

# **Отчёт по лабораторной работе 8**

**дисциплина: Архитектура компьютера**

Дондоков Анжил Зорикович

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	17

## Список иллюстраций

2.1	Программа lab8-1.asm . . . . .	7
2.2	Запуск программы lab8-1.asm . . . . .	7
2.3	Программа lab8-1.asm . . . . .	8
2.4	Запуск программы lab8-1.asm . . . . .	9
2.5	Программа lab8-1.asm . . . . .	10
2.6	Запуск программы lab8-1.asm . . . . .	11
2.7	Программа lab8-2.asm . . . . .	12
2.8	Запуск программы lab8-2.asm . . . . .	12
2.9	Программа lab8-3.asm . . . . .	13
2.10	Запуск программы lab8-3.asm . . . . .	13
2.11	Программа lab8-3.asm . . . . .	14
2.12	Запуск программы lab8-3.asm . . . . .	14
2.13	Программа lab8-4.asm . . . . .	15
2.14	Запуск программы lab8-4.asm . . . . .	16

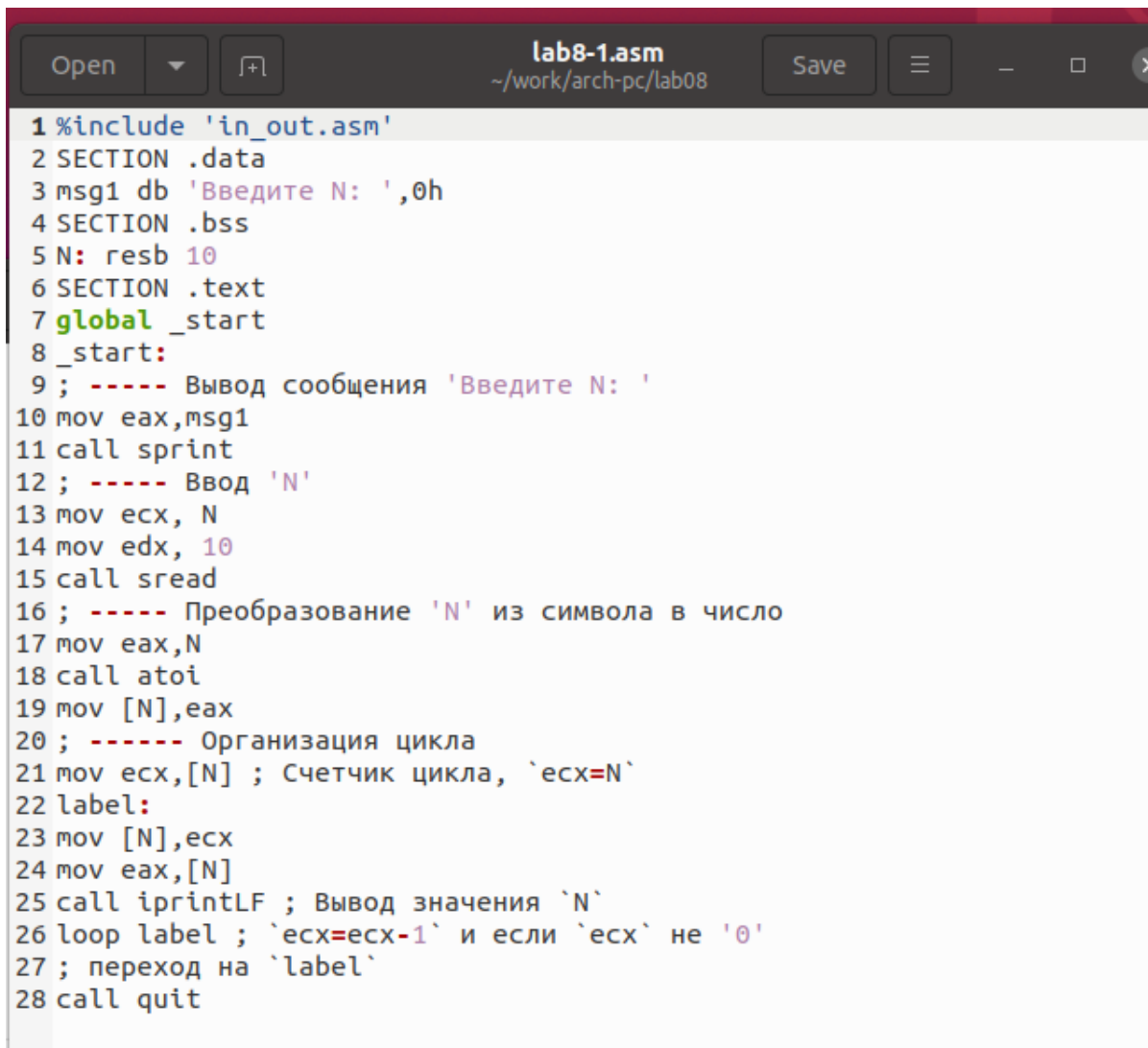
## Список таблиц

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

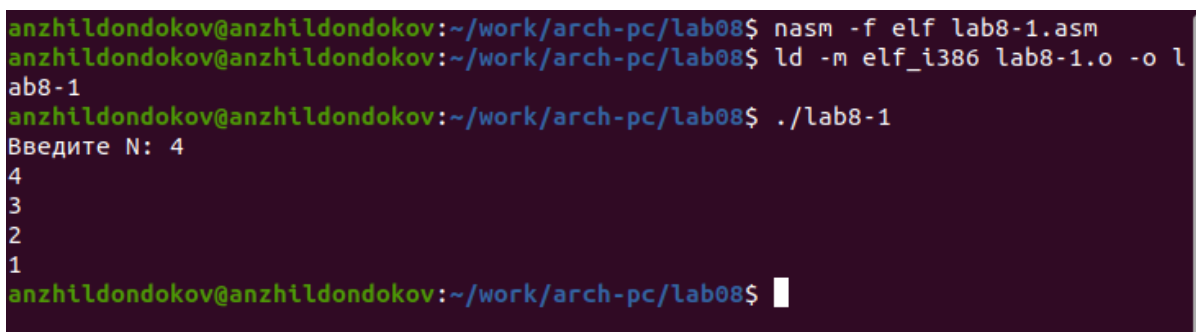
1. Создал каталог для программ лабораторной работы № 8, перешел в него и создал файл lab8-1.asm
2. Написал в файл lab8-1.asm текст программы из листинга 8.1. Создал исполняемый файл и проверил его работу.



```
lab8-1.asm
~/work/arch-pc/lab08

1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 mov [N],ecx
24 mov eax,[N]
25 call iprintLF ; Вывод значения `N`
26 loop label ; `ecx=ecx-1` и если `ecx` не '0'
27 ; переход на `label`
28 call quit
```

