

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки

Гурылев Артем Андреевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение самостоятельной работы	12
4	Выводы	14

Список иллюстраций

2.1	Создание файла lab8-1.asm	6
2.2	Пример работы файла lab8-1.asm	7
2.3	Пример работы изменённой программы	8
2.4	Пример работы программы с использованием стека	9
2.5	Пример работы программы lab8-2	10
2.6	Пример работы программы lab8-3	11
3.1	Пример работы программы	13

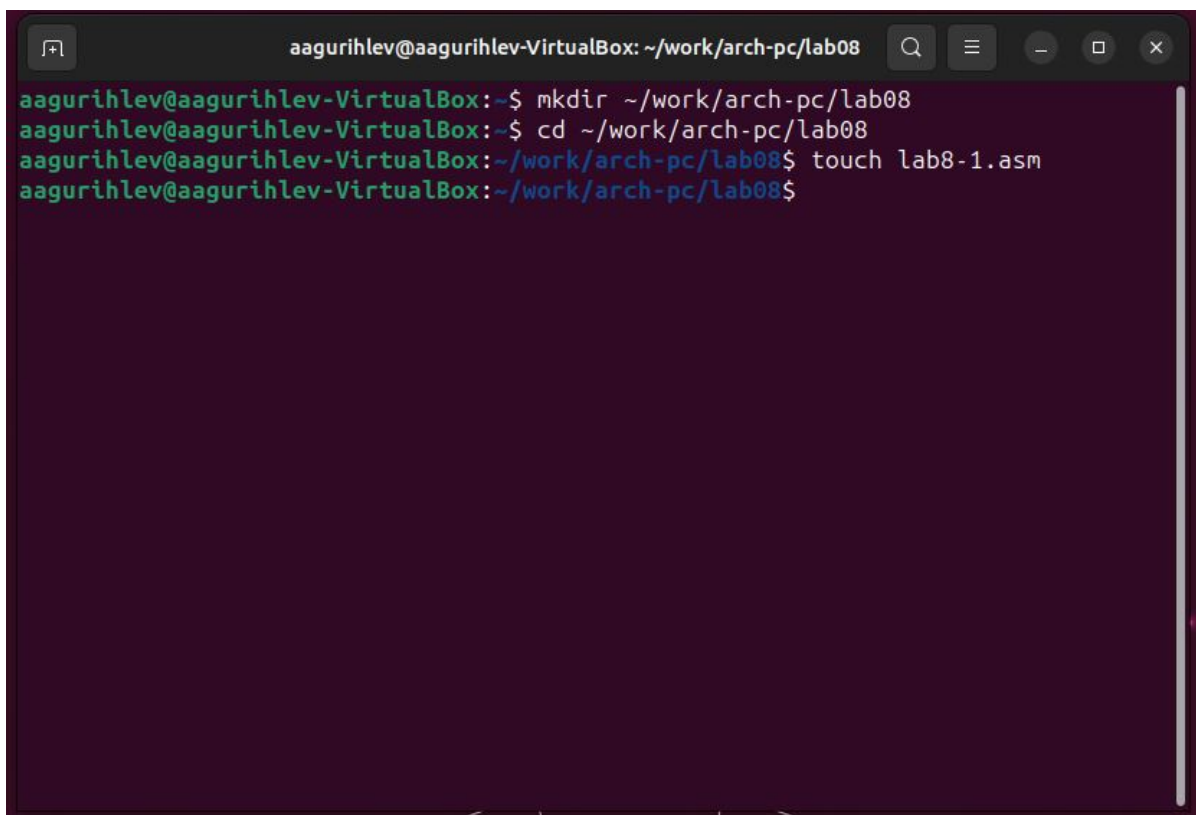
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

