Отчет по лабораторной работе №8

Дисциплина: архитектура компьютера

Царёв Максим Александрович

Содержание

# 1 Цель работы

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

# 2 Задание

1. Программа вывода значений регистра ecx
2. Программа выводящая на экран аргументы командной строки
3. Программа вычисления суммы аргументов командной строки
4. Выполнение заданий для самостоятельной работы

# 3 Теоретическое введение

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. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в ре- гистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указа- тель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции: • добавление элемента в вершину стека (push); • извлечение элемента из вершины стека (pop). 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:

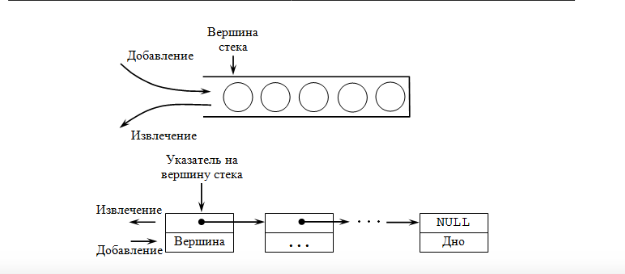


Рис. 1: Организация стека в процессоре

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.

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 ин- струкция loop. Она позволяет организовать безусловный цикл, типичная структура которого имеет следующий вид:

mov ecx, 100 ; Количество проходов

NextStep:

