# Отчёт по лабораторной работе №8

## Программирование цикла. Обработка аргументов командной строки

Лань Цянын

# 1. Цель работы

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

# 2. Порядок выполнения лабораторной работы

## 2.1. Реализация циклов в NASM

Для выполнения лабораторной работы №8 создан каталог `Lab08`, внутри него создан файл `lab8-1.asm` в соответствии с инструкциями методического пособия. (рис. Рисунок 1).

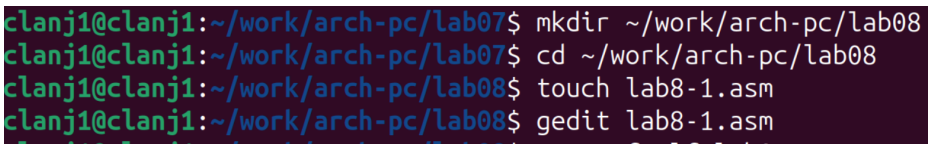



Рисунок 1: Создание каталога и файла для лабораторной работы №8

Программа `lab8-1.asm` была написана в точном соответствии с Листингом 8.1 из методического пособия.(рис. Рисунок 2)

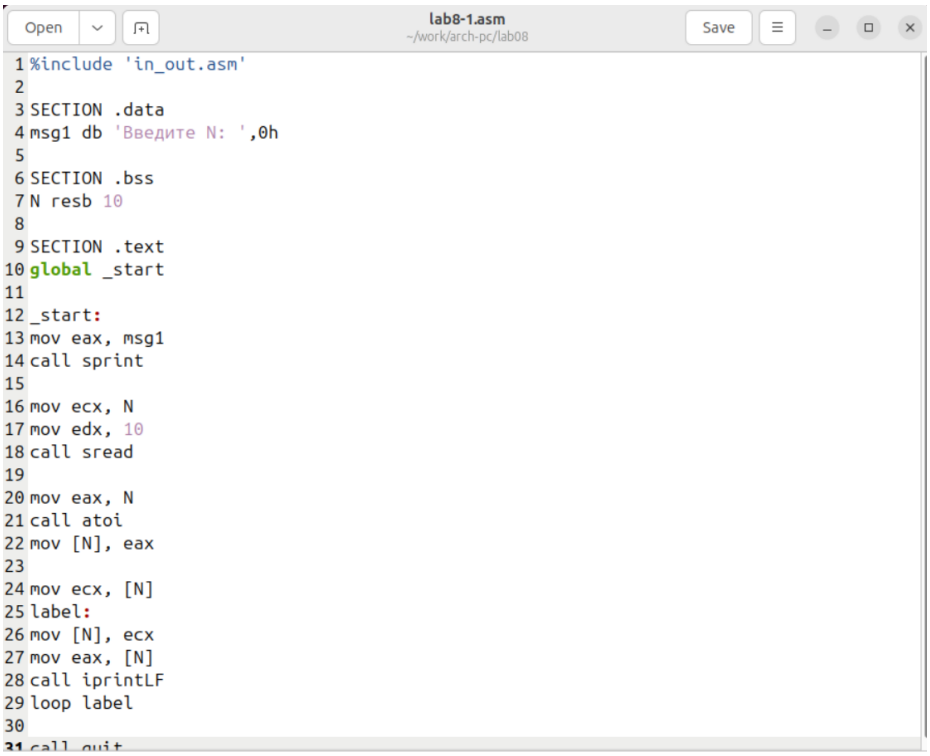



Рисунок 2: Исходный код программы lab8-1.asm (Листинг 8.1)

Программа запрашивает у пользователя число N, преобразует его из строки в целое число и затем использует регистр ecx как счётчик для цикла, выводя последовательно значения от N до 1 с помощью инструкции loop.

Программа lab8-1.asm была откомпилирована и запущена Для её работы в каталог lab08 был предварительно скопирован необходимый файл in_out.asm.(рис. Рисунок 3)


Рисунок 3: Компиляция, сборка и запуск программы lab8-1

Программа успешно выполняется, используя инструкцию `loop` для организации цикла, который выводит значения от введённого числа 5 до 1.