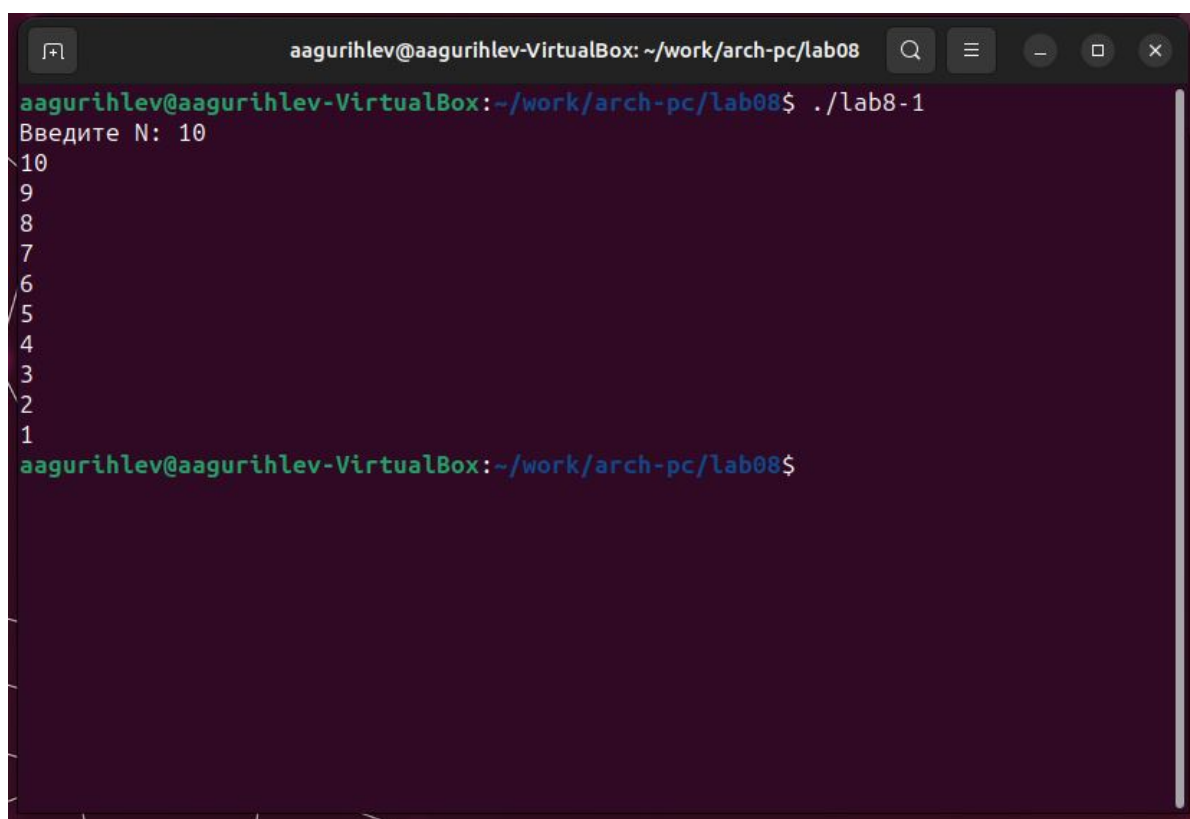
Создадим каталог для работы с программами и создадим файл lab8-1.asm: (рис. [1?])