; тело цикла

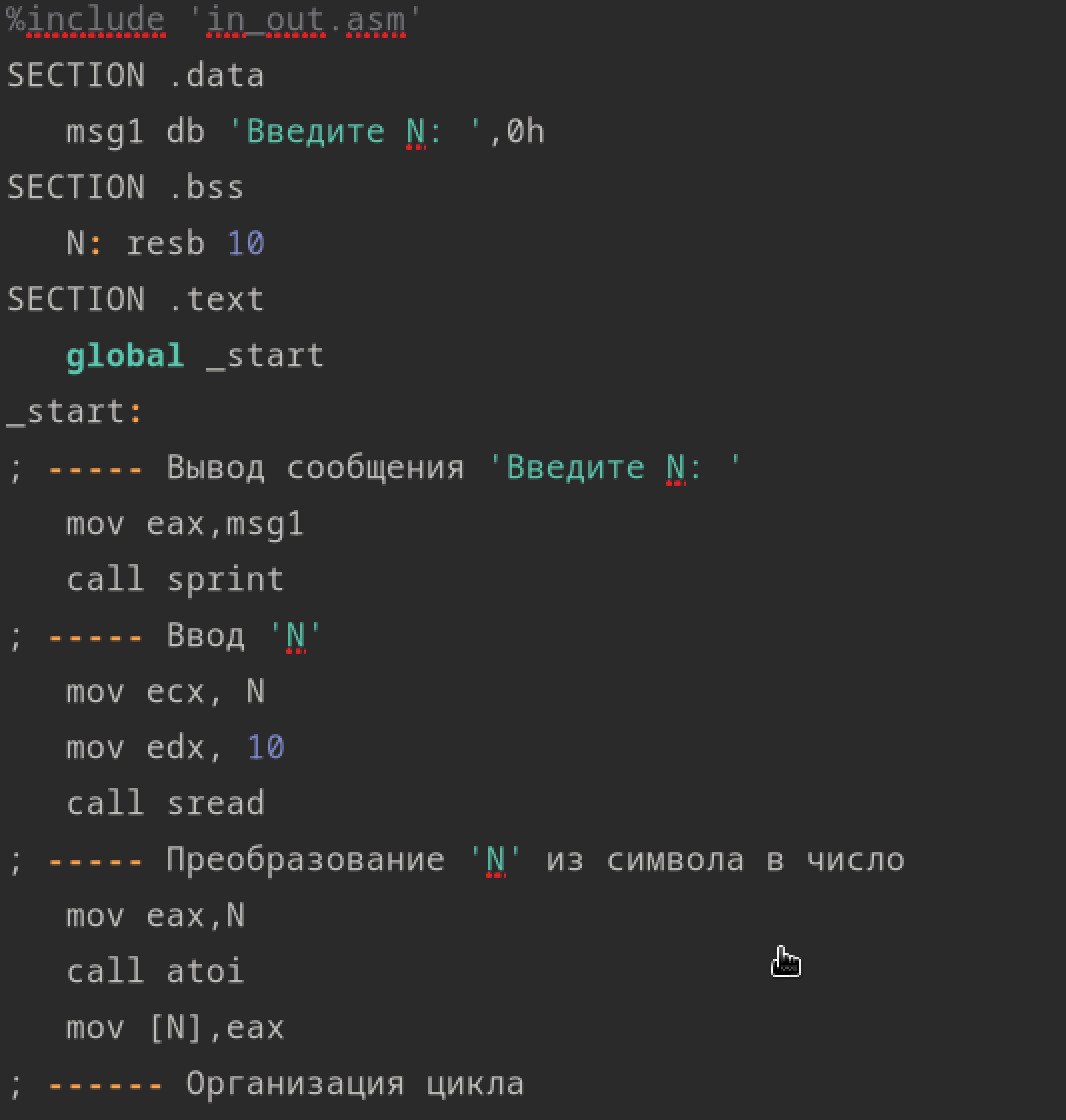
; …

loop NextStep ; Повторить ecx раз от метки NextStep

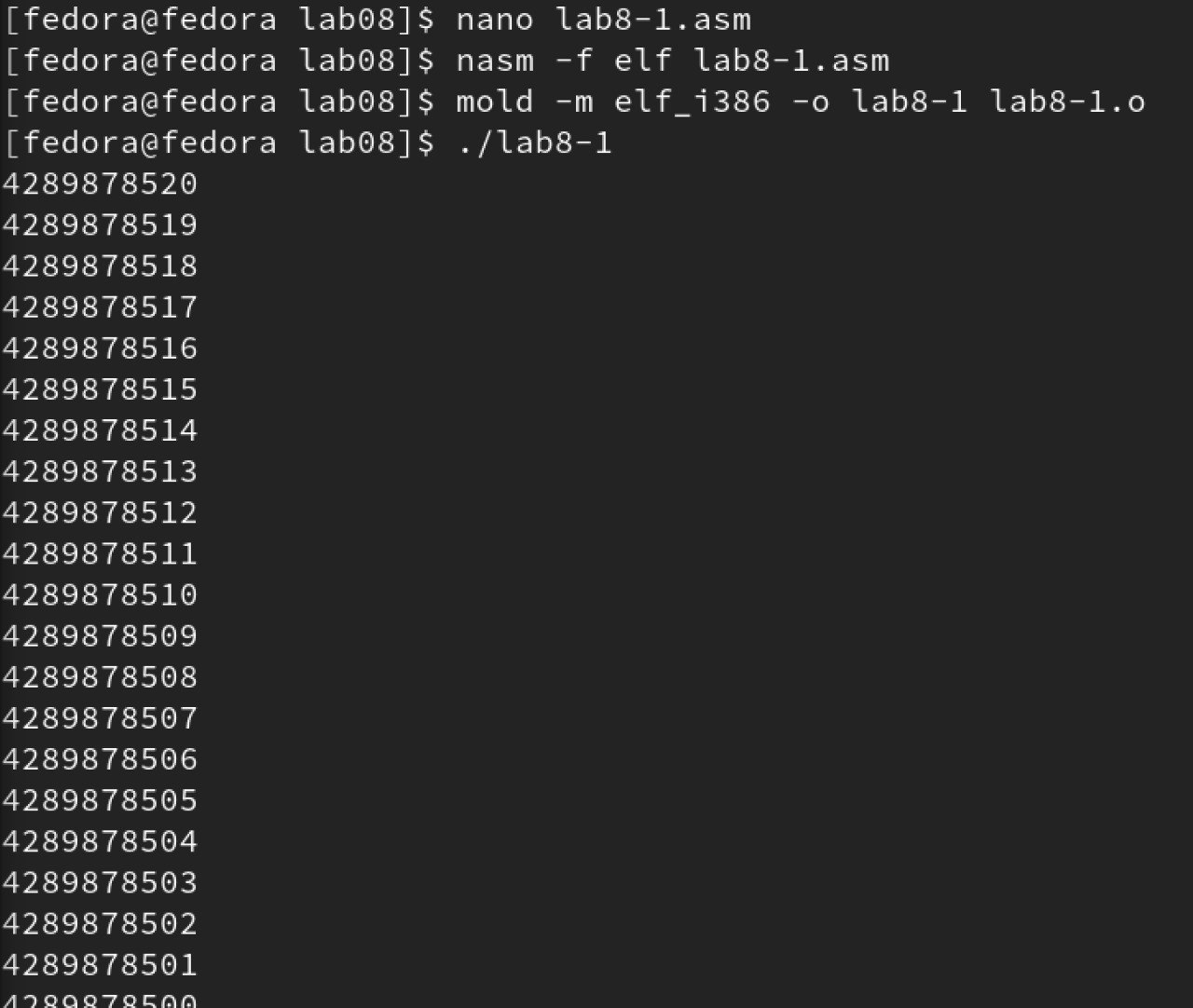
Иструкция loop выполняется в два этапа. Сначала из регистра ecx вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды loop.

