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 2](#_Toc34766982)

[2 Алгоритм Евклида 2](#_Toc34766983)

[3 Initial Orders 1 2](#_Toc34766984)

[3.1 Переменные 2](#_Toc34766985)

[3.2 Основной цикл 4](#_Toc34766986)

[3.3 Обновляем адреса инструкций 5](#_Toc34766987)

[3.4 Данные 5](#_Toc34766988)

[3.5 Проверка 6](#_Toc34766989)

[4 Итого](#_Toc34766996) 6

# **Описание задачи**

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

# **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.

# **Initial Orders 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 положим количество элементов массива (константа ), уменьшенное на один. Далее в цикле будут производиться вычисления, пока это значение не станет меньше нуля (таким образом, выполняется 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]

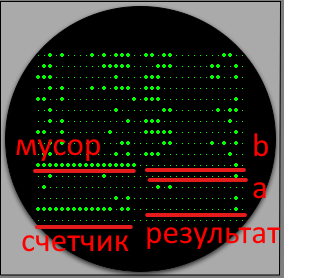
[33]T 0 S

[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 будем хранить в , – в , . На рисунке ниже всё подписано.

Этот скриншот был сделан уже по окончании выполнения программы, поэтому хранит значение “”, равны, и при этом в записано число , означающее, что мы вычитали единицу из искомого нами нуля.

## **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 и загружаем любое из чисел (), затем скидываем его в ответ - .

## **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]

## **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]

## **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.

# **Итого**

Являясь по сути первой ЭВМ, EDSAC может выполнять широкий спектр задач, несмотря на ограничения, вызванные неудобством программирования и малой вычислительной мощностью.