Рис. 2.1: Программа lab8-1.asm

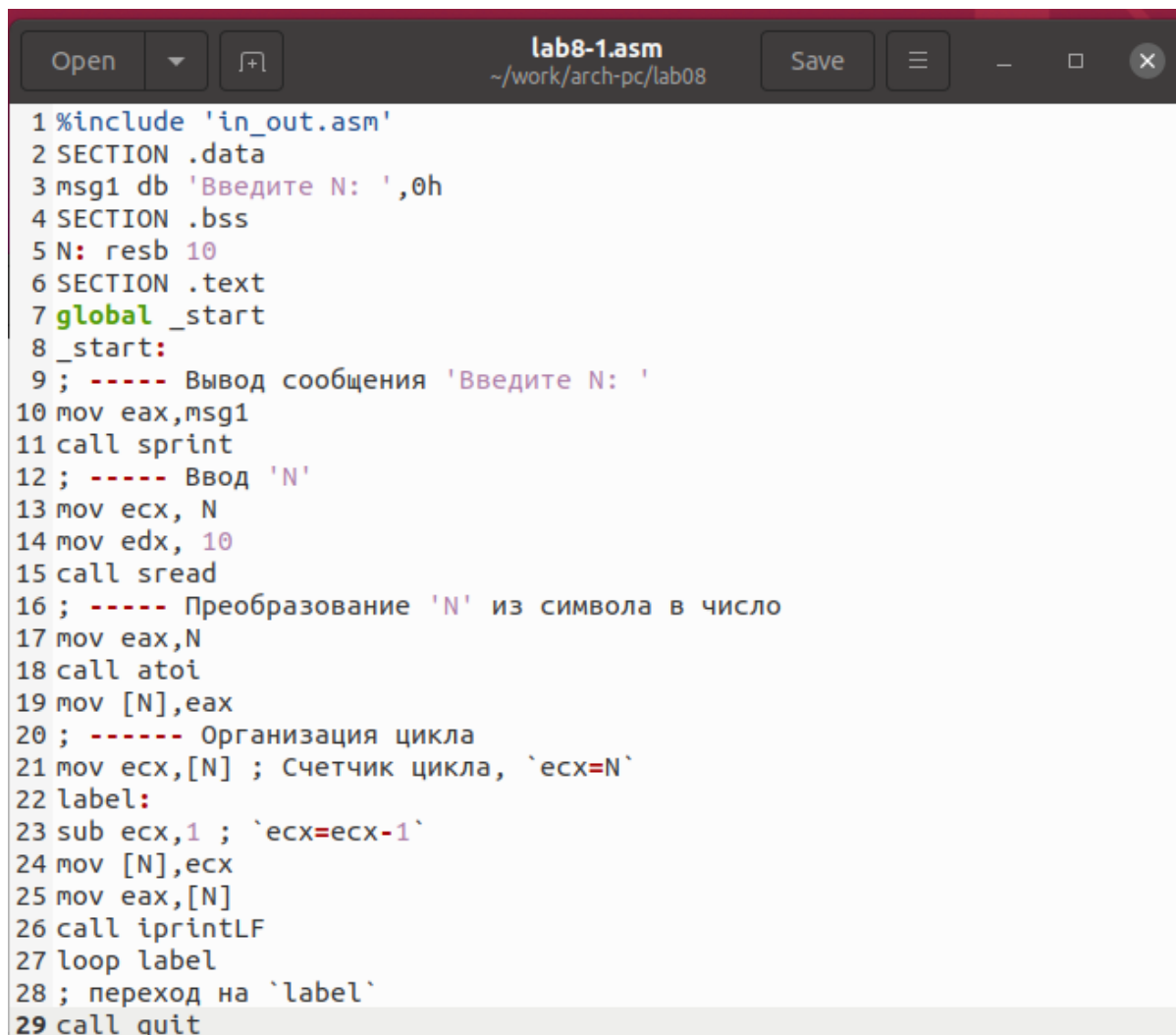


```
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
4
3
2
1
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$
```

Рис. 2.2: Запуск программы lab8-1.asm

3. Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Изменил текст программы добавив изменение значение регистра `ecx` в цикле: Создайте исполняемый файл и проверьте его работу. Какие значения принимает регистр `ecx` в цикле? Соответствует ли число проходов цикла значению `N`, введенному с клавиатуры?

Программа запускает бесконечный цикл при нечетном `N` и выводит только нечетные числа при четном `N`.