# 4 Выполнение лабораторной работы

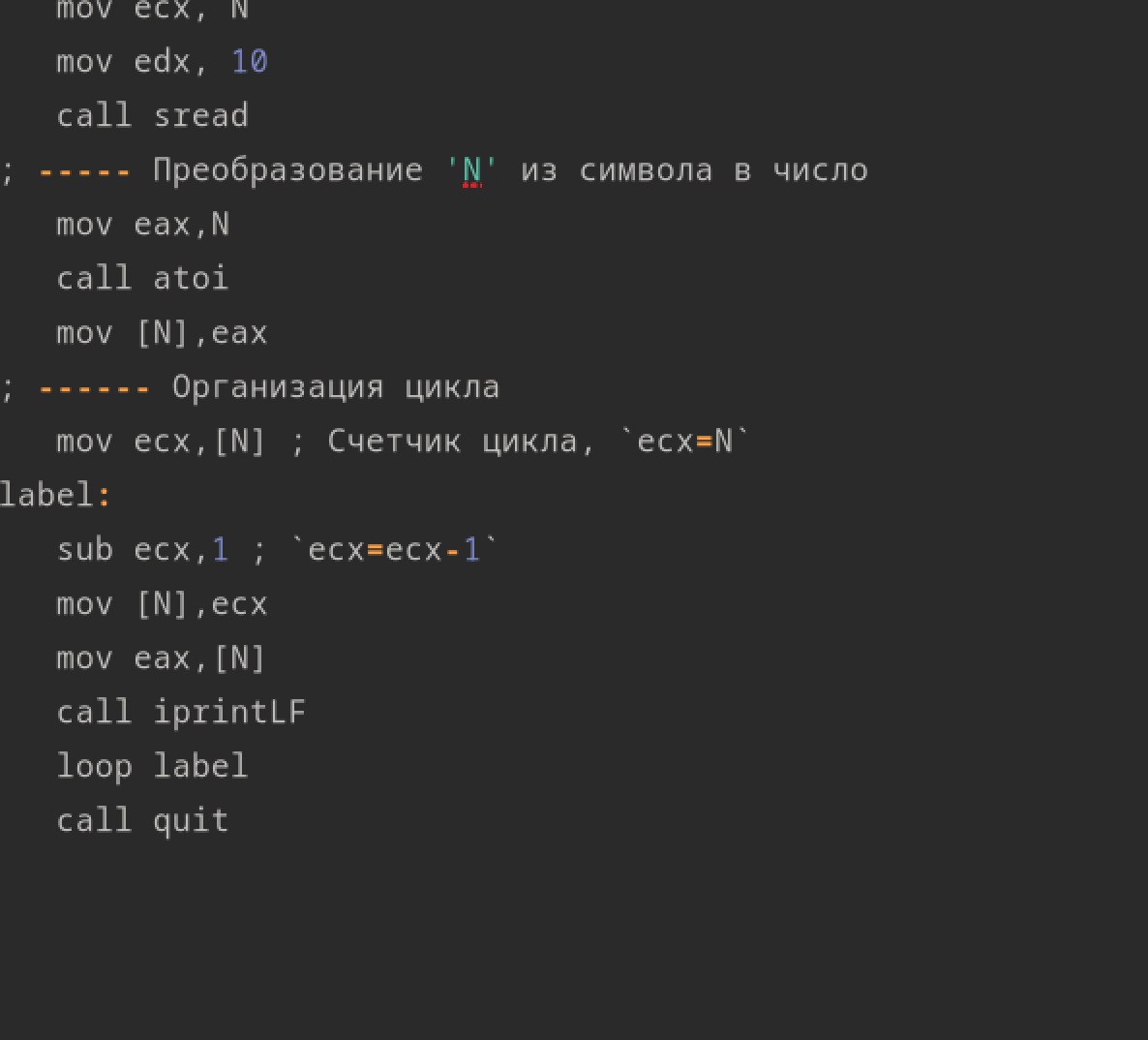
Создаю каталог lab8, перехожу в него,создаю файл lab8-1.asm. Я скопировал внешний файл в созданный ката- лог, ввёл текст программы.



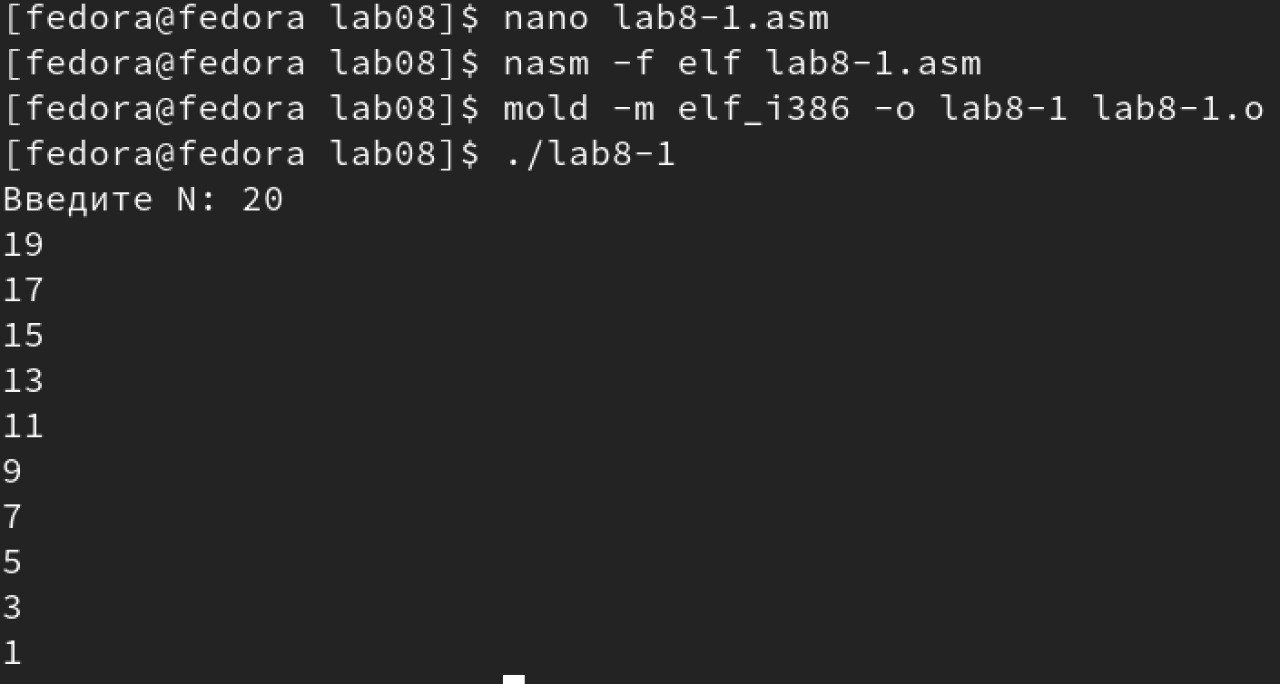
Проверил программу на корректную работу,программа работает некорректно,выводит бесконечное колличество значений



