Отчёт по лабораторной работе №9

Программирование цикла. Обработка аргументов командной строки.

Поляков Глеб Сергеевич

Содержание

# 1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

# 2 Задание

1. Создать каталог и файл lab9-1.asm
2. Написать текст программы lab9-1
3. Создать каталог и файл lab9-2.asm
4. Написать текст программы lab9-2
5. Создать каталог и файл lab9-3.asm
6. Написать текст программы lab9-3
7. Создать каталог и файл lab9-4.asm
8. Написать текст программы lab9-4

# 3 Теоретическое введение

## 3.1 Организация стека

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды.

Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

На рис. 9.1 показана схема организации стека в процессоре.

Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается.

Для стека существует две основные операции:

* добавление элемента в вершину стека (push);
* извлечение элемента из вершины стека (pop).

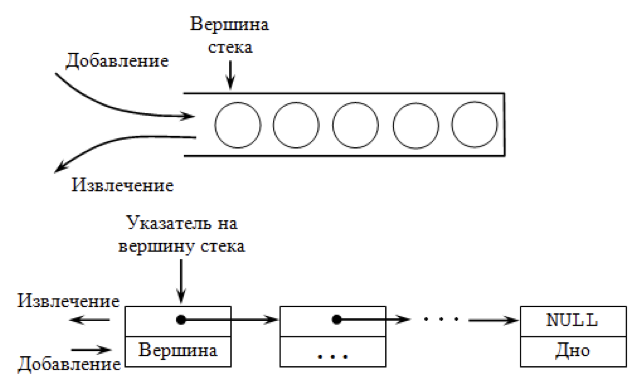


Рис. 1: Организация стека в процессоре

### 3.1.1 Добавление элемента в стек.

Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Данная команда имеет один операнд — значение, которое необходимо поместить в стек. Примеры:

push -10 ; Поместить -10 в стек  
push ebx ; Поместить значение регистра ebx в стек  
push [buf] ; Поместить значение переменной buf в стек  
push word [ax] ; Поместить в стек слово по адресу в ax

Существует ещё две команды для добавления значений в стек. Это команда pusha, которая помещает в стек содержимое всех регистров общего назначения в следующем порядке: ах, сх, dx, bх, sp, bp, si, di. А также команда pushf, которая служит для перемещения в стек содержимого регистра флагов. Обе эти команды не имеют операндов.

### 3.1.2 Извлечение элемента из стека.

Команда pop извлекает значение из стека, т.е. извлекает значение из ячейки памяти, на которую указывает регистр esp, после этого уменьшает значение регистра esp на 4. У этой команды также один операнд, который может быть регистром или переменной в памяти.

Нужно помнить, что извлечённый из стека элемент не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек.

Примеры:

pop eax ; Поместить значение из стека в регистр eax  
pop [buf] ; Поместить значение из стека в buf  
pop word[si] ; Поместить значение из стека в слово по адресу в si

Аналогично команде записи в стек существует команда popa, которая восстанавливает из стека все регистры общего назначения, и команда popf для перемещения значений из вершины стека в регистр флагов. ## Инструкции организации циклов Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре ecx. Наиболее простой является инструкция loop. Она позволяет организовать безусловный цикл, типичная структура которого имеет следующий вид:

mov ecx, 100  
NextStep:  
 ...  
 ... ; тело цикла  
 ...  
 loop NextStep ; Повторить `ecx` раз от метки NextStep

Иструкция loop выполняется в два этапа. Сначала из регистра ecx вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды loop.

# 4 Выполнение лабораторной работы

1. Создал каталог и файл lab9-1.asm
2. Ввел текст программы в файл (рис. 2)

