @ [clena]
[73] A 7 @ [acc = k]
[74] S 4 @ [acc = k - p]
[75] E 108 @ [проверка если k - p >= 0 конец цикла]
[76] T 8 @ [clena]
[77] E 14 @ [jump to clone]
[endC:]
[78] T 8 @ [clean]
[79] T 6 @ [j = 0]
[80] H 5 @ [в умножитель t]
[81] V 3 @ [acc = t * n]
[82] L 2 F [acc *= 16]
[83] L 0 F
[84] A 2 @ [acc += 250]
[85] A 10 @ [acc += i]
[86] L 0 D [acc *= 2]
[87] A 11 @ [acc += T0S]
[88] T 89 @ [init set3]
```

```
[set3:]
[89] T 0 F [записали в нужную ячейку 0]
[90] H 7 @ [в умножитель k]
[91] V 3 @ [acc = k * n]
[92] L 2 F [acc *= 16]
[93] L 0 F
[94] A 2 @ [acc += 250]
[95] A 10 @ [acc += i]
[96] L 0 D [acc *= 2]
[97] A 11 @ [acc += T0S]
[98] T 100 @ [init set4]
[99] A 0 @ [acc = 1]
[set4:]
[100] T 0 F [записали в нужную ячейку 1]
[101] A 5 @ [acc = t]
[102] S 0 @ [acc = t - 1]
[103] T 5 @ [t = t - 1]
[104] A 7 @ [acc = k]
[105] A 0 @ [acc = k + 1]
[106] T 7 @ [k = k + 1]
[107] E 72 @ [jump to fork]
[endfork:]
[108] A 10 @ [acc = k - p + i]
[109] A 0 @ [acc += 1]
[110] T 10 @ [i = acc]
[111] E 57 @ [jump to for]
[endfor:]
[112] T 8 @ [clean]
[113] A 200 F [acc = 96]
[114] A 13 @ [acc += E0F]
[115] T 116 @ [init exit]
[exitT:]
[116] P 0 F [E45F]
[117] E 45 K
[118] P 0 F
```