A screenshot of a terminal window with a dark background. The window title bar shows the user 'aagurihlev' on a host 'aagurihlev-VirtualBox' at the directory '~/work/arch-pc/lab08'. The terminal contains four lines of text: the first line shows the command 'mkdir ~/work/arch-pc/lab08' being executed; the second line shows 'cd ~/work/arch-pc/lab08'; the third line shows 'touch lab8-1.asm'; and the fourth line shows the prompt after the file has been created.

```
aagurihlev@aagurihlev-VirtualBox: ~/work/arch-pc/lab08
aagurihlev@aagurihlev-VirtualBox:~$ mkdir ~/work/arch-pc/lab08
aagurihlev@aagurihlev-VirtualBox:~$ cd ~/work/arch-pc/lab08
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$ touch lab8-1.asm
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.1: Создание файла lab8-1.asm

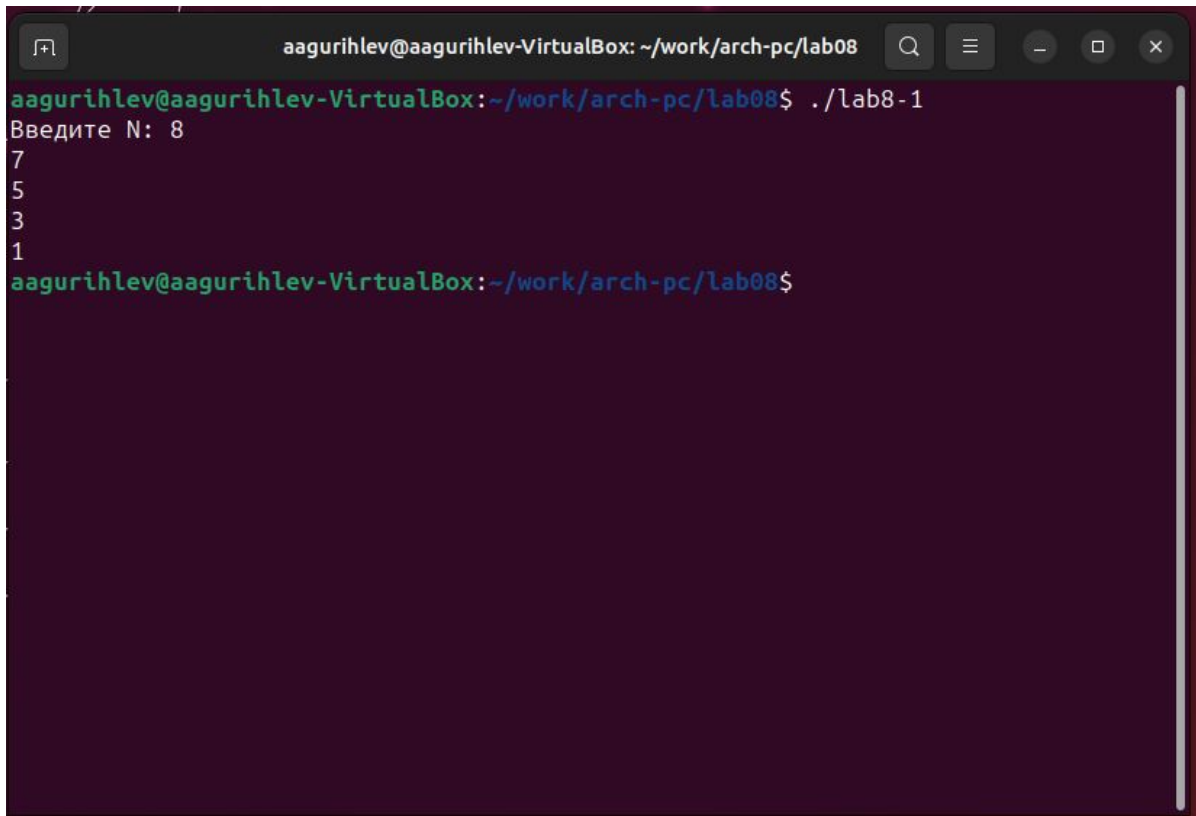
Введём код из листинга в файл lab8-1.asm, преобразуем его и проверим работоспособность: (рис. [2?])



