Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Задание №2

Определение НОД всех элементов массива

Дисциплина: «Низкоуровневое программирование»

Выполнил студент гр. 3530901/90002 Сергиенко Н.И.

Преподаватель Степанов Д.С.

Санкт-Петербург
2021

Содержание

1 Описание задачи

Реализовать нахождение наибольшего общего делителя (НОД) для массива чисел при помощи EDSAC.

2 Алгоритм Евклида

Идея алгоритма заключается в том, что мы вычитаем из большего числа меньшее и заменяем первое на их разность до тех пор, пока их разность не станет равна нулю. В таком случае уменьшаемое и вычитаемое как раз и будут искомым числом.

$a = 35;\ b = 15$

$a)\ a - b = 35 - 15 = 20;\ a = 20$

$b)\ a - b = 20 - 15 = 5;\ a = 5$

$c)\ b - a = 15 - 5 = 10;\ b = 10$

$d)\ b - a = 10 - 5 = 5;\ b = 5$

$e)\ a - b = 5 - 5 = 0$

$GCD = 5$

3 Initial Orders 1

3.1 Переменные

В этом разделе опишем, какие значения нам необходимо хранить и где.

Прежде всего, нам понадобится обходить числа в цикле, а, следовательно, нужен счетчик. В качестве него в слово i положим количество элементов массива (константа [длина]), уменьшенное на один. Далее в цикле будут производиться вычисления, пока это значение не станет меньше нуля (таким образом, выполняется [длина − 1] итераций).

Для хранения результата будем использовать m[2] (2 S). Изначально это значение обнуляется.

[31]T 87 [конец программы] S

[32]Z 0 S

[Переменные]

[33]T 0 S

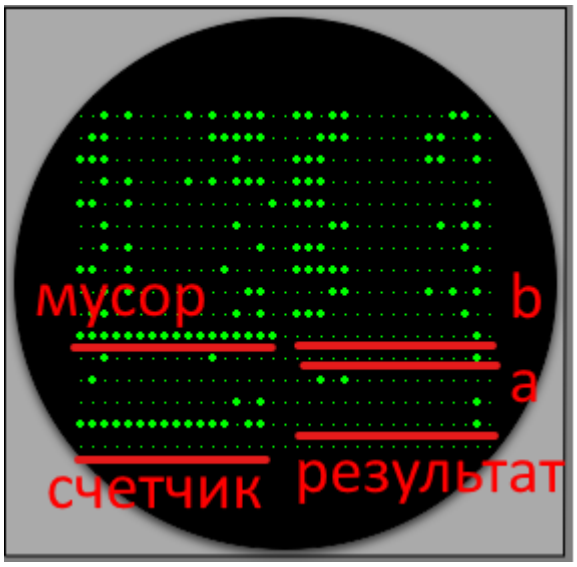
[34]A 79 [адрес где хранится длина] S

[35]S 77 [адрес единицы] S [длина-1 итераций]

[36]T 1 S [счетчик для кол-ва сравнений]

[37]T 2 S [ответ]

Так как вычисление необходимо провести в несколько шагов, то потребуются дополнительные переменные. Значение a будем хранить в $m[8]$,



b – в $m[10]$, а проверка (мусор) $-$ в $m[11]$. На рисунке ниже всё подписано.

Этот скриншот был сделан уже по окончании выполнения программы, поэтому $i = m[1]$ хранит значение “0”, $m[8]$ и $m[10]$ равны, и при этом в $m[11]$ записано число -1, означающее, что мы вычитали единицу из искомого нами нуля.

3.2 Основной цикл

Рассмотрим основной цикл программы, на каждой итерации которого получается новый НОД после добавления нового числа:

[46]A 1 S [загружаем счетчик]