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 sub ecx,1 ; `ecx=ecx-1`
24 mov [N],ecx
25 mov eax,[N]
26 call iprintLF
27 loop label
28 ; переход на `label`
29 call quit
```

Рис. 2.3: Программа lab8-1.asm

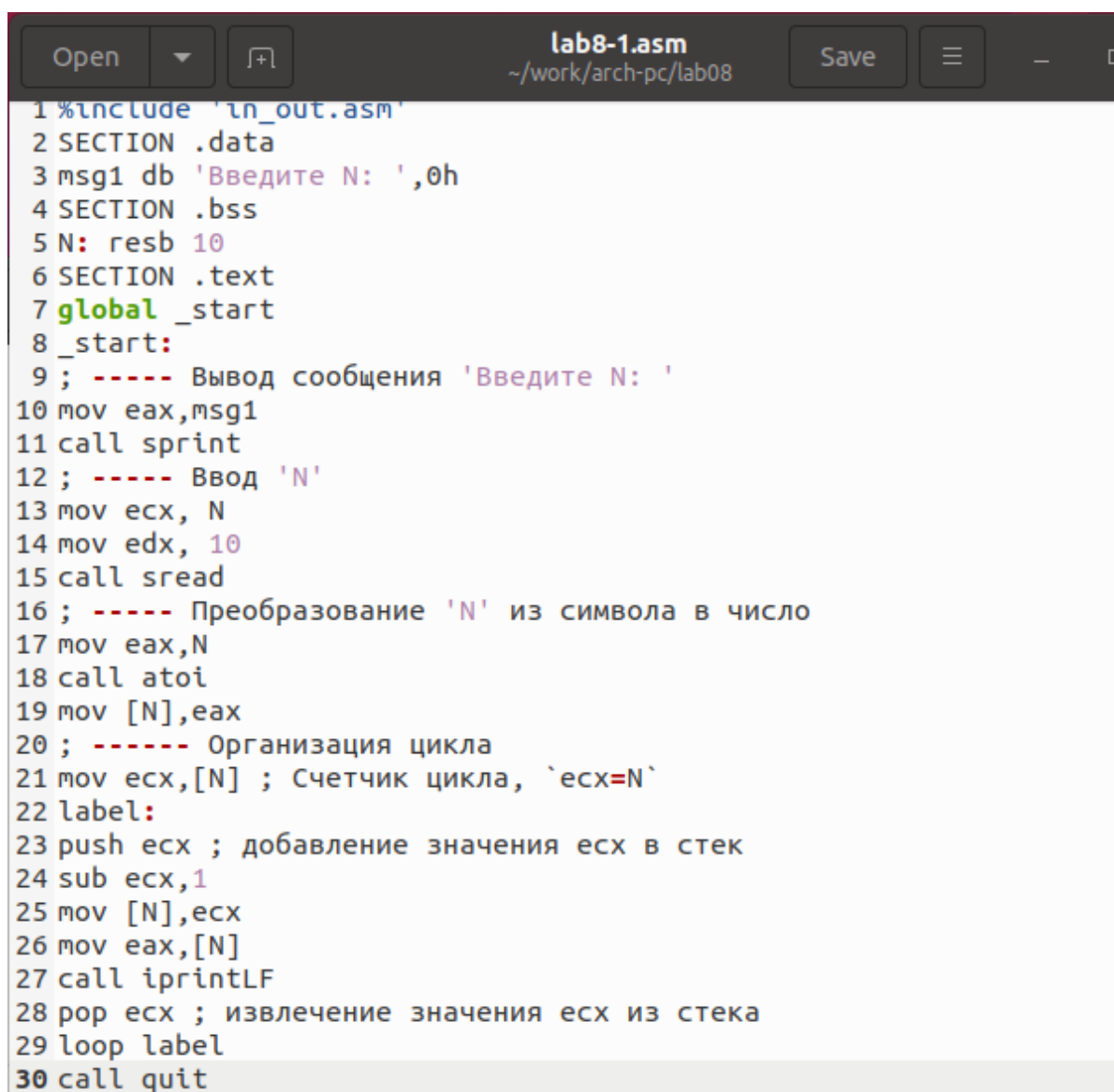


```
4294785480
4294785478
4294785476
4294785474
4294785472
4294785470
4294785468
4294785466
4294^C
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$
```

Рис. 2.4: Запуск программы lab8-1.asm

4. Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Внеси изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`. Создал исполняемый файл и проверьте его работу. Соответствует ли в данном случае число проходов цикла значению `N` введенному с клавиатуры?

Программа выводит числа от `N-1` до `0`, число проходов цикла соответствует `N`.



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 push ecx ; добавление значения ecx в стек
24 sub ecx,1
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF
28 pop ecx ; извлечение значения ecx из стека
29 loop label
30 call quit
```

