**Министерство науки и высшего образования РФ**

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра радиоэлектронных средств

Отчет по лабораторной работе №1

«Использование математического микропроцессора»

Дисциплина «Цифровые устройства и микропроцессоры»

Вариант № 19

Выполнил: студент группы ИНБс-3301-01-00 \_\_\_\_\_\_\_\_\_\_\_\_ В.О. Игнатович

Проверил: преподаватель кафедры РЭС \_\_\_\_\_\_\_\_\_\_\_\_ М.А. Земцов

Киров 2025

**1 Цели и задачи**

Цель работы — изучение принципов выполнения арифметических команд с помощью математического сопроцессора FPU МП с x86 архитектурой.

Задачи:

* реализовать решение;
* сформировать тестовый набор значений;
* произвести проверку полученных значений.

**2 Общий ход решения задач, анализ результата работы алгоритма**

Задание по варианту №19 приведено на рисунке 1.

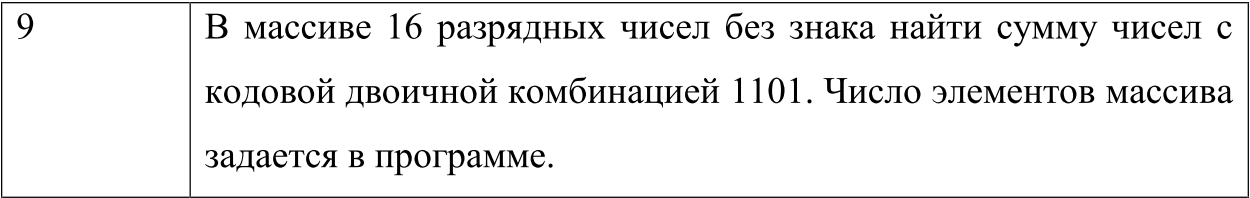


Рисунок 1 — Задания по варианту

Для решения поставленных задач был составлен тестовый набор, а именно массив из 11 целочисленных 16-битовых элементов: 13, 29, 221, 5389, 12345, 56789, 45, 0, 1024, 255, 43690, 32768, 8191.

Анализ данных элементов массива по поставленному по заданию условию представлен в таблице 1.1

Таблица 1.1 — Анализ подобранных значений.

|  |  |  |
| --- | --- | --- |
| **Число** | **Двоичная запись** | **Содержание комбинации** |
| 13 | 1101 | Содержит |
| 29 | 11101 | Содержит |
| 221 | 11011101 | Содержит |
| 5389 | 1010100001101 | Содержит |
| 12345 | 11000000111001 | — |
| 56789 | 1101110111010101 | Содержит |
| 45 | 101101 | Содержит |
| 0 | 0 | — |
| 1024 | 10000000000 | — |
| 255 | 11111111 | — |
| 43690 | 1010101010101010 | — |
| 32768 | 1000000000000000 | — |
| 8191 | 1111111111111 | — |

С учётом сформированных значений сумма подходящих значений будет равна 231213, что представлено на рисунке 1.1.

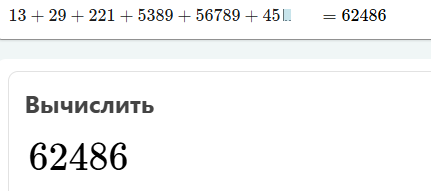


Рисунок 1.1 — Сумма подходящих элементов массива

Для решения данной задачи был написан программный код для языка ассемблера (MASM) в среде программирования Visual Studio 2022 с использованием шаблона «Консольное приложение С++».

По заданию преподавателя была добавлена возможность ввода/вывода значений переменных через консоль.

Конечный результат работы данного алгоритма представлен на рисунке 1.2.1.

Значения, полученные через ручной анализ и с помощью программы на языке ассемблера, оказались равны, что позволило подтвердить корректность работы составленного алгоритма.