Исходный код программы `lab8-1.asm` был модифицирован в соответствии с заданием: в тело цикла `label` добавлена инструкция `sub ecx,1`, которая явно уменьшает значение регистра-счётчика на каждой итерации.(рис. Рисунок 4)

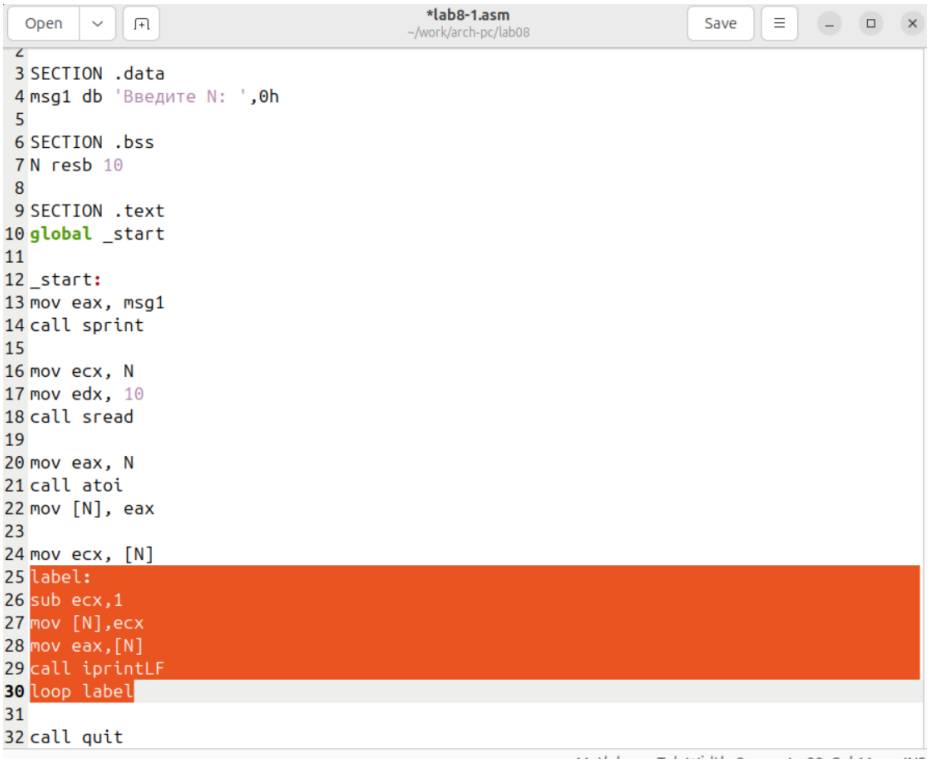



Рисунок 4: Модифицированный код программы lab8-1.asm

Программа, модифицированная добавлением `sub ecx,1` в тело цикла, при выполнении вошла в бесконечный цикл.(рис. Рисунок 5)