[47]S 77 S [уменьшаем его на 1]

```
[48]G 86 S [если итог <0 завершаем работу]
[49]T 1 S  [обновили счетчик, очистили акк]
[первое число]
[50]A 0 S [загрузка первого значения в акк]
[51]T 8 S [выгрузка его в строку 8]
[второе число]
[52]A 0 S [загрузка второго значения в акк]
[53]T 10 S [выгрузка его в строку 10]
[Поиск НОД]
[54]A 8 S [значение первого числа в акк]
[55]S 10 S [вычитаем второе из первого]
[56]E 62 S [если результат вычитания >0 идем в 62 строку]
[если второе число больше первого]
[57]T 11 S [выгружаем ненужное значение]
[58]A 10 S [загружаем второе число]
[59]S 8 S  [вычитаем из него первое]
[60]T 10 S [выгружаем разность на место большего]
[61]E 54 S [продолжаем искать тк числа очевидно не равны]
[проверяем является ли рез-тат вычитания 0]
[62]S 77 S [вычили единицу]
[63]G 67 S [если был 0 идем в строку 67]
[пока что не НОД]
[64]A 77 S [если нет - возвращаем единицу]
[65]T 8 S  [выгружаем разность на место большего]
[66]E 54 S [возвращаемся к поиску НОД]
[нашли НОД]
[67]T 11 S [выкидываем лишнее]
[68]A 8 S  [загружаем НОД в акк]
[69]T 2 S  [выгружаем НОД в ячейку с ответом]
```

Рассмотрим, что тут происходит. Сначала мы смотрим, нужно ли нам еще с каким-то числом найти НОД, если нет - выходим. Дальше загружаем наши 2 исследуемых числа (или же найденный НОД и новое число). В следующих командах мы ищем НОД по алгоритму Евклида, вычитая из большего числа меньшее (если первое загружаемое в аккумулятор число меньше второго, то избавляемся от мусора и загружаем в аккумулятор второе число после чего вычитаем первое). Если первое число было изначально больше второго, то после вычитания проверяем не получилась ли разность

равной нулю, то есть для этого вычтем единицу и посмотрим - не отрицательный ли знак. Если всё еще не ноль, то возвращаем единицу и идём вычитать заново, если же оказался ноль, то скидываем из аккумулятора «-1» в $m[11]$ и загружаем любое из чисел ($m[8]$или $m[10]$), затем скидываем его в ответ - $m[2]$.

3.3 Обновляем адреса инструкций

[Начальная установка адресов инструкций]
[38]A 85 [первое число] S
[39]L 0 L
[40]A 50 S [добавляем инструкцию из 50 строки]
[41]T 50 S [выгружаем в 50 строку адрес 1 числа]
[42]A 86 [второе число] S
[43]L 0 L
[44]A 52 S[аналогично добавляем и потом выгружаем]
[45]T 52 S

После выполнения цикла необходимо сделать аналогичное обновление значений инструкций, только первым числом для вычитания теперь всегда будет текущий НОД. А вторым числом будет следующее значение из массива.

[70]A 68 S [инструкция с адресом текущего НОД]
[71]T 50 S [выгружаем ее в строку 50]
[72]A 77[единица] S
[73]L 0 L
[74]A 52 S [теперь инструкция берет следующий эл-т массива]
[75]T 52 S [выгружаем ее в 52 строку]
[76]E 46 S [возвращаемся к началу цикла]

3.4 Данные

Константы и прочие значения располагаются в конце программы.

[77]P 0 L [единица]
[78]P 1 L [двойка - нужна для проверки]
[79]P 2 L [длина массива - у нас 5]
[числа]
[80]P 9 S [18]

[81]P 18 S [36]

[82]P 12 S [24]

[83]P 15 S [30]