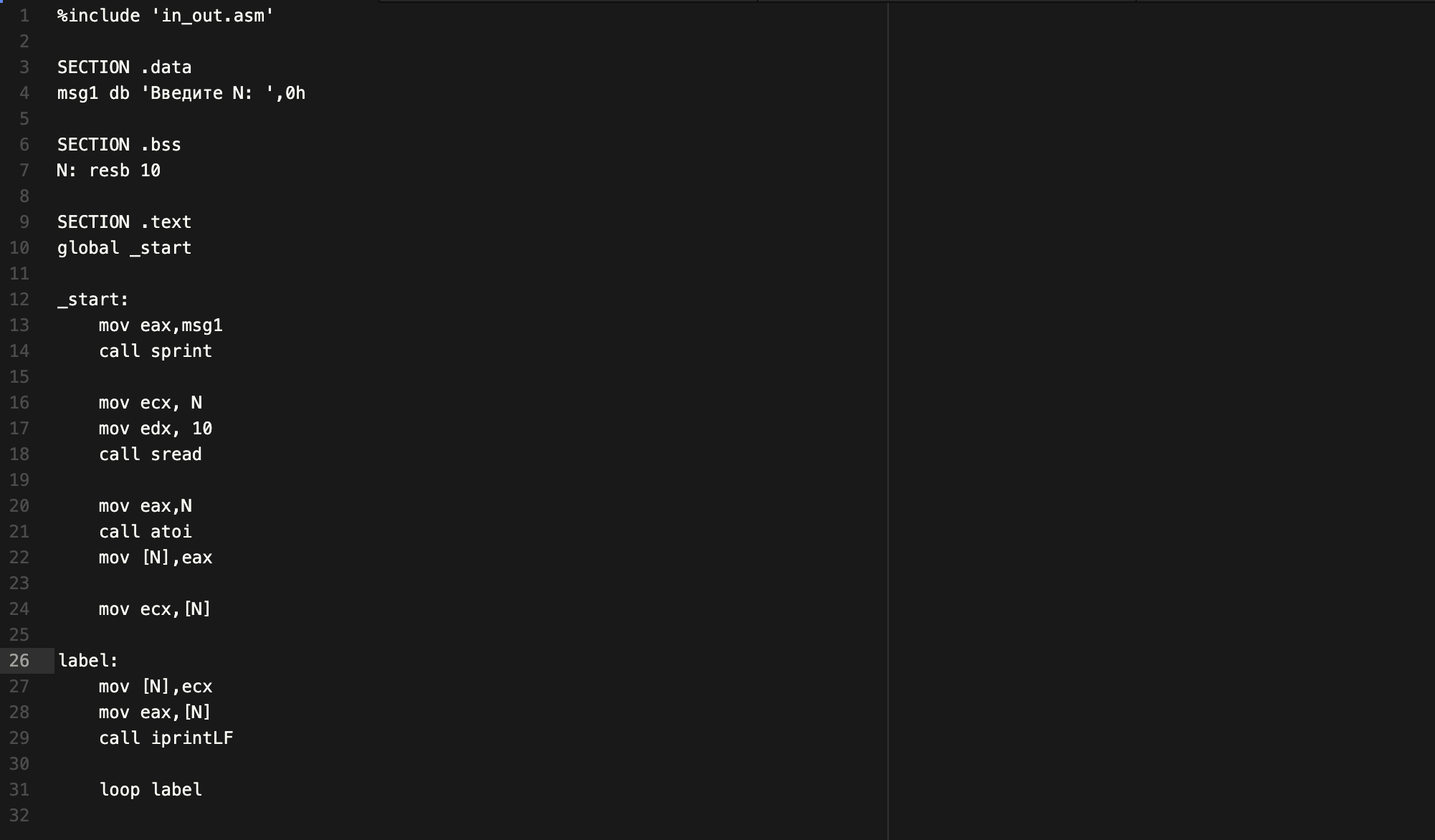


Рис. 2: Текст программы lab9-1.asm

1. Проверил работоспособность программы (рис. 3)

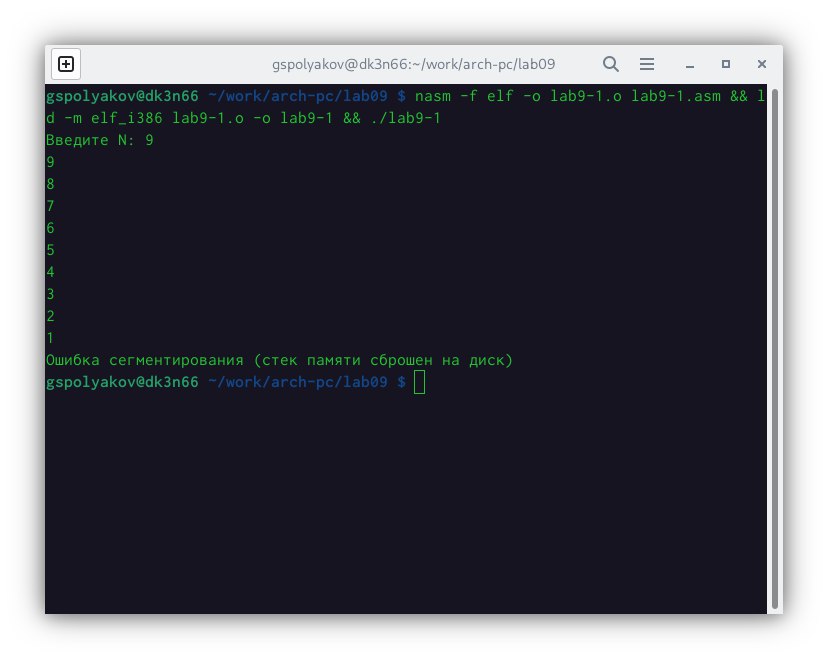


Рис. 3: Результат программы lab9-1.asm

1. Также проверил работу программы с изменениями (рис. 4) и (рис. 5), регистр ecx принимает значения 9-1, число проходов соответствует значению N.