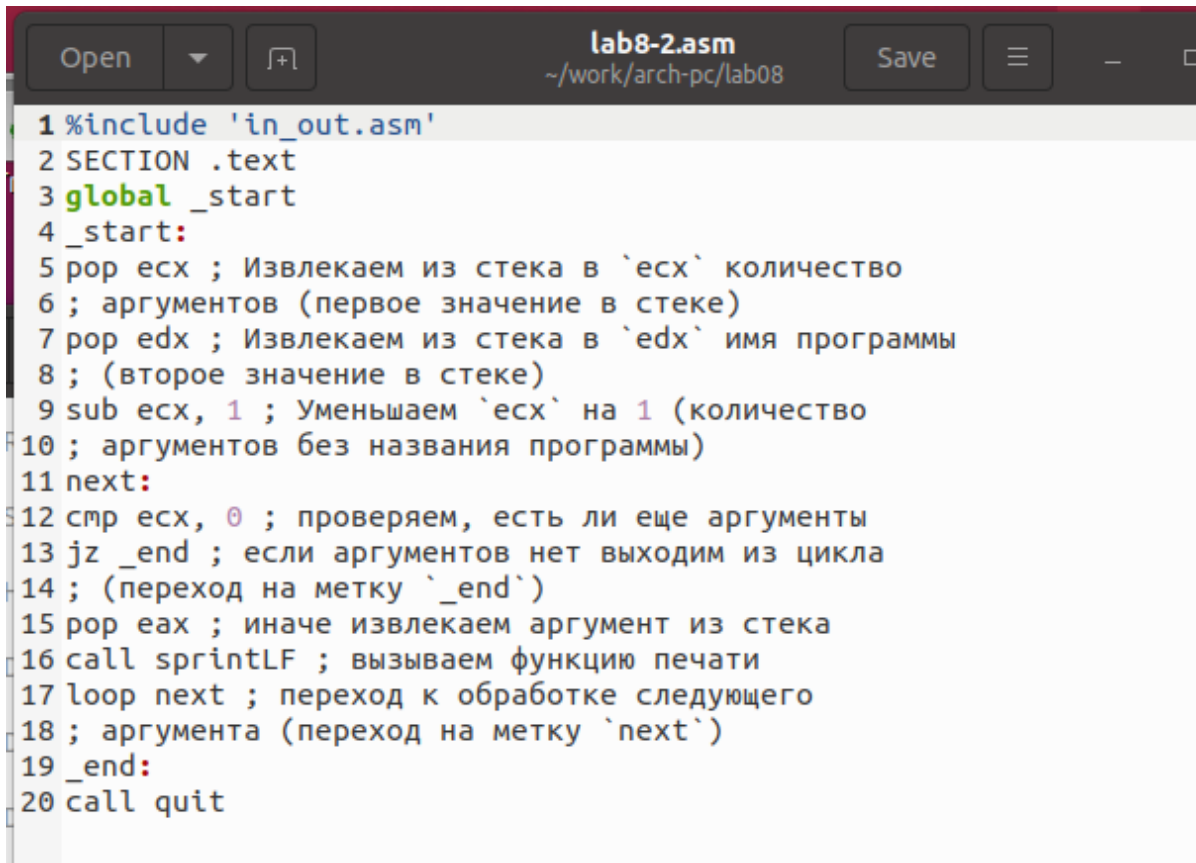
Рис. 2.5: Программа lab8-1.asm

```
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$  
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm  
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1  
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 4  
3  
2  
1  
0  
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 3  
2  
1  
0  
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$
```