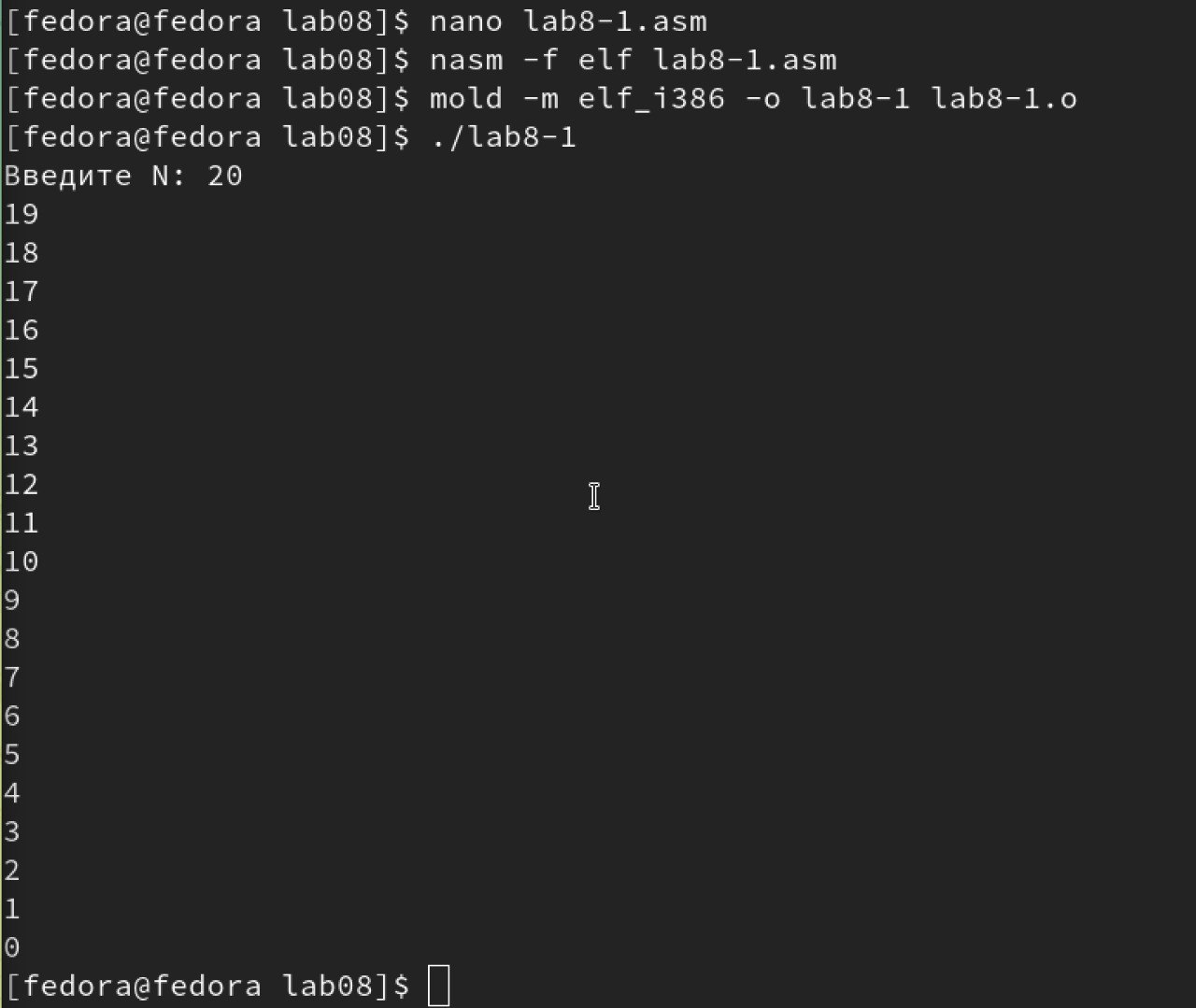
Изменяю текст программы добавив изменение значения регистра ecx в цикле