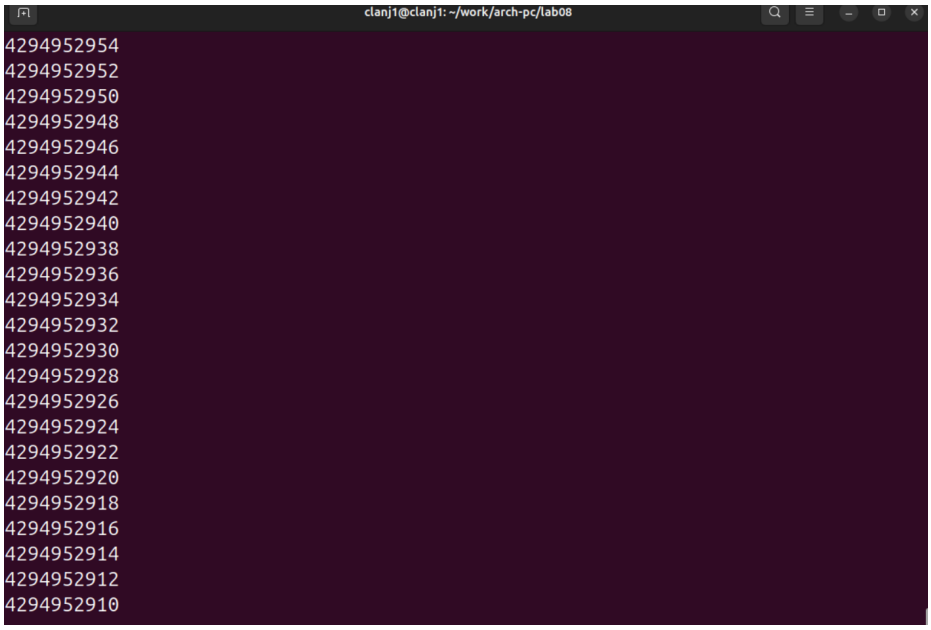



Рисунок 5: Бесконечный цикл в модифицированной программе

Это происходит потому, что инструкция `loop` также уменьшает `ecx` на 1, и их совместное действие может привести к тому, что `ecx` никогда не достигнет нуля (например, перескочив с 1 на -1), в результате чего условие завершения цикла не выполняется.

**Ответы на вопросы**

1. **Какие значения принимает регистр ecx в цикле?**

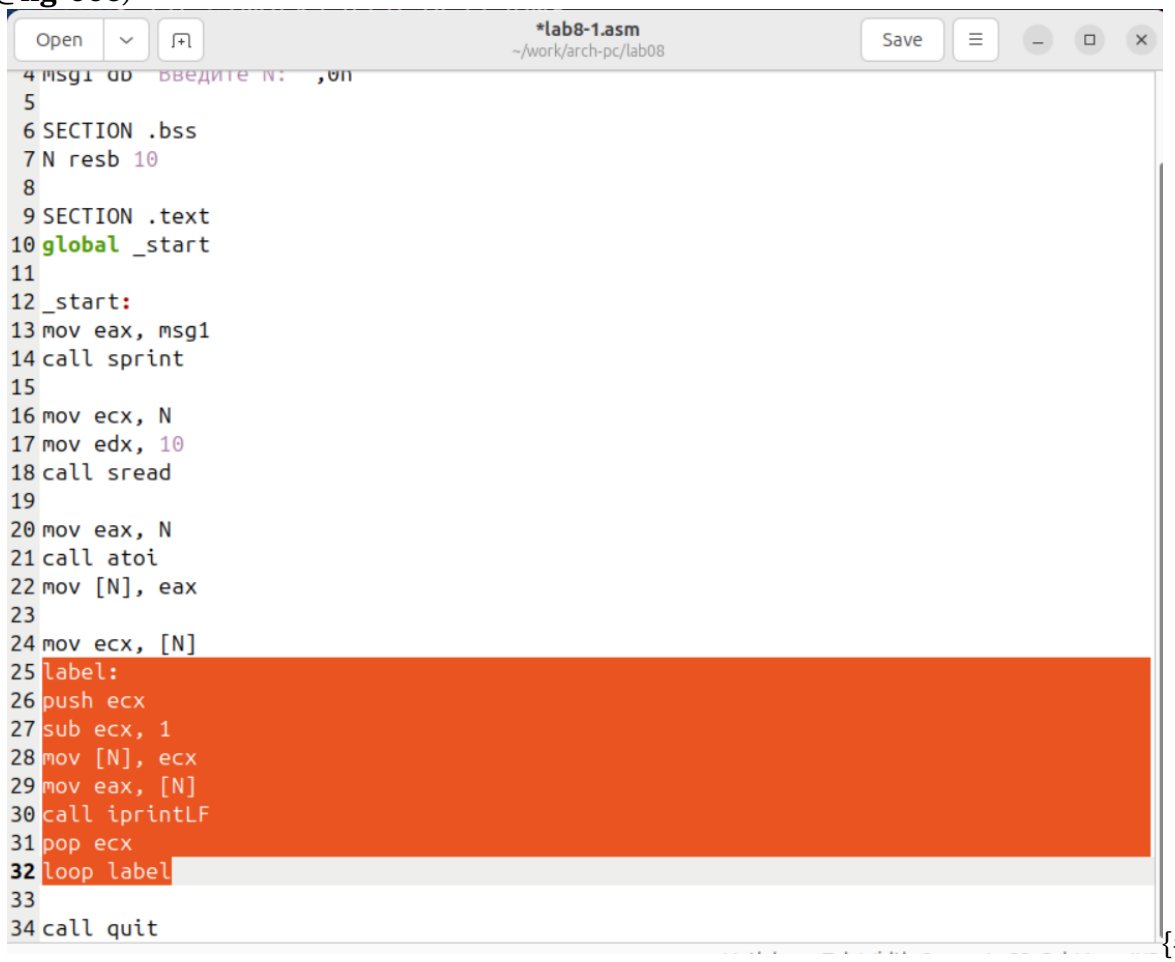
   Значение регистра ecx уменьшается дважды за одну итерацию цикла: один раз инструкцией sub, и второй раз — инструкцией loop. В результате выводимая последовательность чисел уменьшается на 2 (например: N−1, N−3, N−5 и т.д.).При нечётном значении N регистр ecx принимает отрицательные значения, что приводит к зацикливанию программы.