```
aagurihlev@aagurihlev-VirtualBox: ~/work/arch-pc/lab08
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.2: Пример работы файла lab8-1.asm

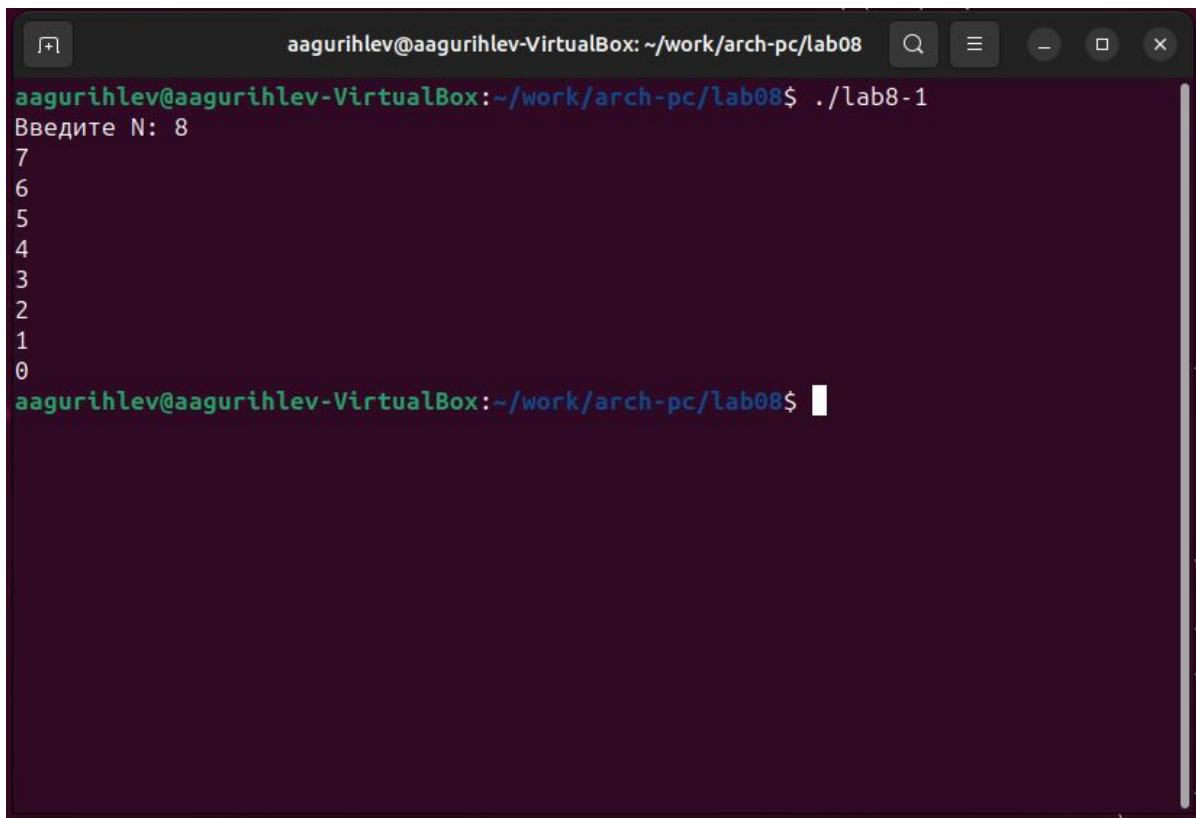
Изменим текст программы, добавив туда строчку с вычитанием из регистра есх единицы, и проверим её работу(рис. [3?]):



```
aagurihlev@aagurihlev-VirtualBox: ~/work/arch-pc/lab08
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 8
7
5
3
1