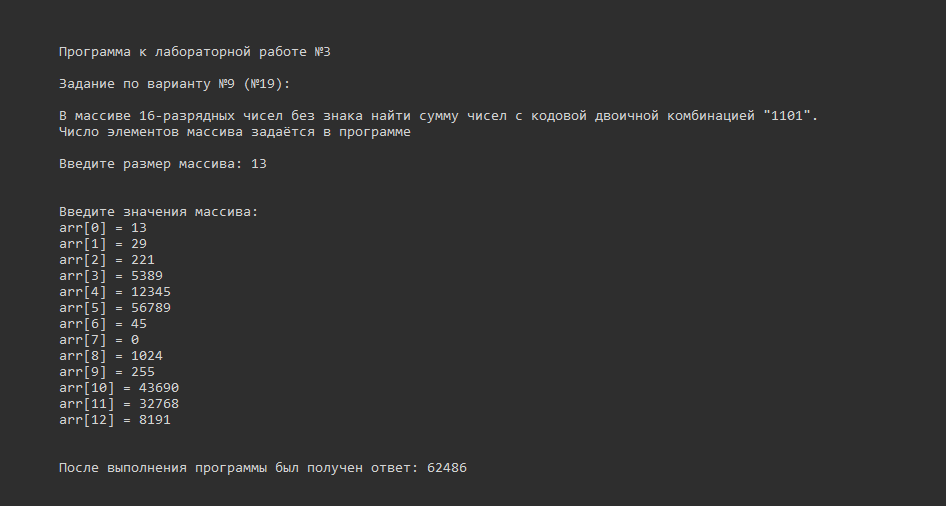


Рисунок 1.2.1 — Результат вычислений алгоритма

1. **Анализ работы алгоритма**

Алгоритм работы программы был составлен следующим образом: если текущие 4 бита значения текущего элемента массива равны 13 (что эквивалентно двоичной комбинации «1101»), то прибавить значение элемента массива к переменной «res» и перейти к следующему элементу массива; иначе взять следующие 4 бита; повторять 16 – 4 + 1 = 13 раз.

Далее была рассмотрена работа данного алгоритма на примере элемента по позиции 2 в массиве (значение элемента равно 221).

Стоит отметить, что приведённое выше число в своей двоичной форме записи представляет из себя набор из двух подряд идущих строк «1101», то есть оно равно числу 11011101b; если анализировать данное число по алгоритму, то необходимо учитывать, что нужная комбинация представлена в числе дважды, и не использовать число два раза.

Далее были рассмотрены значения в ключевых точках системы на этапах работы алгоритма над данным элементом массива.

Код цикла, обрабатывающего значение элемента массива, представлен на рисунке 1.3.1.