2. **Соответствует ли число проходов цикла значению N ?**

   Нет. Поскольку регистр ecx уменьшается два раза за одну итерацию, фактическое число проходов цикла примерно равно N/2 (при чётном N). При нечётном значении N программа не завершается и переходит в

бесконечный цикл.

Программа `lab8-1.asm` была модифицирована с использованием стека для сохранения и восстановления значения счётчика цикла `ecx`, что позволяет корректно работать инструкции `loop` при изменении `ecx` в теле цикла(рис. **?@fig-006**)



{#fig-006 width=80%

Программа, модифицированная с использованием стека для сохранения значения счётчика `ecx`, была успешно откомпилирована и запущена(рис. Рисунок 6)

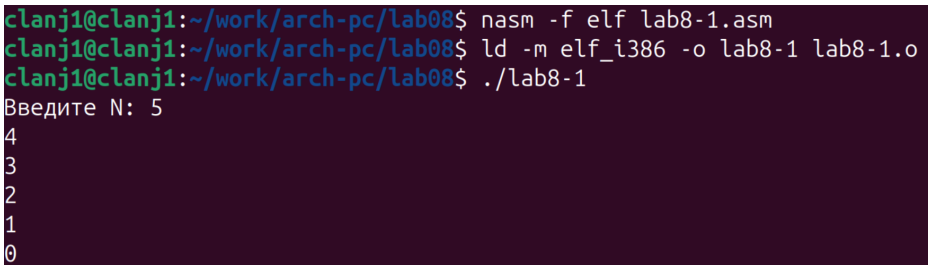



Рисунок 6: Выполнение программы с использованием стека

Использование `push ecx` и `pop ecx` обеспечивает корректную работу инструкции `loop`, гарантируя выполнение цикла ровно `N` раз, что подтверждается выводом последовательных чисел (например, при `N=5` выводится 4, 3, 2, 1, 0)

**Ответ на вопрос задания:**

**Соответствует ли число проходов цикла значению N ?**

Да, полностью соответствует. Использование стека для сохранения состояния регистра `ecx` устраняет проблему двойного уменьшения счётчика, и цикл выполняется ровно `N` раз, как и задумано изначально.

## 2.2. Обработка аргументов командной строки

Программа `lab8-2.asm` для обработки аргументов командной строки была написана в точном соответствии с Листингом 8.2(рис. Рисунок 7)


Рисунок 7: Исходный код программы lab8-2.asm

Программа `lab8-2.asm` была успешно откомпилирована, собрана и протестирована с передачей аргументов командной строки(рис. Рисунок 8)

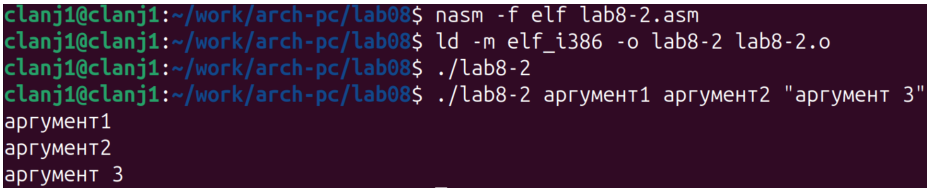



Рисунок 8: Выполнение программы lab8-2 с аргументами командной строки

Программа корректно обработала и вывела все три переданных аргумента, что подтверждает правильность реализации извлечения аргументов из стека и работы циклической обработки

**Ответ на вопрос задания:**

**Сколько аргументов было обработано программой?**

Программа обработала **3 аргумента**, что соответствует количеству аргументов, переданных в командной строке (после имени программы).

