Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки

Гурылев Артем Андреевич

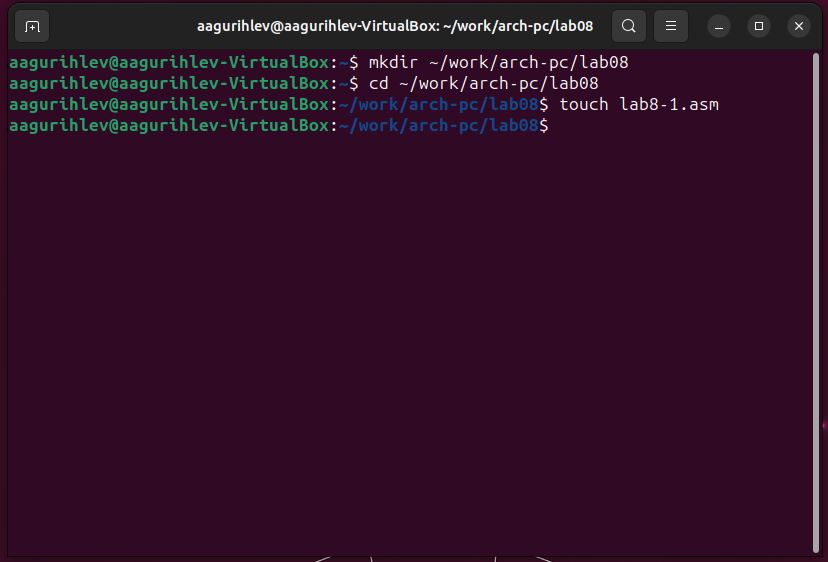
Содержание

# 1 Цель работы

Целью работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

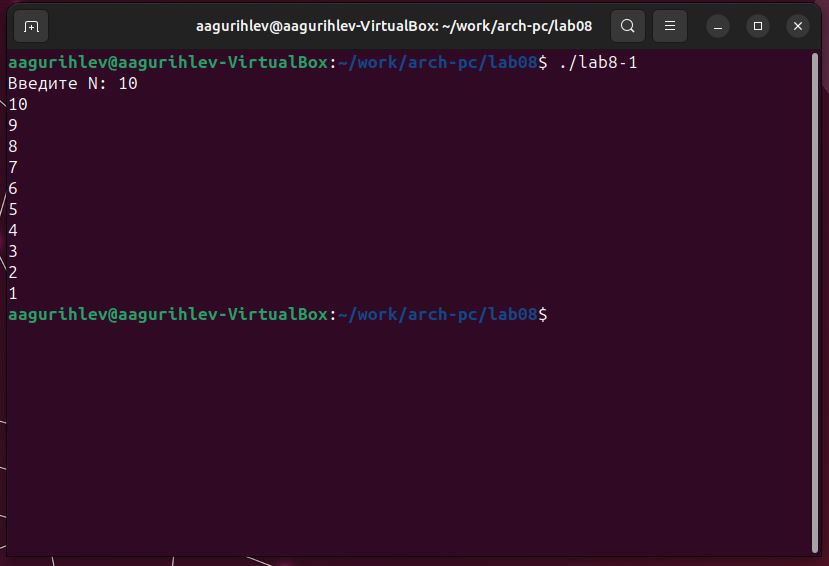
# 2 Выполнение лабораторной работы

Создадим каталог для работы с программами и создадим файл lab8-1.asm: (рис. [**1?**])



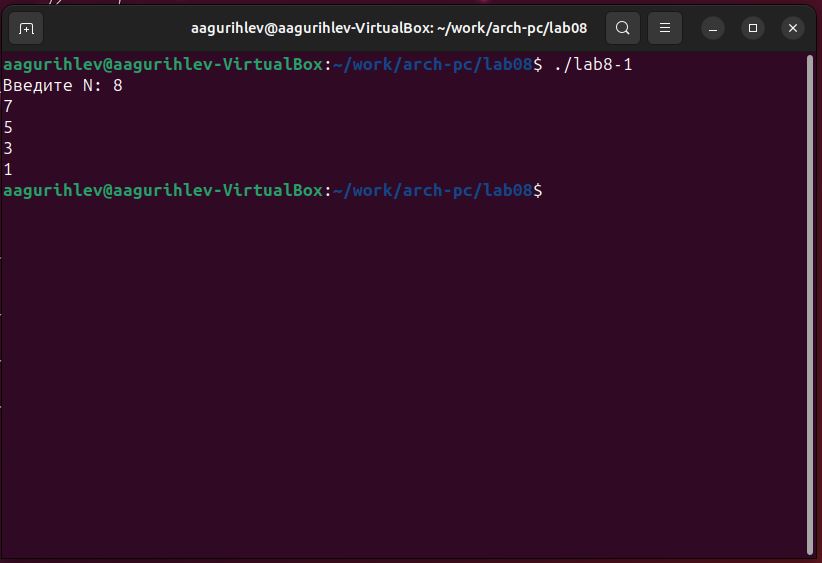
Создание файла lab8-1.asm

Введём код из листинга в файл lab8-1.asm, преобразуем его и проверим работоспособность: (рис. [**2?**])