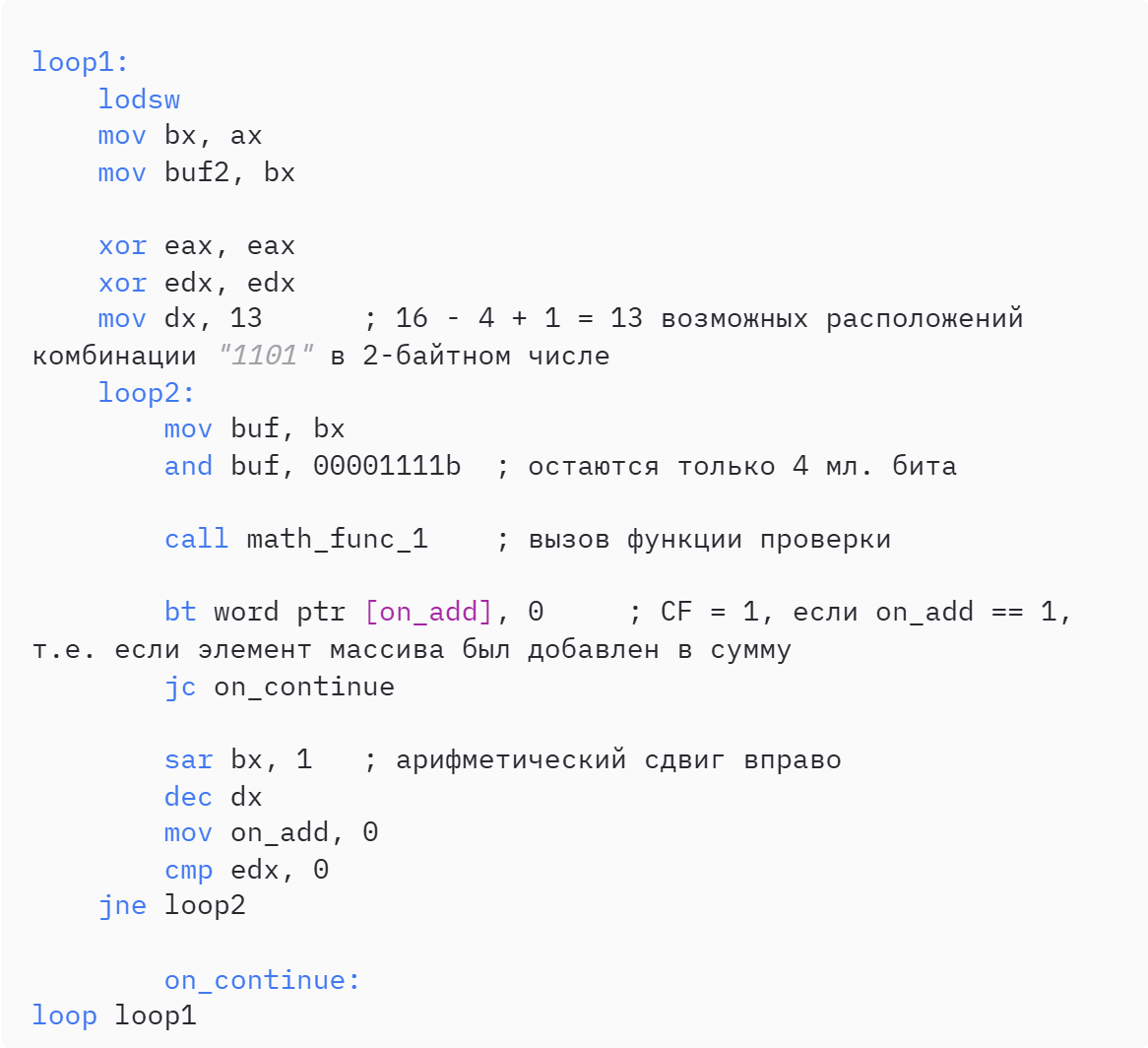


Рисунок 1.3.1 — Код цикла

Работа внешнего цикла loop1 начинается с команды “lodsw”, которая загружает значение элемента массива в регистр ax, после чего это значение сохраняется в регистр bx для последующей обработки и в переменную buf2 для сохранения значения при дальнейшем обращении в коде.

Далее, после очищения регистров eax, edx, в регистр dx загружается значение 13, равное проверочной комбинации «1101» в двоичной СС, и начинается тело внутреннего цикла loop2, который выполняется до тех пор, пока сравнение регистра edx с нулём возвращает значение «больше». Далее были описаны действия на каждой итерации внутреннего цикла.

Первым шагом значение регистра bx, которое равно значению текущего элемента массива, копируется в переменную buf, над которой производится побитовое логическое «И» с числом «1111» в 2 ОС, после чего в переменной остаётся значение только 4 младших бит исходного числа.

Далее вызывается функция обработки math\_cad\_1, код которой представлен на рисунке 1.3.2. При этом если содержимое переменной buf окажется равно в своей двоичной форме числу 1101, то после выполнения данной функции исходное анализируемое число прибавится в значении к переменной res, хранящей в себе сумму подходящих чисел, и переменная on\_add станет равна 1, что в коде внутреннего цикла loop2 будет проверяться сразу после вызова функции. Если значение данной переменной окажется равно 1, то будет осуществлён переход на точку on\_continue, в следствие чего работа внутреннего цикла будет окончена, а значит, внешний цикл начнёт обработку следующего элемента массива. В ином случае будет произведён арифметический сдвиг вправо регистра bx и внутренний цикл повторит свою работу.

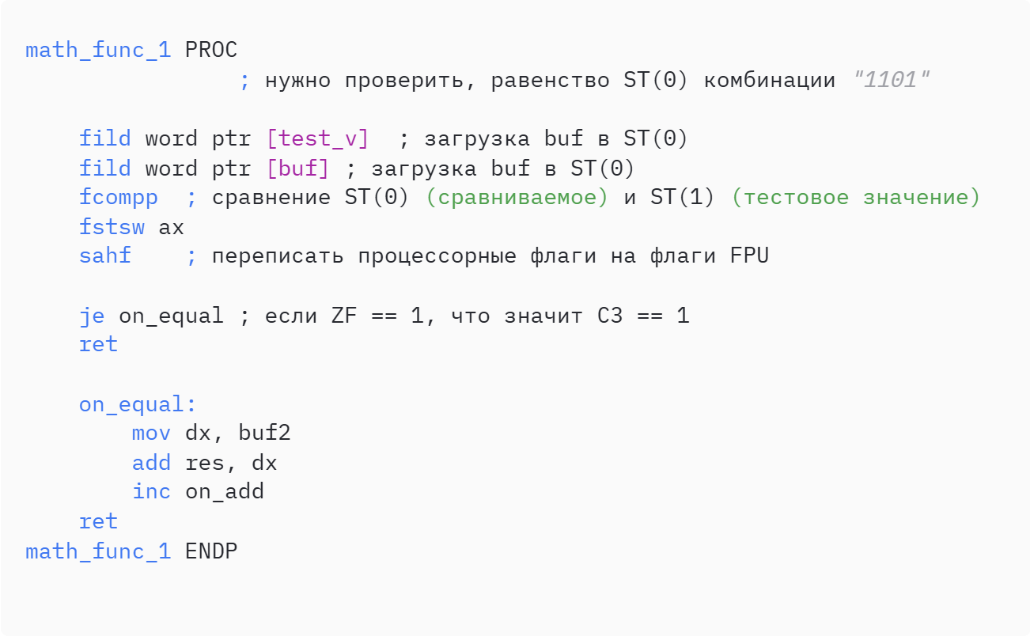


Рисунок 1.3.2 — Код функции обработки

Функция обработки числа math\_func\_1 позволяет проверять число, хранящееся в переменной buf, на соответствие со значением переменной test\_v, равной 13 (1101 в 2 ОС).