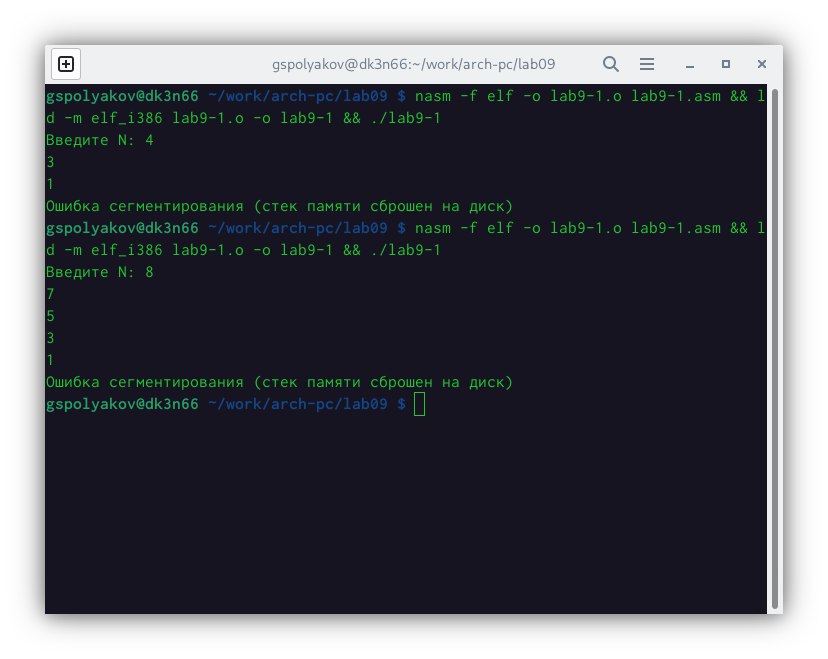


Рис. 4: Результат программы lab9-1.asm

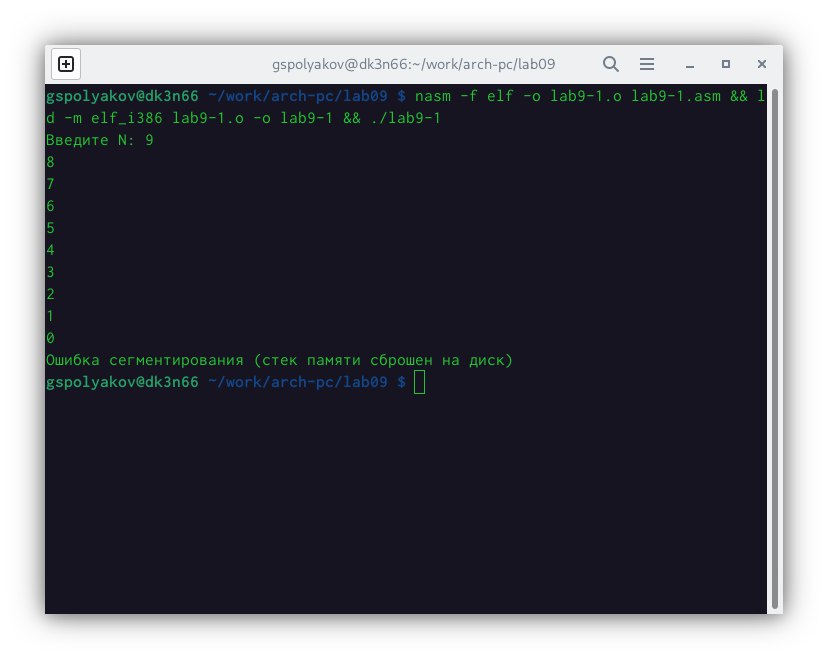


Рис. 5: Результат программы lab9-1.asm

1. Создал файл lab9-2.asm
2. Ввел текст программы в файл (рис. 6)