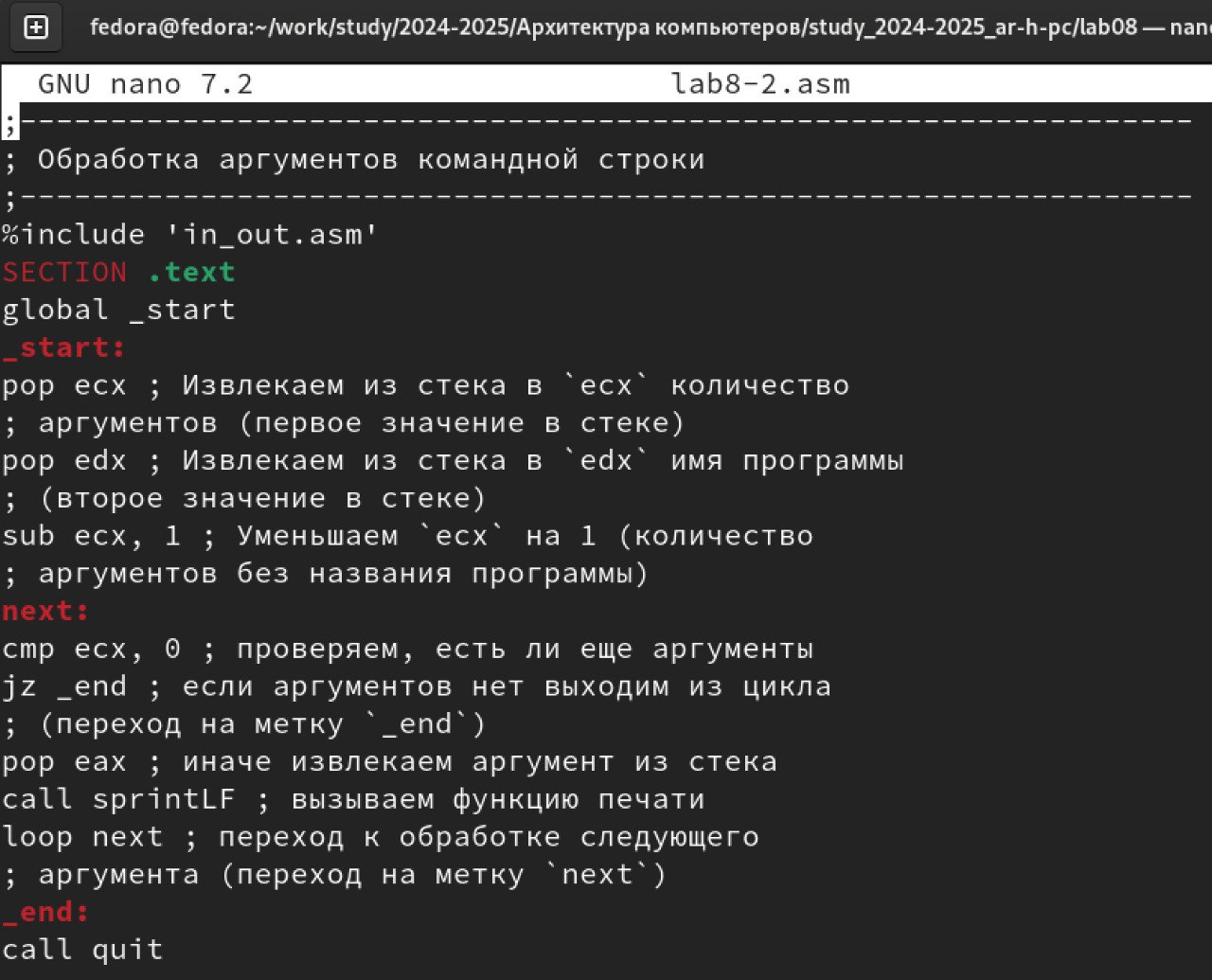
запускаю программу,теперь она работает верно



В данной программе регистр ecx принимает значения от N−1N−1 до 00, где NN — это значение, введённое пользователем с клавиатуры. ecx устанавливается равным введённому значению N, т.е. ecx = N. В каждом проходе цикла ecx уменьшается на 1 с помощью sub ecx,1. Команда loop label проверяет значение ecx. Если ecx не равно нулю, то выполняется следующий проход цикла. Когда ecx достигает нуля, выполнение цикла завершается. Нет, число проходов цикла не соответствует значению N, введённому с клавиатуры, так как команда loop повторяет цикл ровно N/2 раз, начиная с N и заканчивая 0. Вношу изменения в текст программы добавив команды push и pop для сохранения значения счетчика цикла loop