Данная функция выполняет свою функцию посредством взаимодействия с математическим сопроцессором, который оперирует 8 регистрами, хранящими значения с плавающей точкой, стека ST.

После загрузки обеих переменных в верхние регистры стека производится операция их сравнения. В случае их равенства выполняется переход на точку on\_equal, код которой позволяет прибавить число к переменной res и приравнять значение переменной on\_add к единице. В случае неравенства значение данной переменной останется равным нулю, и больше ничего не произойдёт вплоть до окончания работы функции.

**4 Вывод**

В ходе выполнения данной лабораторной работы были реализованы навыки работы с математическим сопроцессором на языке программирования ассемблера.

Код программы представлен на рисунке 2.

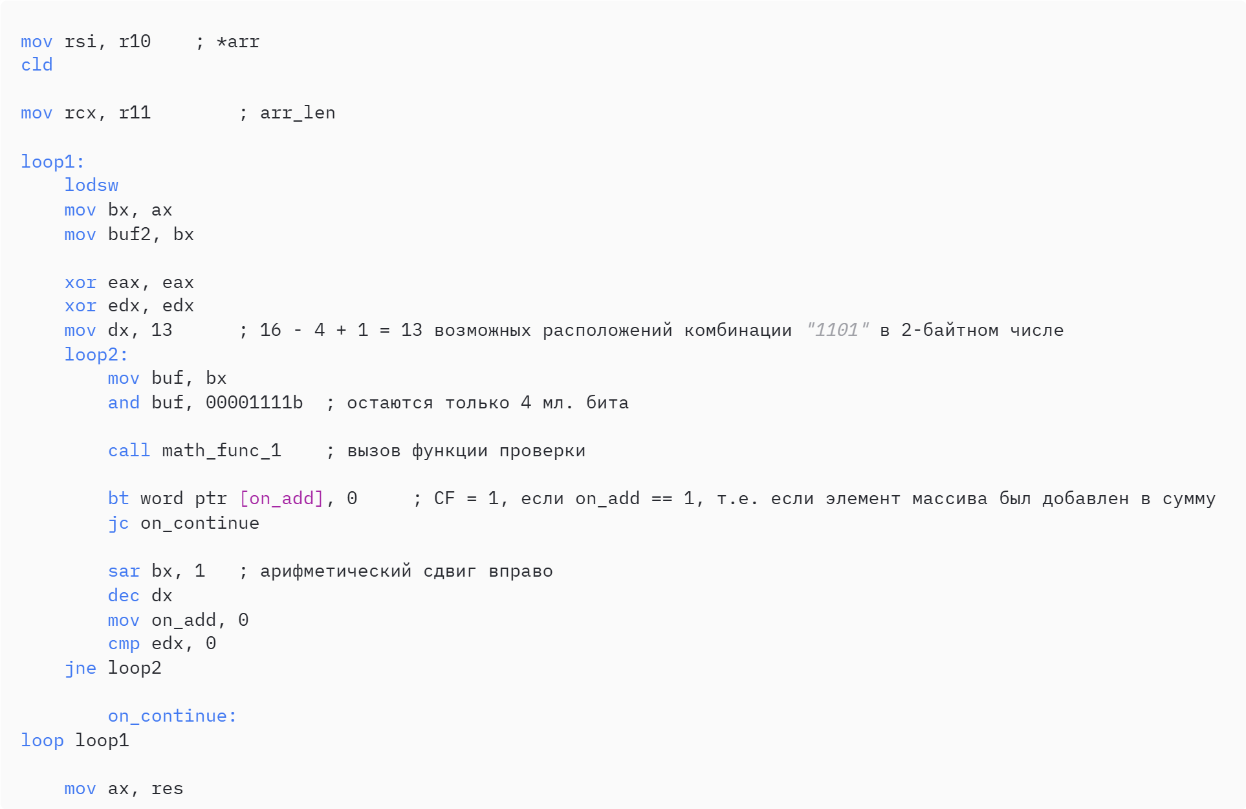


Рисунок 2 — Код программы