Рис. 2.6: Запуск программы lab8-1.asm

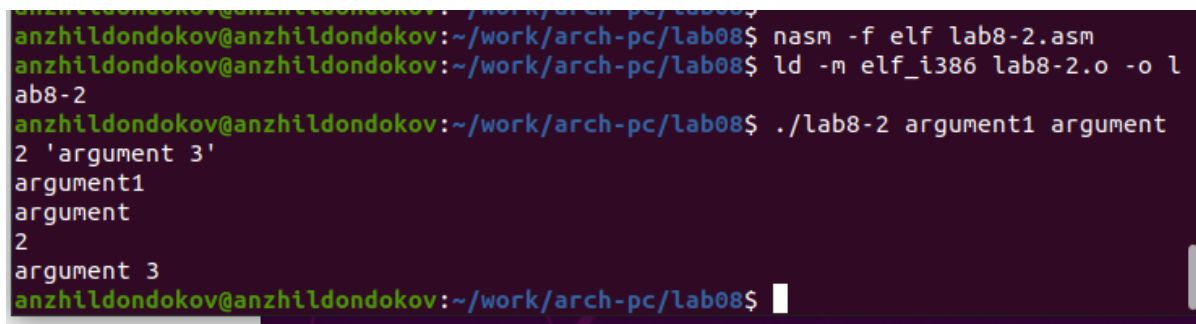
5. Создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввел в него текст программы из листинга 8.2. Создал исполняемый файл и запустил его, указав аргументы. Сколько аргументов было обработано программой?

Программа обработала 5 аргументов.



```
1 %include 'in_out.asm'
2 SECTION .text
3 global _start
4 _start:
5 pop ecx ; Извлекаем из стека в `ecx` количество
6 ; аргументов (первое значение в стеке)
7 pop edx ; Извлекаем из стека в `edx` имя программы
8 ; (второе значение в стеке)
9 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
10 ; аргументов без названия программы)
11 next:
12 cmp ecx, 0 ; проверяем, есть ли еще аргументы
13 jz _end ; если аргументов нет выходим из цикла
14 ; (переход на метку `_end`)
15 pop eax ; иначе извлекаем аргумент из стека
16 call sprintf ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18 ; аргумента (переход на метку `next`)
19 _end:
20 call quit
```