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.3: Пример работы изменённой программы

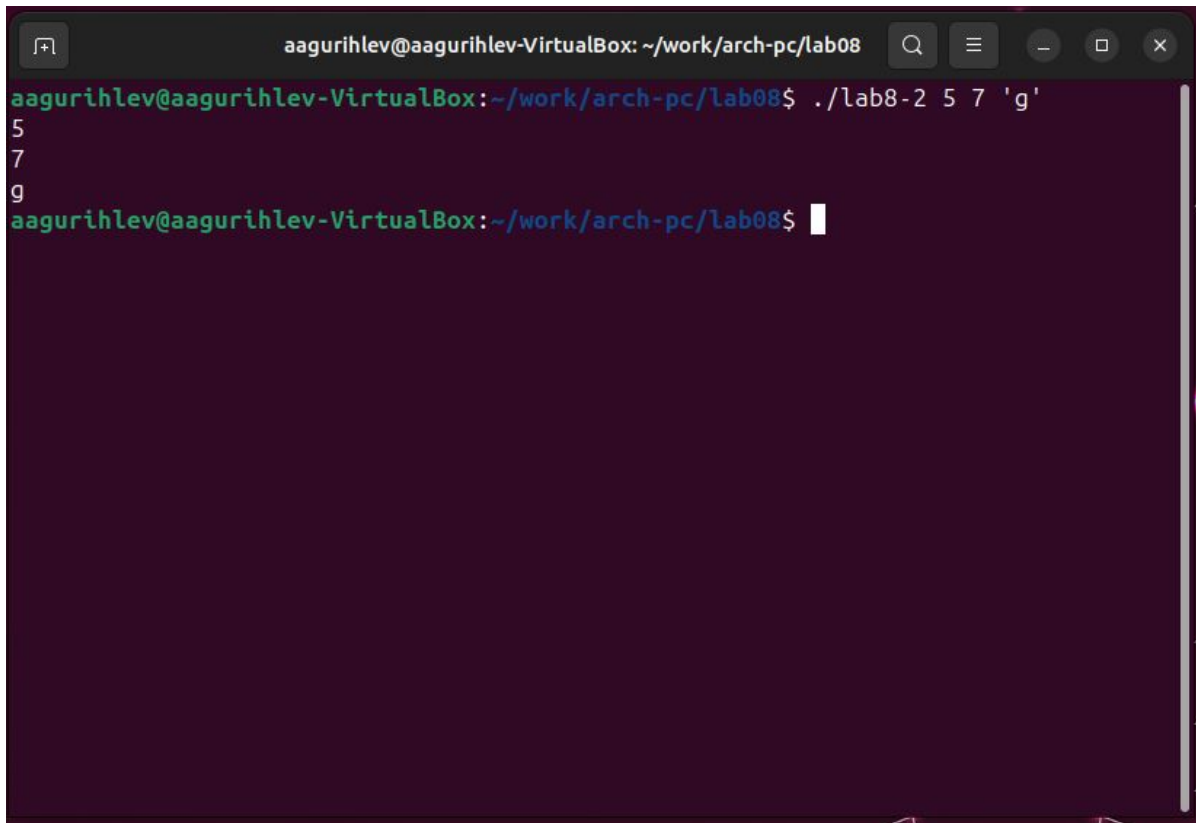
Так как инструкция `loop` всегда сама по себе вычитает единицу из регистра `ecx`, то в программе этого не нужно делать, иначе проходов будет примерно в два раза меньше, так как при каждом из них из `ecx` будет вычитаться 2, а не 1. Теперь изменим код программы, чтобы в ней использовался стек, и проверим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