Программа `lab8-3.asm`, реализующая вычисление суммы числовых аргументов командной строки, была написана на основе Листинга 8.3(рис. Рисунок 9)


Рисунок 9: Исходный код программы lab8-3.asm

Программа `lab8-3.asm` была откомпилирована и запущена с передачей числовых аргументов для проверки работы алгоритма суммирования(рис. Рисунок 10)

```
clanj1@clanj1:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
clanj1@clanj1:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
clanj1@clanj1:~/work/arch-pc/lab08$ ./lab8-3 1 2 3 4
Результат: 10
```

Рисунок 10: Компиляция, сборка и запуск программы lab8-3

Программа успешно обработала аргументы `1 2 3 4`.Ожидаемый результат суммирования — `10`. Это подтверждает корректную работу всех этапов: извлечения количества аргументов, их преобразования из строк в числа (`atoi`), накопления суммы в регистре `esi` и вывода итога

Программа `lab8-3.asm` была модифицирована для вычисления **произведения** числовых аргументов командной строки вместо их суммы(рис. Рисунок 11)

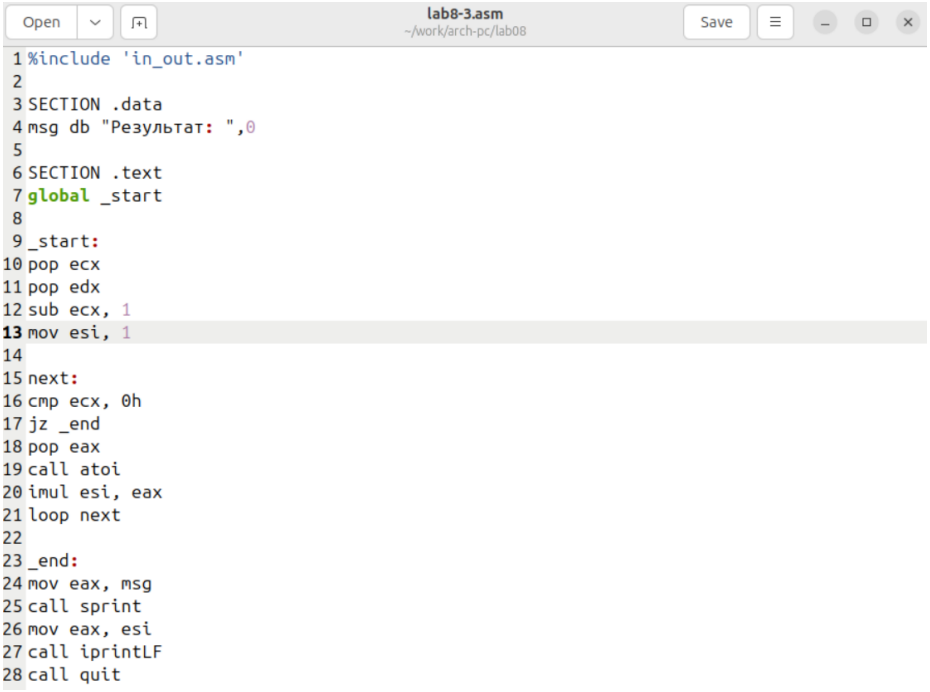



Рисунок 11: Модифицированный код программы lab8-3.asm для вычисления произведения

Регистр `esi` (в коде `est`) инициализируется значением **`1`** вместо `0`, так как произведение любого числа на 1 равно самому числу.Инструкция сложения `add est, eax` заменена на инструкцию умножения **`imul est, eax`**

Модифицированная программа `lab8-3.asm` была успешно откомпилирована и запущена для проверки вычисления произведения аргументов(рис. Рисунок 12)

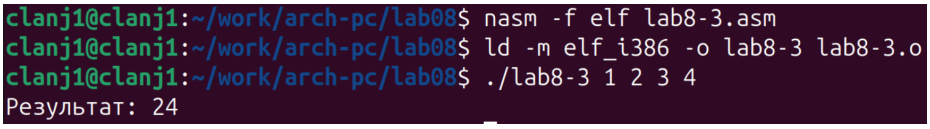



Рисунок 12: Выполнение программы для вычисления произведения

Программа корректно вычислила и вывела результат 24, что соответствует произведению чисел 1, 2, 3 и 4.