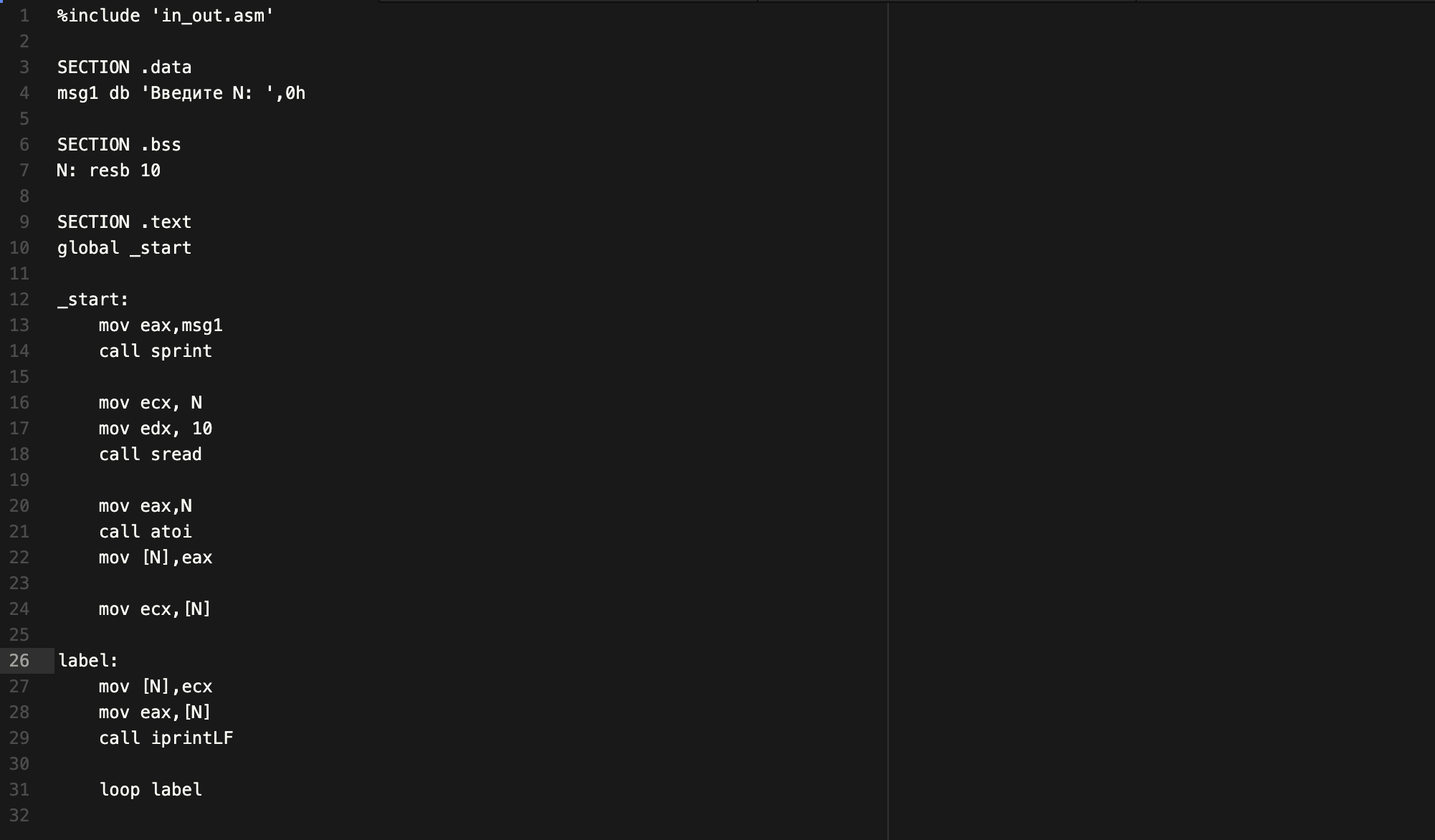


Рис. 6: Текст программы lab9-2.asm

1. Также проверил работу программы с изменениями (рис. 7), программа обработала 3 аргумента.