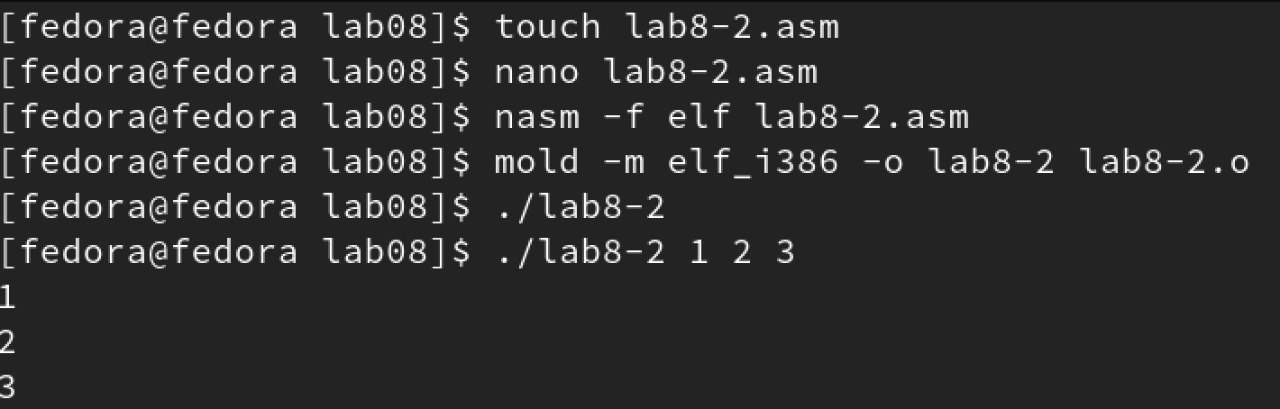


Теперь программа работает верно, делает ровно N проходов

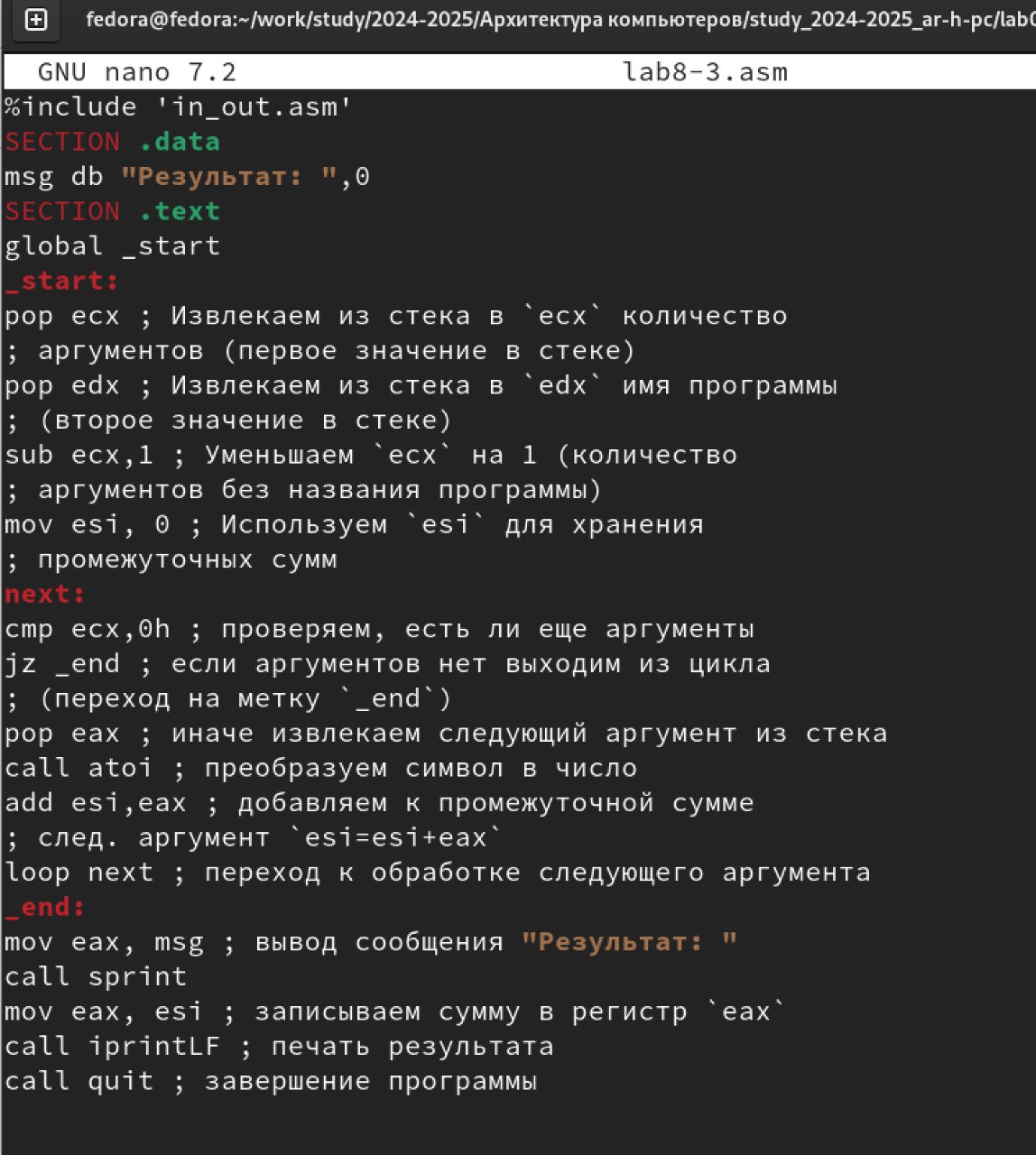


## 4.1 Обработка аргументов командной строки

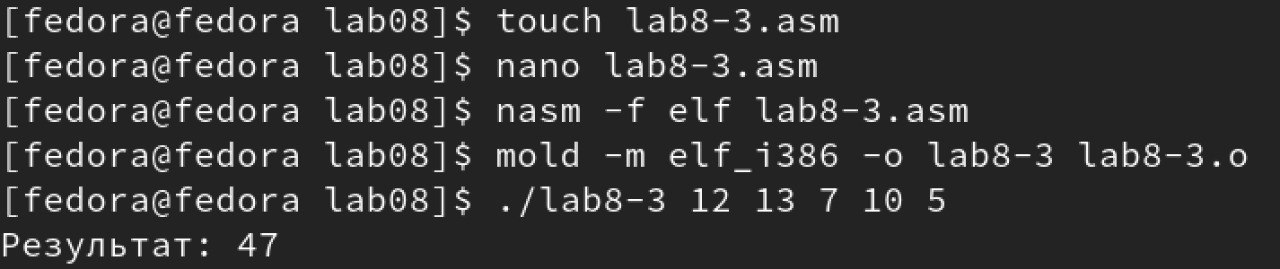
Создаю файл lab8-2.asm и ввожу в него текст программы из листинга 8.2