```
aagurihlev@aagurihlev-VirtualBox: ~/work/arch-pc/lab08
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 8
7
6
5
4
3
2
1
0
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.4: Пример работы программы с использованием стека

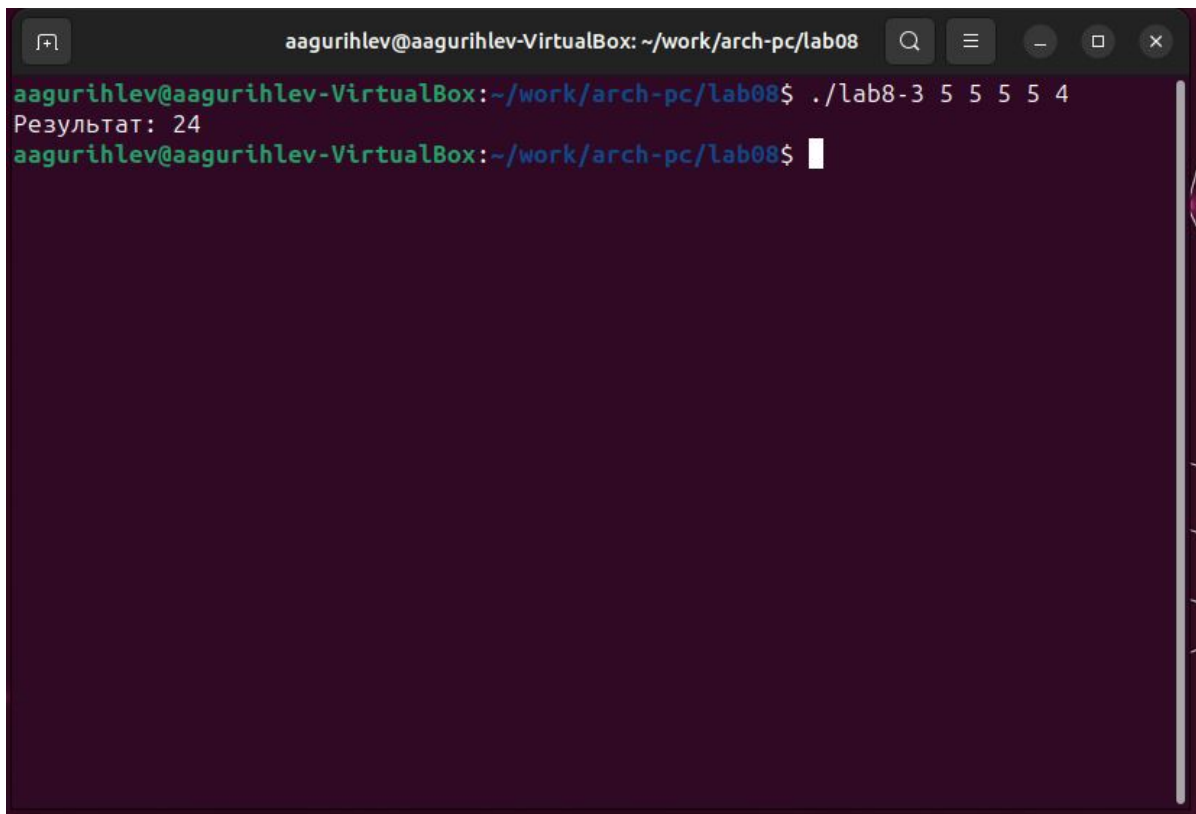
Число проходов соответствует введенному числу. Создадим файл lab8-2.asm, введём из него код из листинга и создадим исполняемый файл, после чего запустим его из командной строки с аргументами(рис. [5?]):

A screenshot of a terminal window titled 'aagurihlev@aagurihlev-VirtualBox: ~/work/arch-pc/lab08'. The terminal shows a command prompt where the user has entered './lab8-2 5 7 'g''. The program has executed and printed the numbers '5', '7', and 'g' on separate lines. The prompt is now ready for the next command.

```
aagurihlev@aagurihlev-VirtualBox: ~/work/arch-pc/lab08
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 5 7 'g'
5
7
g
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.5: 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 исполняемый файл, после чего запустим его из командной строки с аргументами(рис. [6?]):

A terminal window with a dark background and light green text. The window title is 'aagurihlev@aagurihlev-VirtualBox: ~/work/arch-pc/lab08'. The prompt is 'aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08\$'. The command './lab8-3 5 5 5 5 4' has been entered. The output 'Результат: 24' is displayed on the next line. The prompt is now 'aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08\$' with a cursor.

```
aagurihlev@aagurihlev-VirtualBox: ~/work/arch-pc/lab08
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 5 5 5 5 4
Результат: 24
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 2.6: Пример работы программы lab8-3

Программа выводит сумму всех аргументов.