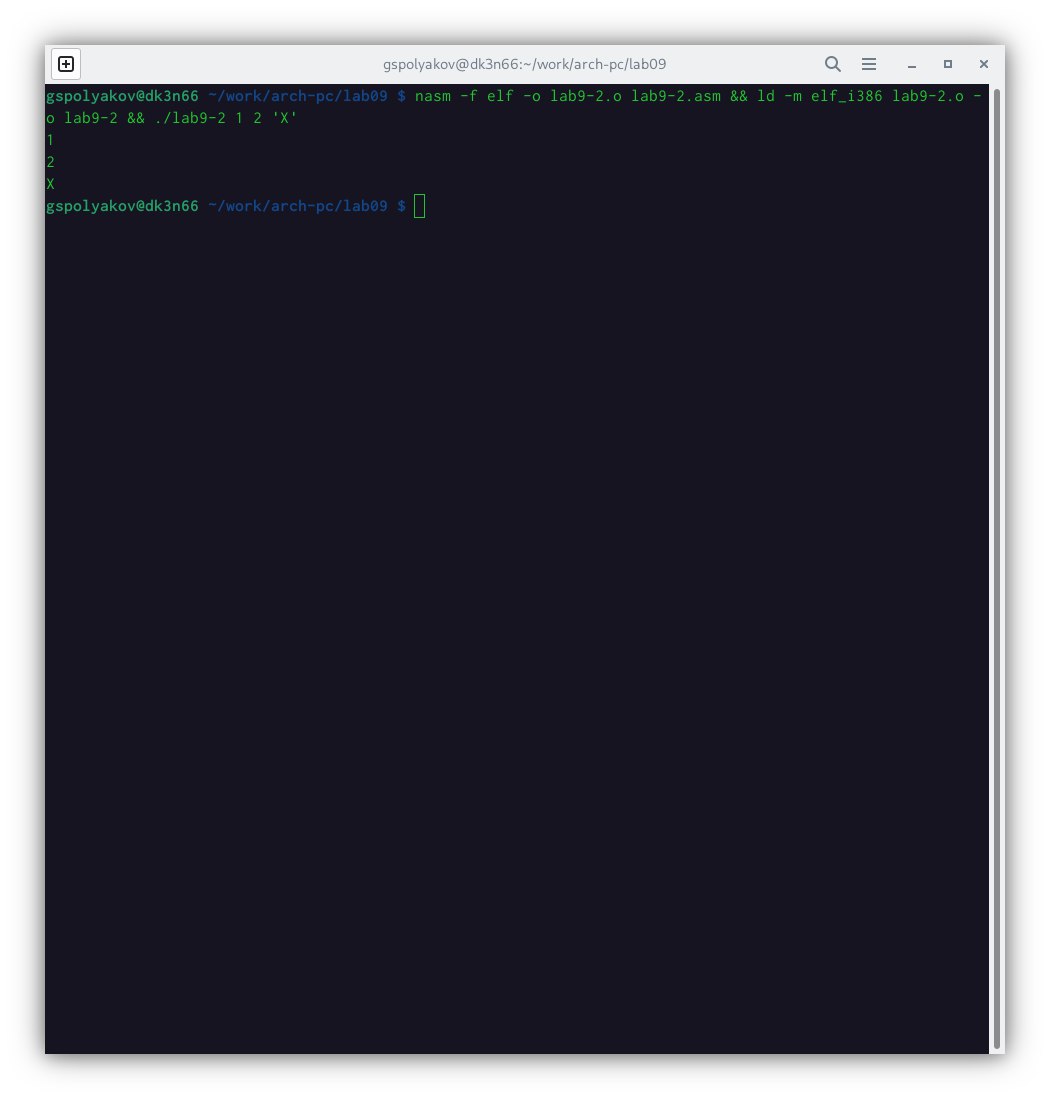


Рис. 7: Результат программы lab9-2.asm

1. Создал файл lab9-3.asm
2. Ввел текст программы в файл (рис. 8)

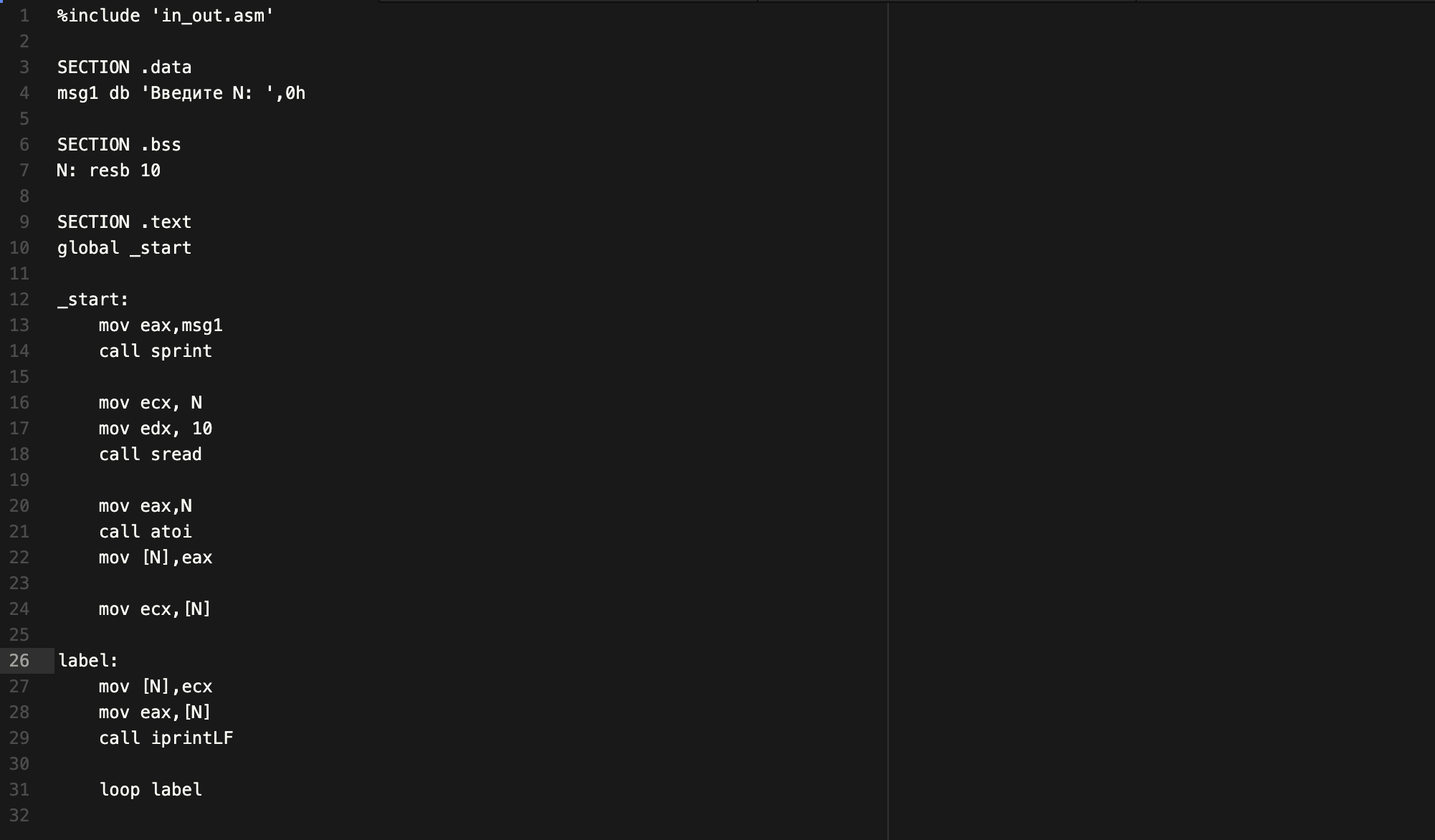


Рис. 8: Текст программы lab9-3.asm

1. Проверил работоспособность программы (рис. 9).