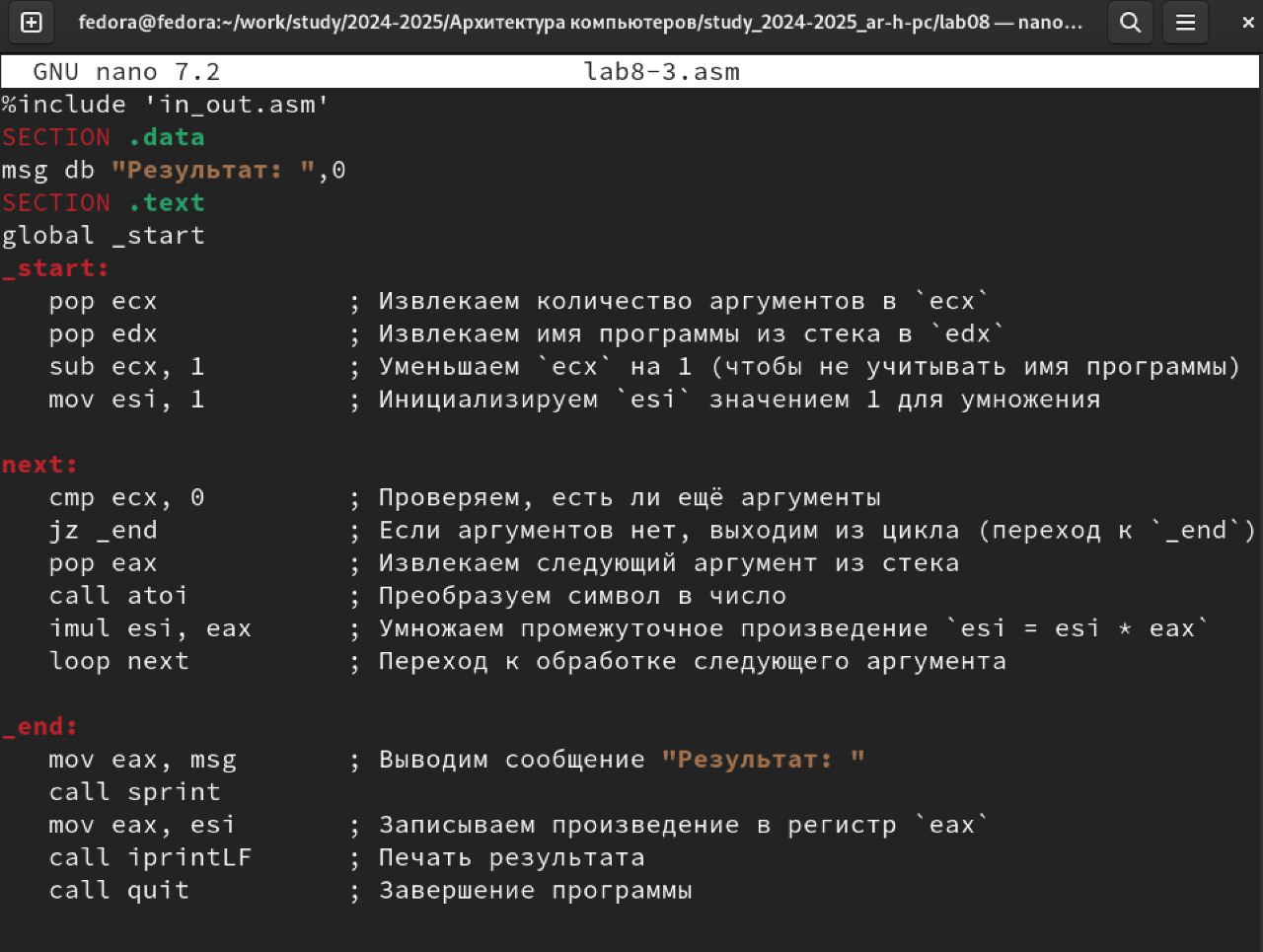
Создал исполняемый файл и запустил его, указав аргументы,программой был обработан каждый аргумент



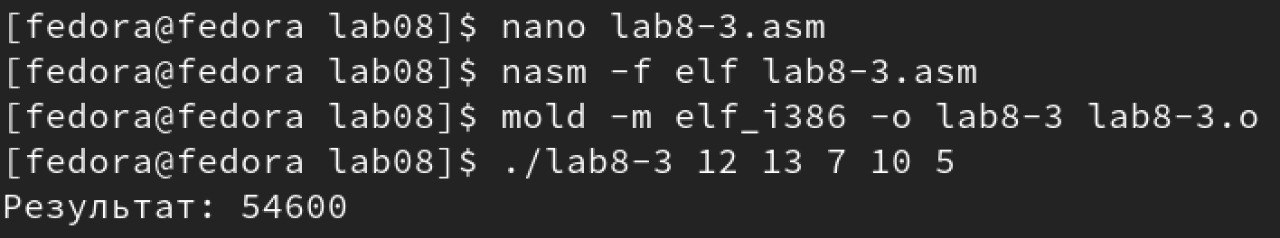
Создал файл lab8-3.asm и ввожу в него текст программы из листинга 8.3



Создал исполняемый файл и запустил его, указав аргументы,программой был обработан каждый аргумент