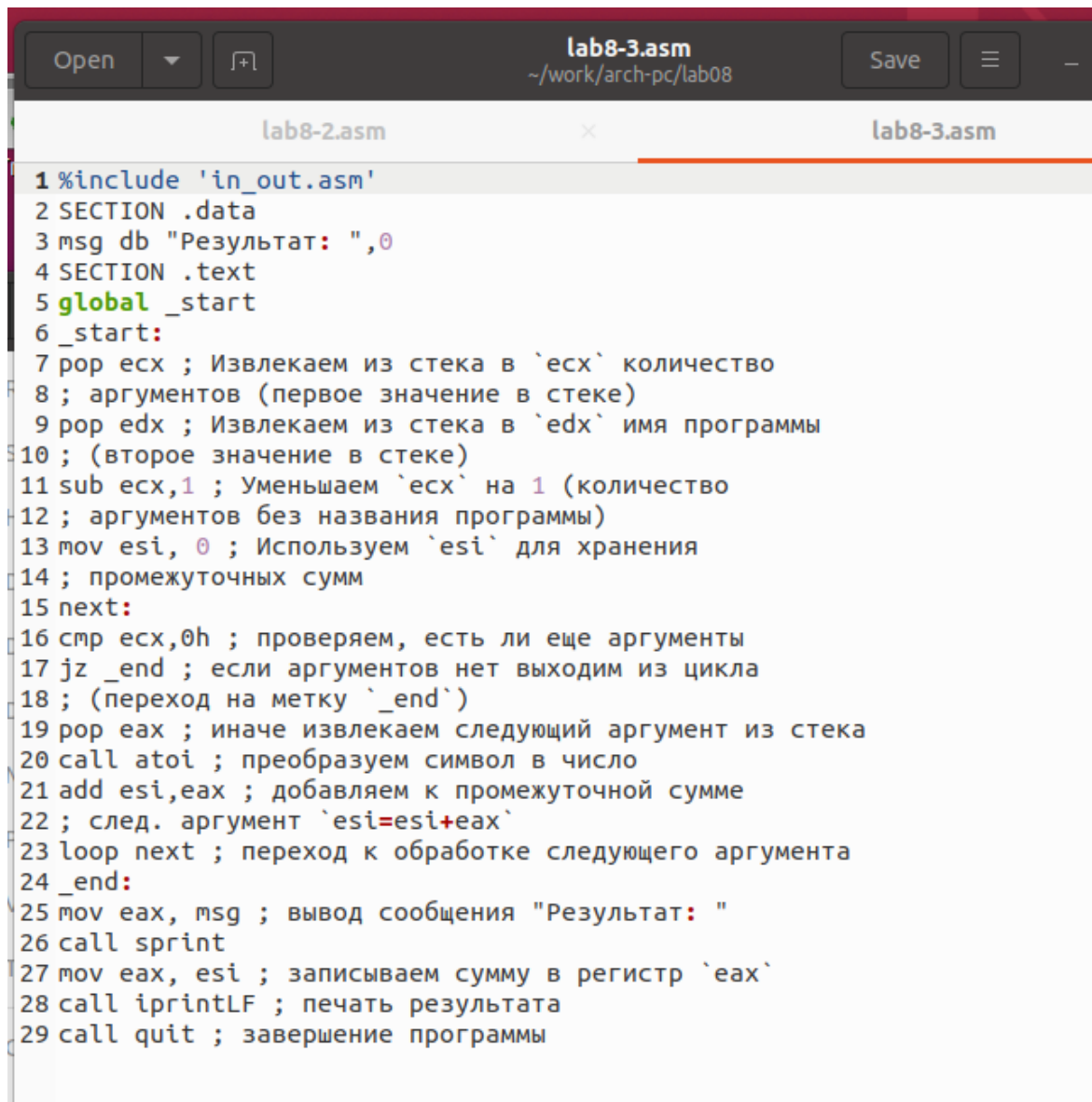
Рис. 2.7: Программа lab8-2.asm



```
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ./lab8-2 argument1 argument
2 'argument 3'
argument1
argument
2
argument 3
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$
```

Рис. 2.8: Запуск программы lab8-2.asm

6. Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы.

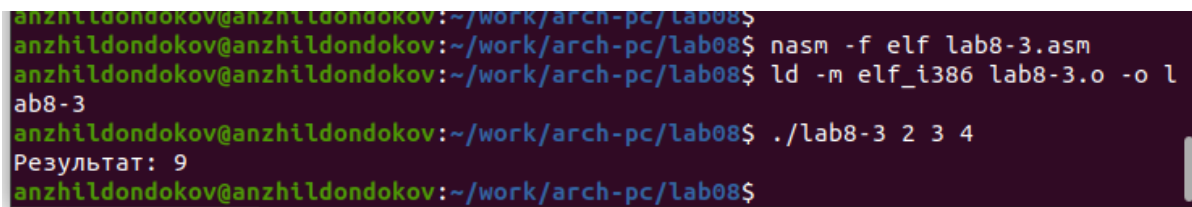


```
lab8-3.asm
~/work/arch-pc/lab08

lab8-2.asm x lab8-3.asm

1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi,0 ; Используем `esi` для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку `_end`)
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 add esi,eax ; добавляем к промежуточной сумме
22 ; след. аргумент `esi=esi+eax`
23 loop next ; переход к обработке следующего аргумента
24 _end:
25 mov eax,msg ; вывод сообщения "Результат: "
26 call sprint
27 mov eax,esi ; записываем сумму в регистр `eax`
28 call iprintLF ; печать результата
29 call quit ; завершение программы
```