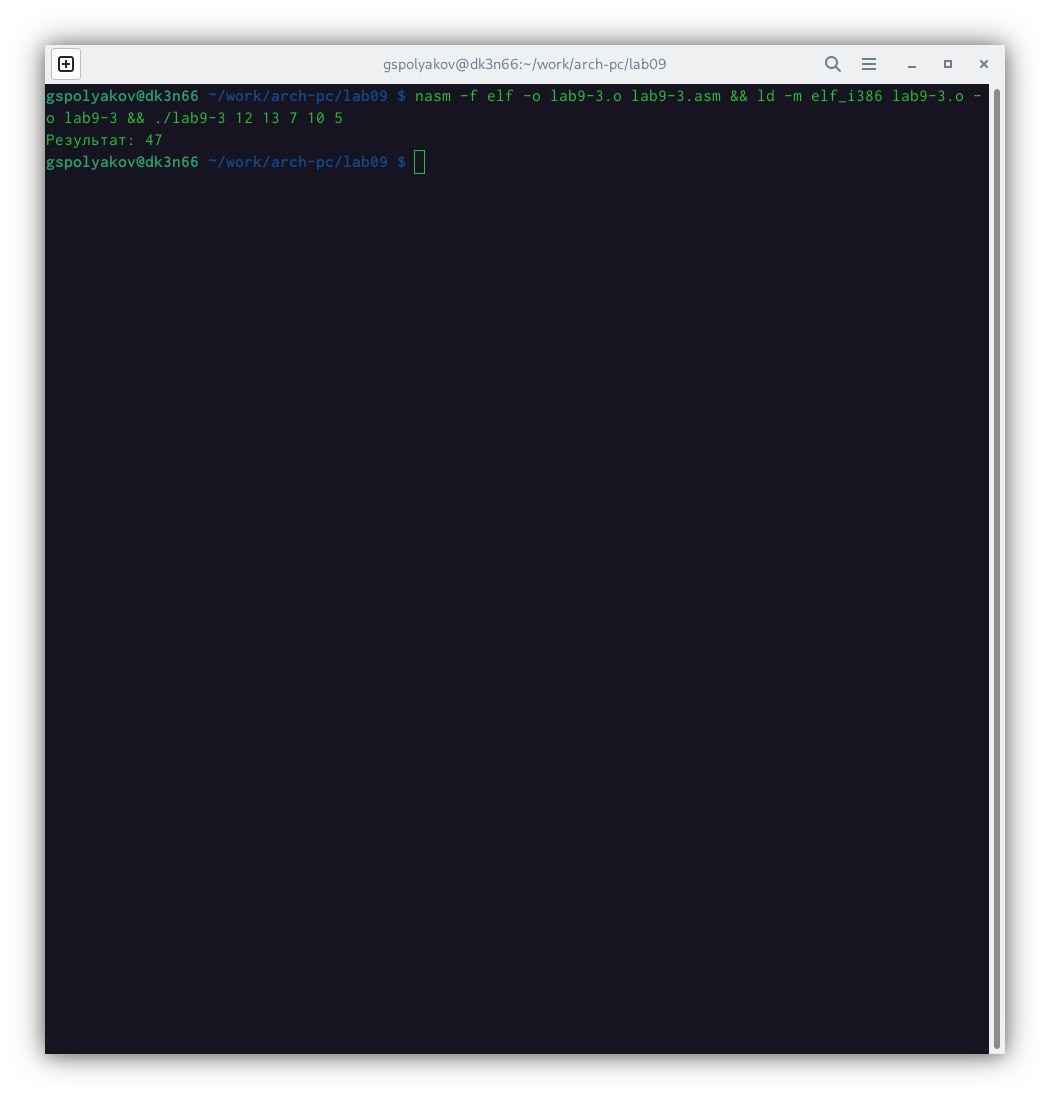


Рис. 9: Результат программы lab9-3.asm

1. Также проверил работу программы с изменениями (рис. 10).