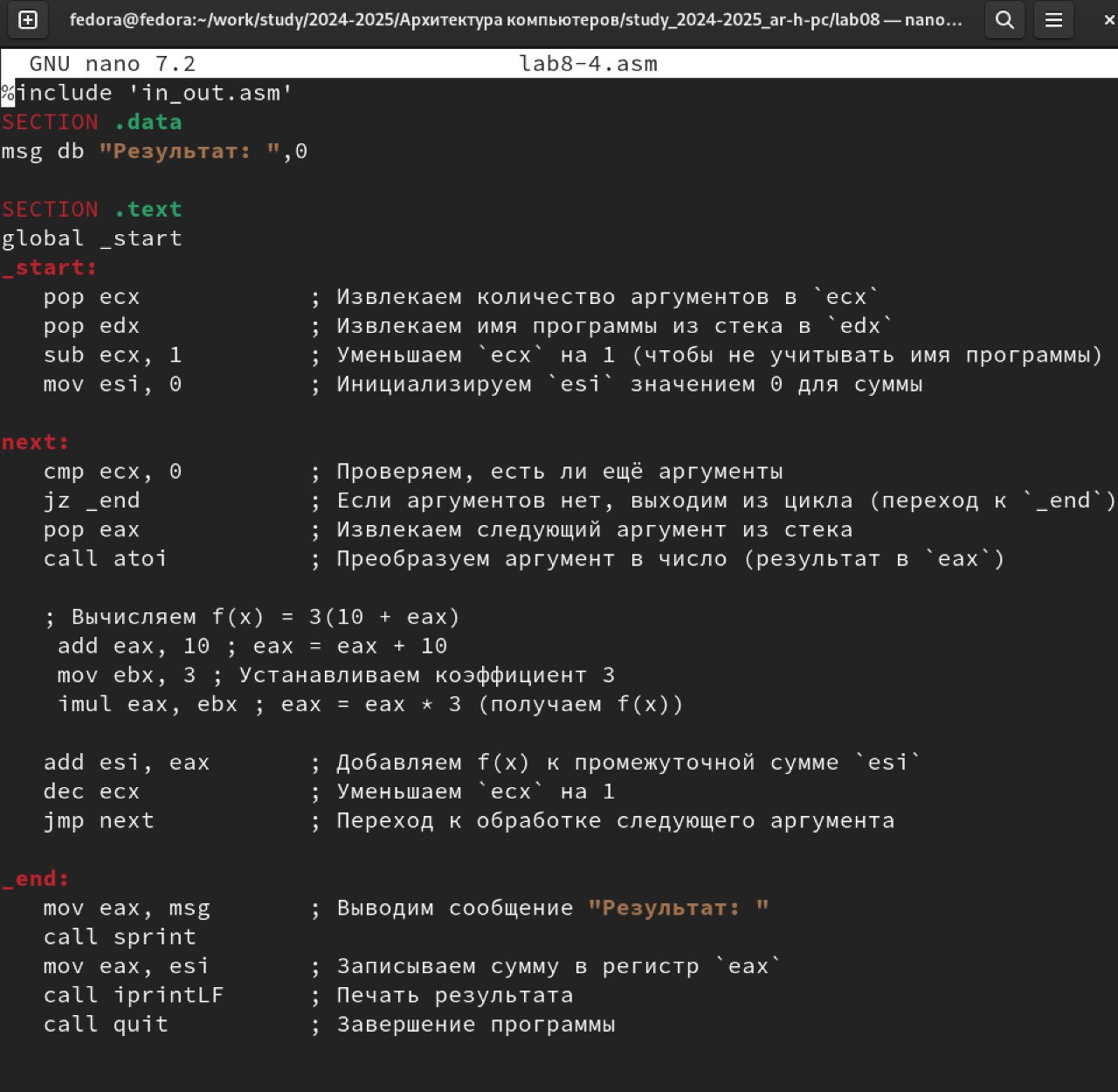
Изменил текст программы из листинга 8.3 для вычисления произведения аргументов командной строки

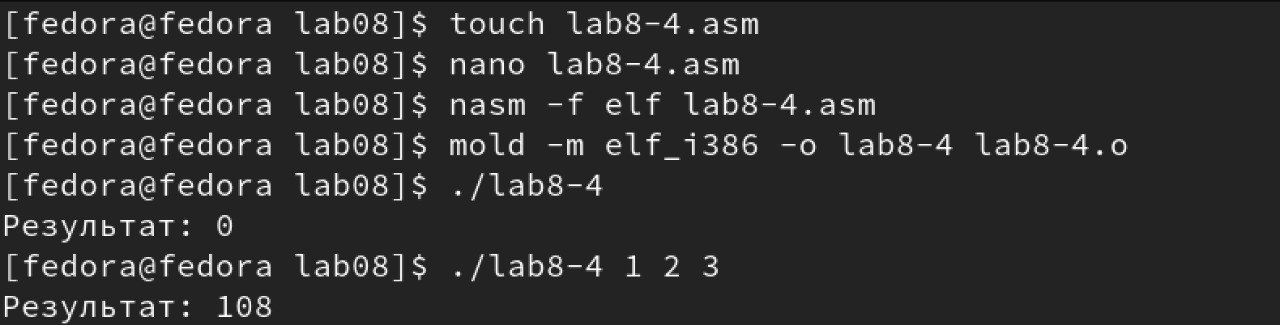


Программа работает верно

## 4.2 Выполнение заданий для самостоятельной работы

Создаю файл lab8-4.asm,пишу программу соответственно своему варианту - 20





Программа работает корректно

# 5 Выводы

Я приобрел навыки написания программ с использованием циклов и обработкой аргументов командной строки.