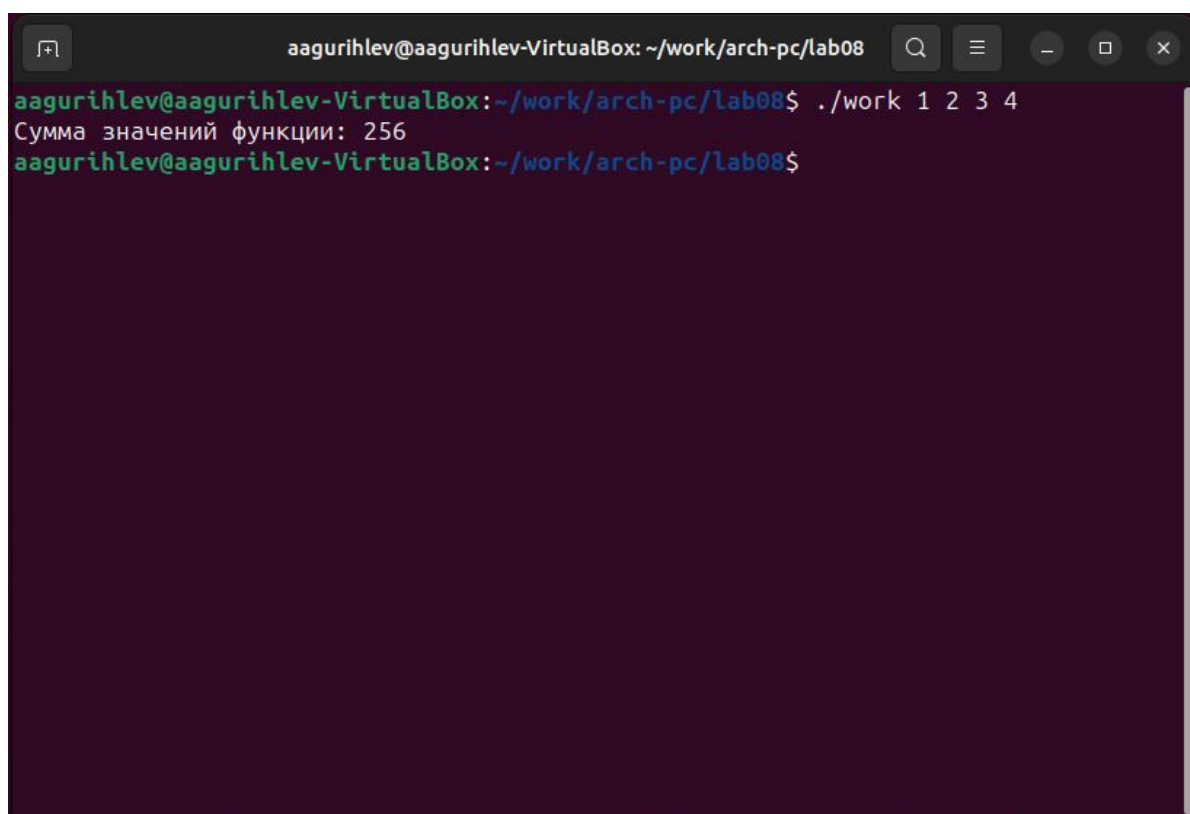
3 Выполнение самостоятельной работы

Мой вариант - 16, следовательно функция, с которой я буду работать - 30х - 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?]):

A terminal window titled 'aagurihlev@aagurihlev-VirtualBox: ~/work/arch-pc/lab08'. The prompt is 'aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08\$'. The user enters './work 1 2 3 4'. The program outputs 'Сумма значений функции: 256'. The prompt returns to 'aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08\$'.

```
aagurihlev@aagurihlev-VirtualBox: ~/work/arch-pc/lab08$ ./work 1 2 3 4
Сумма значений функции: 256
aagurihlev@aagurihlev-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 3.1: 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.