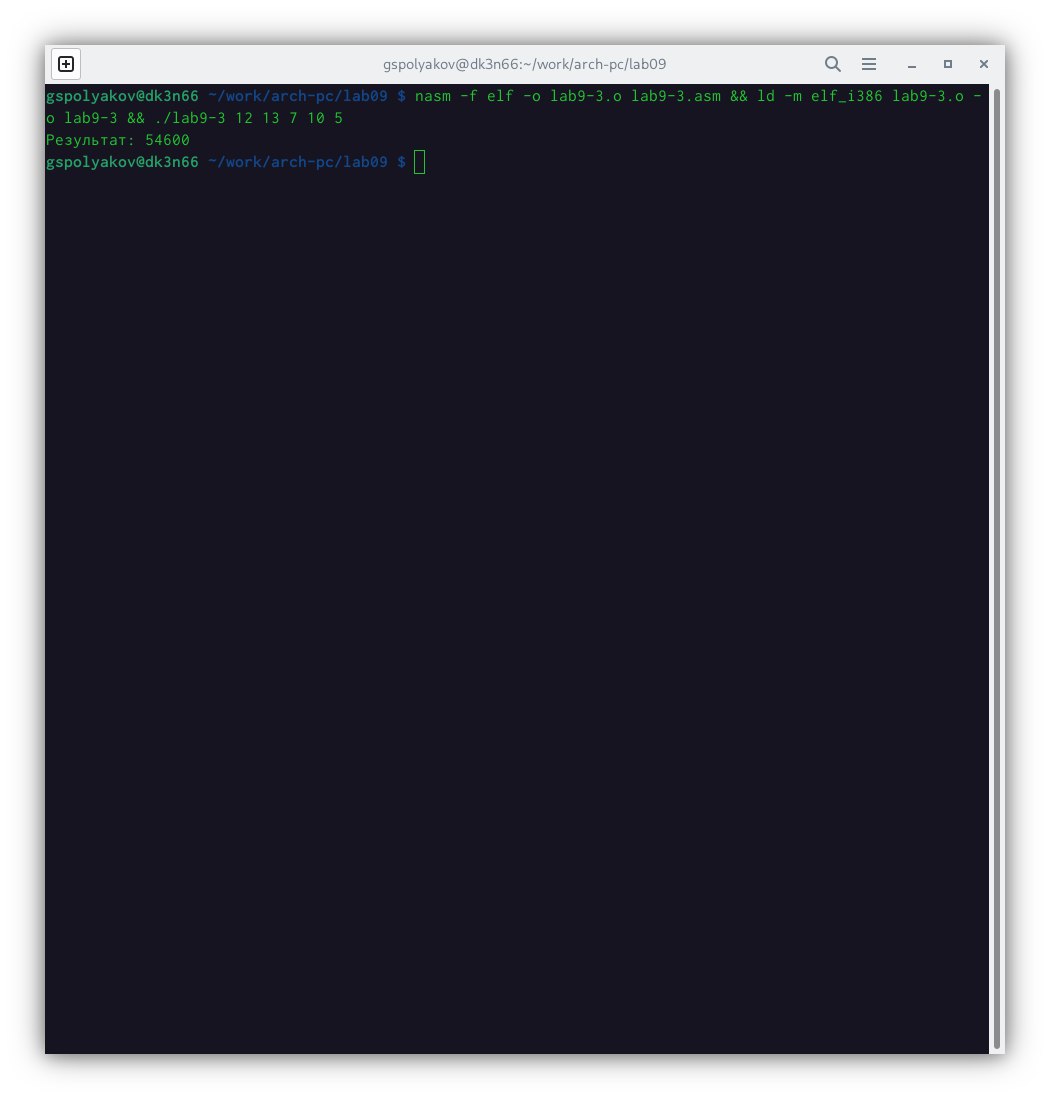


Рис. 10: Результат программы lab9-3.asm

## 4.1 Задание для самостоятельной работы

1. Создал файл lab9-4.asm
2. Ввел текст программы в файл (рис. 11)