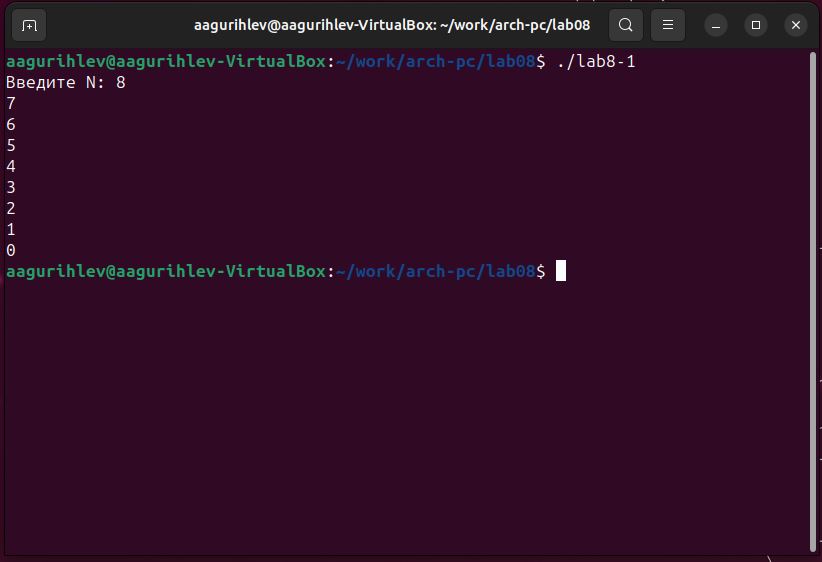
Пример работы файла lab8-1.asm

Изменим текст программы, добавив туда строчку с вычитанием из регистра ecx единицы, и проверим её работу(рис. [**3?**]):



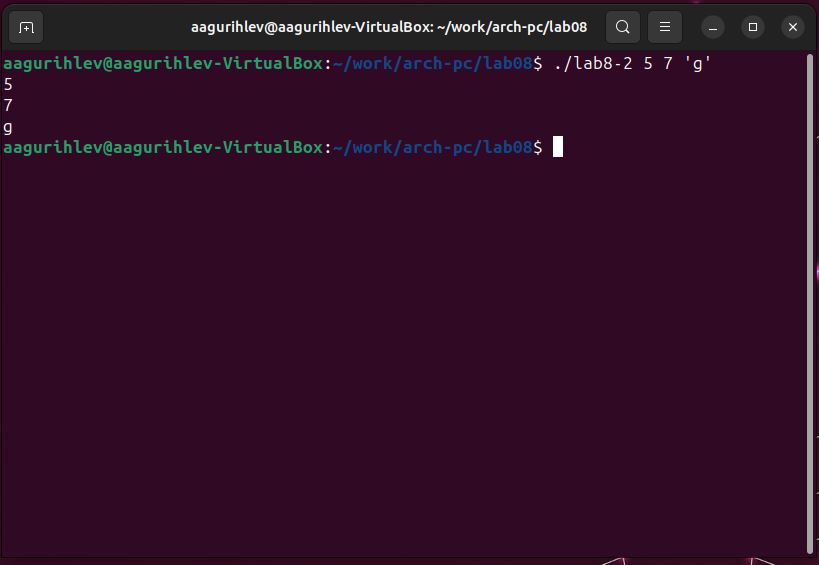
Пример работы изменённой программы

Так как инструкция loop всегда сама по себе вычитает единицу из регистра ecx, то в программе этого не нужно делать, иначе проходов будет примерно в два раза меньше, так как при каждом из них из ecx будет вычитаться 2, а не 1. Теперь изменим код программы, чтобы в ней использовался стек, и проверим её работу(рис. [**4?**]):