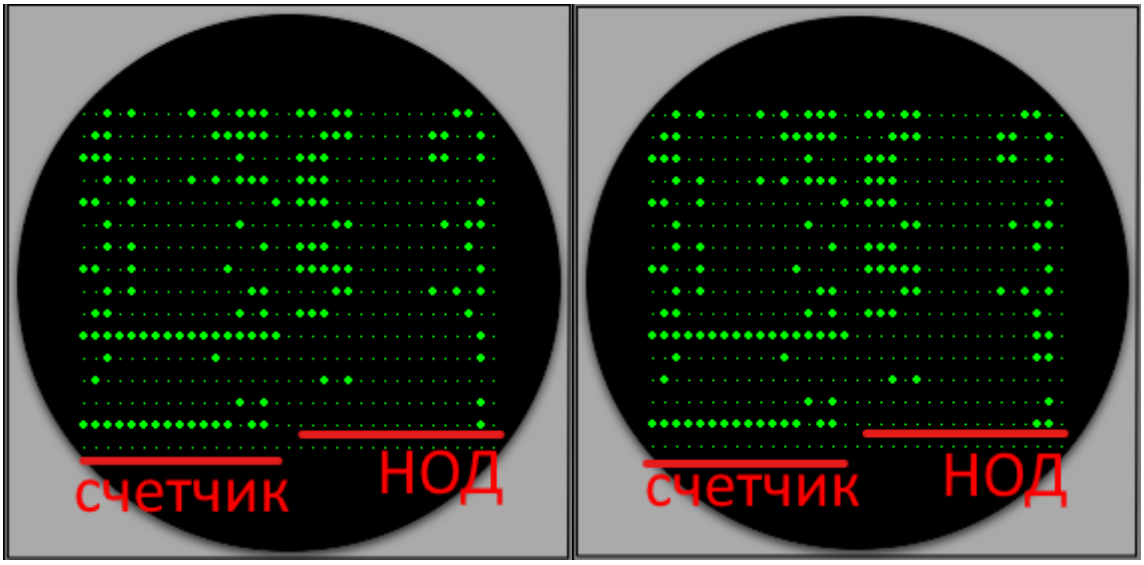
[84]P 20 S [40]

[85]P 40 S [номер строки для цифры 18]

[86]P 40 L [номер строки для цифры 36]

3.5 Проверка

Для проверки попросту возьмем и уменьшим количество итераций на 1, чтобы не брать последнее значение.



Как можно заметить, для пяти элементов {18, 36, 24, 30, 40} НОДом является число 2, но можно посчитать для первых четырех элементов и он окажется равным 6.

4 Initial Orders 2

4.1 Программа

Основным отличием Initial Orders 2 от Initial Orders 1 являются возможность написания подпрограмм и возможность использования относительной адресации.

Текст программы, которая вызывает подпрограмму и передает ей изначальные данные (координаты первого и второго числа, длину массива)

```
GK
[106] [0]A 9 @    [ЗАГРУЗКА В АКК ДЛИНЫ МАССИВА]
```

```
[107] [1]T 20 F    [ВЫГРУЗКА В РАБОЧУЮ ЯЧЕЙКУ]
[108] [2]A 15 @    [ЗАГРУЗКА В АКК АДРЕСА ПЕРВОГО НОМЕРА]
[109] [3]T 21 F    [ВЫГРУЗКА В РАБОЧУЮ ЯЧЕЙКУ]
[110] [4]A 16 @    [ЗАГРУЗКА В АКК АДРЕСА ВТОРОГО НОМЕРА]
[111] [5]T 22 F    [ВЫГРУЗКА В РАБОЧУЮ ЯЧЕЙКУ]
ZF
[112] [6]A 7 @     [ВХОД]
[113] [7]G 56 F    [В ПОДПРОГРАММУ]
[114] [8]
[115] [9]P 2 D     [ДЛИНА МАССИВА]
[116] [10]P 9 F                [18]
[117] [11]P 18 F           [36]
[118] [12]P 12 F           [24]
[119] [13]P 15 F           [30]
[120] [14]P 20 F           [40]
[121] [15]P 10 @           [НОМЕР СТРОКИ ПЕРВОГО ЧИСЛА]
[122] [16]P 11 @           [НОМЕР СТРОКИ ВТОРОГО ЧИСЛА]
EZ PF
```