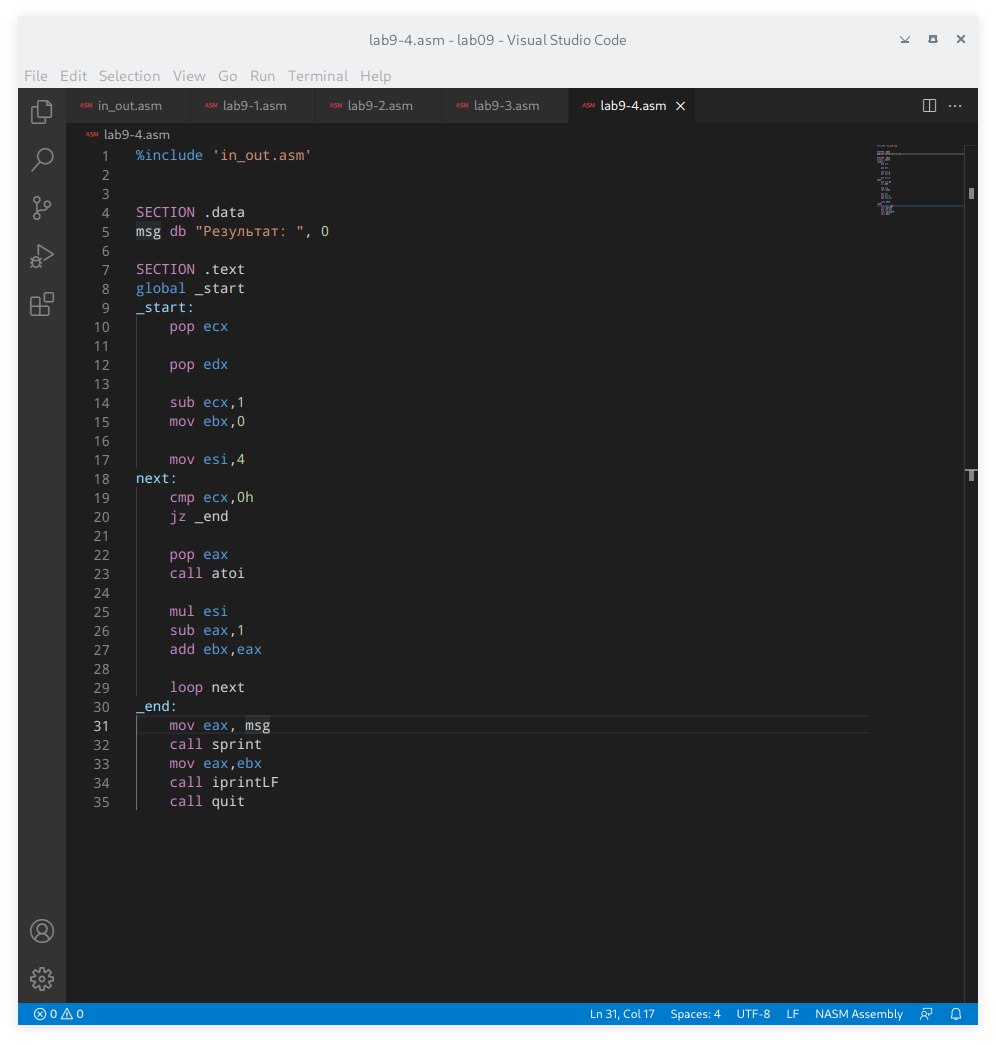


Рис. 11: Текст программы lab9-4.asm

1. Проверил работоспособность программы (рис. 12)

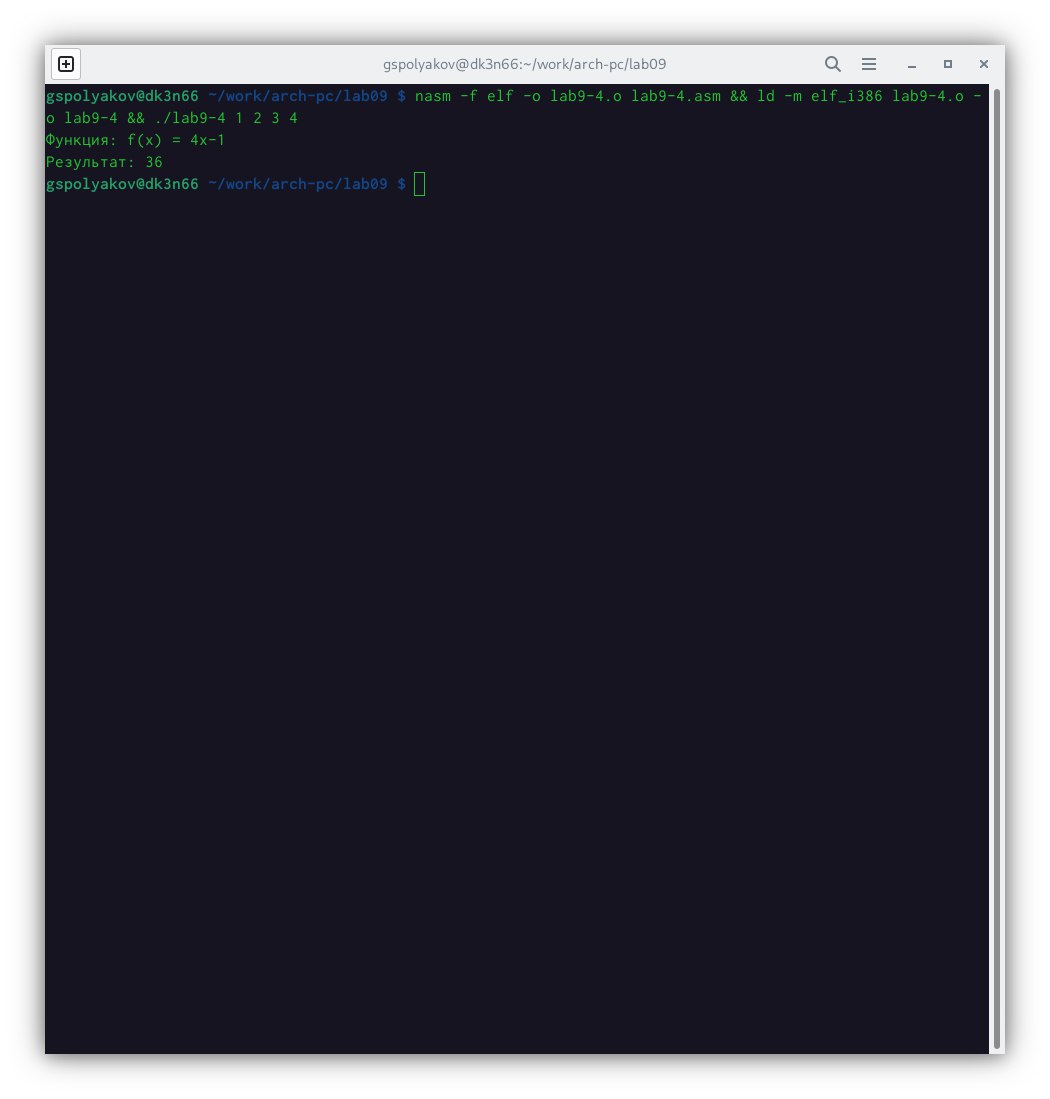


Рис. 12: Результат программы lab9-4.asm

# 5 Выводы

Делая лабораторную работу, я приобрел навыки написания программ с использованием циклов и обработкой аргументов командной строки на языке Assembler.

# Список литературы