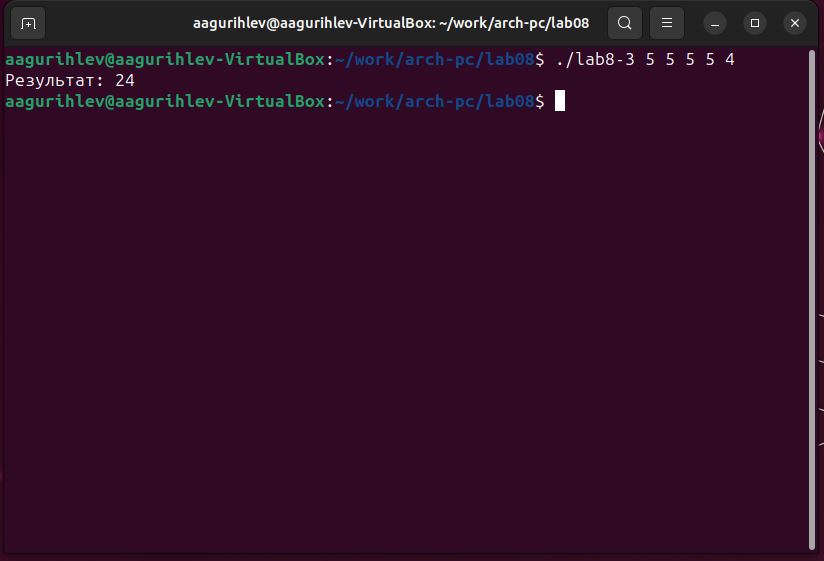
Пример работы программы с использованием стека

Число проходов соответствует введенному числу. Создадим файл lab8-2.asm, введём из него код из листинга и создадим исполнительный файл, после чего запустим его из командной строки с аргументами(рис. [**5?**]):



Пример работы программы lab8-2

Программа успешно обработала все аргументы, записанные в командной строке. Создадим файл lab8-3.asm, введём из него код из листинга и создадим исполнительный файл, после чего запустим его из командной строки с аргументами(рис. [**6?**]):



Пример работы программы lab8-3

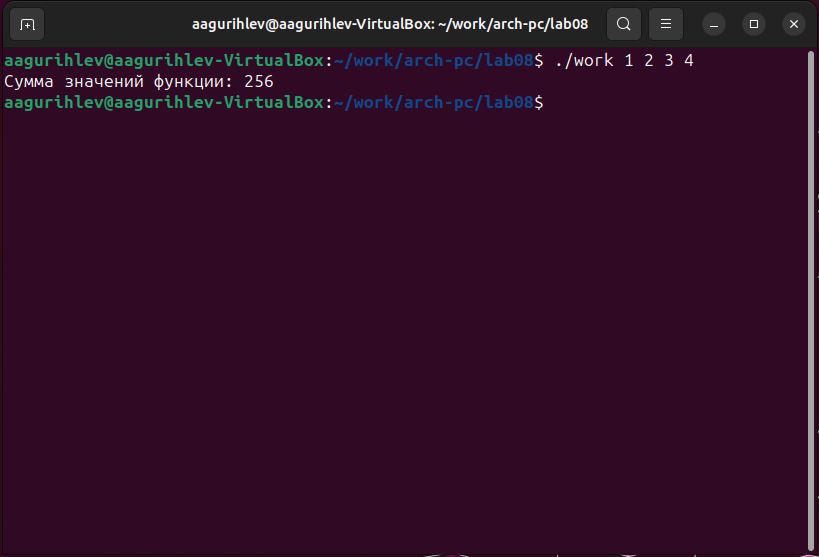
Программа выводит сумму всех аргументов.

# 3 Выполнение самостоятельной работы

Мой вариант - 16, следовательно функция, с которой я буду работать - *30x - 11*. Листинг кода программы:

%include 'in\_out.asm' SECTION .data msg: DB 'Сумма значений функции: ', 0 ; SECTION .text global \_start \_start: pop ecx ; pop edx ; dec ecx ; mov esi, 0 ; esi - сумма всех значений функции func: cmp ecx, 0 ; jz \_end ; pop eax ; call atoi ; mov ebx, 30 ; mul ebx ; sub eax, 11 ; add esi, eax ; loop func ; \_end: mov eax, msg ; call sprint ; mov eax, esi ; call iprintLF ; call quit ;

Пример работы программы(рис. [**7?**]):



Пример работы программы

# 4 Выводы

В данной лабораторной работе я научился использовать стек, циклы и обработку аргументов командной строки в программах ассемблера NASM, которые нужны не только для правильной работы программ, но и для их оптимизации.