4.2 Подпрограмма

Текст подпрограммы представляет собой переработанную программу для IO1 с использованием относительной адресации и заранее переданных при вызове данных

```
T 56 K
GK
[56] [0] A 3 F
[57] [1] T 47 @
[58] [2] T 0 F
[59] [3] A 20 F      [ДЛИНА]
[60] [4] S 48 @      [ДЛИНА-1 ДЛЯ КОРРЕКТНОГО ЧИСЛА ИТЕРАЦИЙ]
[61] [5] T 1 F       [СЧЕТЧИК]
[62] [6] T 2 F         [РЕЗУЛЬТАТ]

        [ОБНОВЛЕНИЕ АДРЕСОВ ИНСТРУКЦИЙ]

[63] [7]  A 21 F
[64] [8]  XF
[65] [9]  A 19[ЗАГРУЗКА ПЕРВОГО] @
[66] [10] T 19[ЗАГРУЗКА ПЕРВОГО] @
[67] [11] A 22 F
```

```
[68] [12] XF
[69] [13] A 21[ЗАГРУЗКА ВТОРОГО] @
[70] [14] T 21[ЗАГРУЗКА ВТОРОГО] @

        [START OF LOOP]

[71] [15] A 1 F         [ЗАГРУЗКА СЧЕТЧИКА В АКК]
[72] [16] S 48 @        [ДЕКРЕМЕНТ СЧЕТЧИКА]
[73] [17] G 47 @            [МЕНЬШЕ 0 - НА ВЫХОД]
[74] [18] T 1 F                 [ПЕРЕЗАПИСЬ СЧЕТЧИКА]
[75] [19] A 0 F                 [ЗАГРУЗКА ПЕРВОГО ЧИСЛА]
[76] [20] T 8 F                 [ВЫГРУЗКА ПЕРВОГО ЧИСЛА В 8 ЯЧЕЙКУ]
[77] [21] A 0 F                 [ЗАГРУЗКА ВТОРОГО ЧИСЛА]
[78] [22] T 10 F            [ВЫГРУЗКА ВТОРОГО ЧИСЛА В 10 ЯЧЕЙКУ]

        [CALCULATE GCD]

[79] [23] A 8 F                 [ЗАГРУЗКА ПЕРВОГО В АКК]
[80] [24] S 10 F            [ВЫЧИТАНИЕ ИЗ НЕГО ВТОРОГО]
[81] [25] E 31 @            [ПРОВЕРКА ЗНАКА ВЫЧИТАНИЯ]
        [SECOND>FIRST]
[82] [26] T 11 F            [ОЧИСТИЛИ АКК]
[83] [27] A 10 F            [ЗАГРУЗКА ВТОРОГО В АКК]
[84] [28] S 8 F                 [ВЫЧИТАНИЕ ИЗ НЕГО ПЕРВОГО]
[85] [29] T 10 F            [ВЫГРУЗКА РАЗНОСТИ НА МЕСТО ВТОРОГО]
[86] [30] E 23 @            [ВЕРНУЛИСЬ К ВЫЧИТАНИЮ]
        [CHECK IF ZERO]
[87] [31] S 48 @        [ВЫЧЛИ 1]
[88] [32] G 36 @                [НЕ НОД - ИДЕМ ДАЛЬШЕ]
[89] [33] A 48 @        [ВЕРНУЛИ 1]
[90] [34] T 8 F                 [ВЫГРУЗИЛИ РАЗНОСТЬ]
[91] [35] E 23 @            [ПРОДОЛЖАЕМ ВЫЧИТАТЬ]
        [FOUND GCD]
[92] [36] T 11 F            [ВЫКИНУЛИ МУСОР]
[93] [37] A 8 F                 [НОД В АКК]
[94] [38] T 2 F                 [ВЫГРУЗКА НОД]

        [ОБНОВЛЕНИЕ АДРЕСОВ ИНСТРУКЦИЙ]

[95] [39] A 37 @        [ИНСТРУКЦИЯ С АДРЕСОМ ПОСЛЕДНЕГО НОД]
[96] [40] T 19 @            [ЗАПИСАЛИ В 1 ЗНАЧЕНИЕ]
[97] [41] A 48 @
[98] [42] XF                [ДОБАВИЛИ ДВА]
```

```
[99] [43] A 21 @          [ДОБАВИЛИ ПРЕДЫДУЩУЮ ИНСТРУКЦИЮ]
[100] [44] T 21 @               [ВЫГРУЗИЛИ НОВОЕ ЗНАЧЕНИЕ]
[101] [45] E 15 @               [ВЕРНУЛИСЬ В НАЧАЛО ЦИКЛА]
[102] [46] T 11 F
[103] [47] E 0 F          [ВЫХОД ИЗ ПОДПРОГРАММЫ]
      [ДАННЫЕ]
[104] [48] P 0 D          [ЕДИНИЦА]
[105] [49] P 1 F          [ДВОЙКА]
```

5 Итого

Являясь одной из первых ЭВМ, EDSAC может выполнять широкий спектр задач, несмотря на ограничения, вызванные неудобством программирования и малой вычислительной мощностью.