# 3. Задание для самостоятельной работы

Программа `lab8-4.asm` была написана для вычисления суммы значений функции f(x) = 10x - 4 (вариант 9) для набора чисел x , переданных в качестве аргументов командной строки(рис. Рисунок 1)

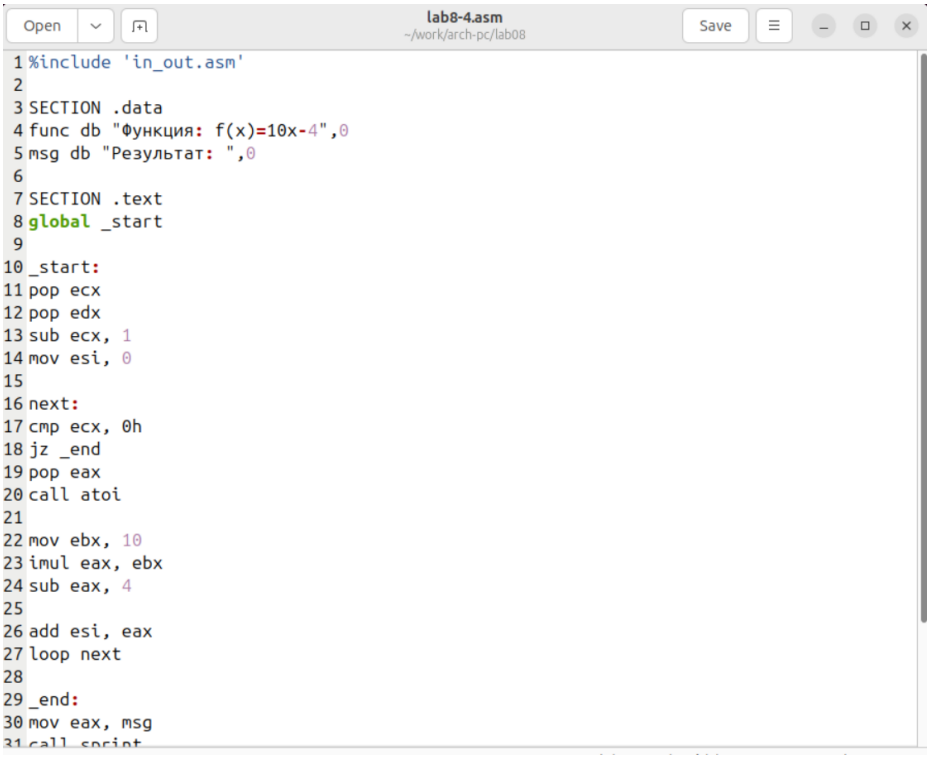



Рисунок 1: Исходный код программы lab8-4.asm

Программа `lab8-4.asm` была успешно откомпилирована и протестирована

на наборе значений 1  2  3  4(рис. Рисунок 2)

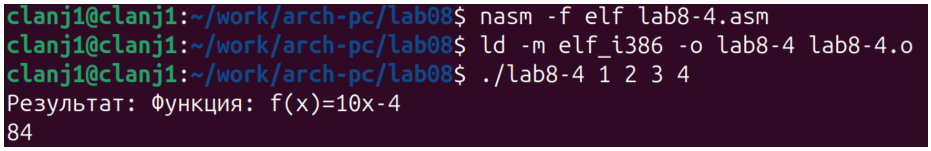



Рисунок 2: Тестирование программы lab8-4.asm

Программа вывела ожидаемый результат: 84 Это подтверждает, что она корректно вычисляет значение функции f(x) = 10x - 4 для каждого переданного аргумента x , суммирует эти значения и выводит итог

# 4 Выводы

В ходе лабораторной работы № 8 были изучены инструкции организации циклов в NASM и способы обработки аргументов командной строки. Рассмотрено влияние изменения регистра ecx на корректность работы инструкции loop и показана необходимость использования стека для сохранения счётчика цикла. Получены практические навыки работы с циклами, стеком и аргументами командной строки, что соответствует цели лабораторной работы.