Рис. 2.9: Программа lab8-3.asm

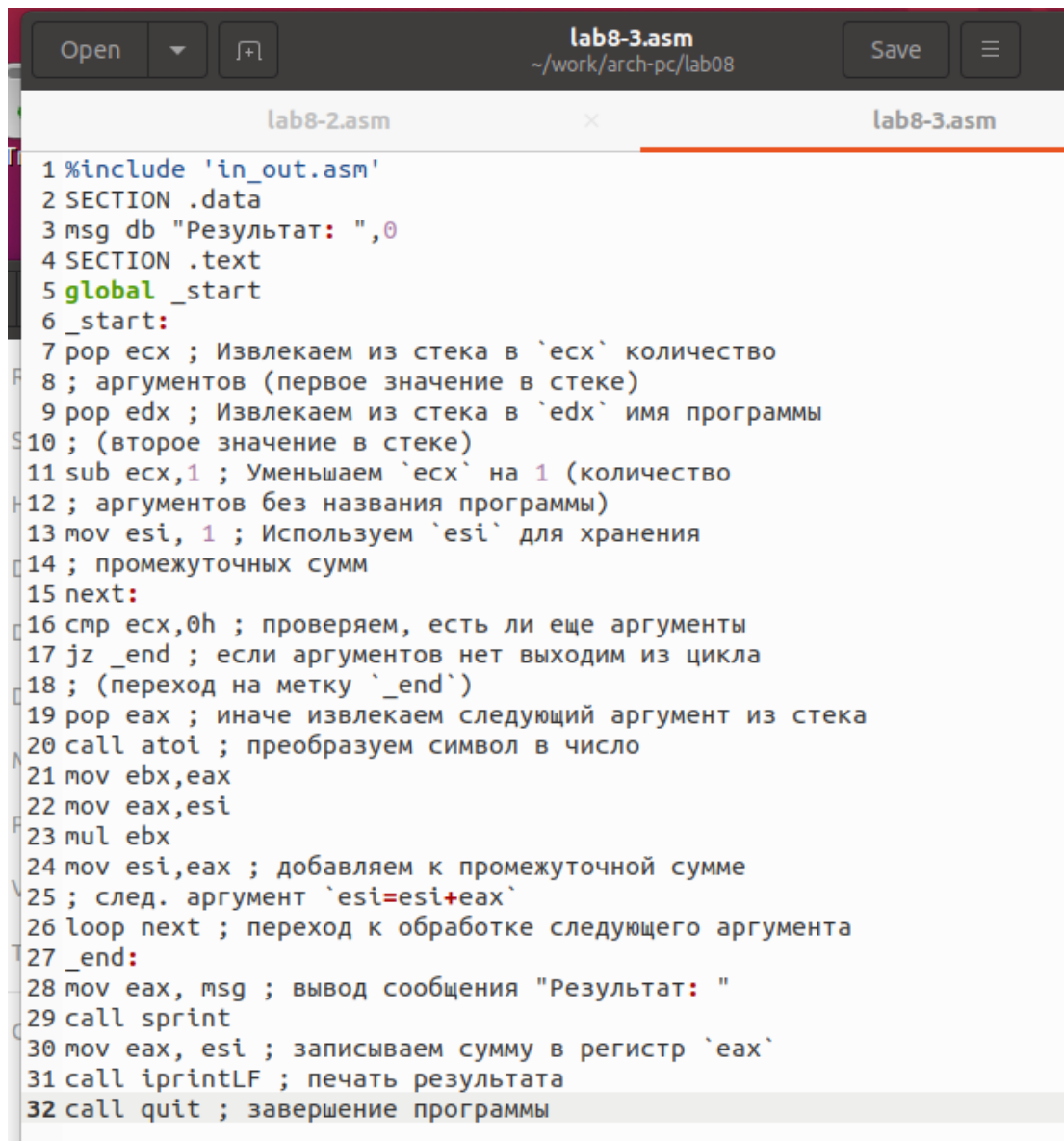


```
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ./lab8-3 2 3 4
Результат: 9
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$
```

Рис. 2.10: Запуск программы lab8-3.asm

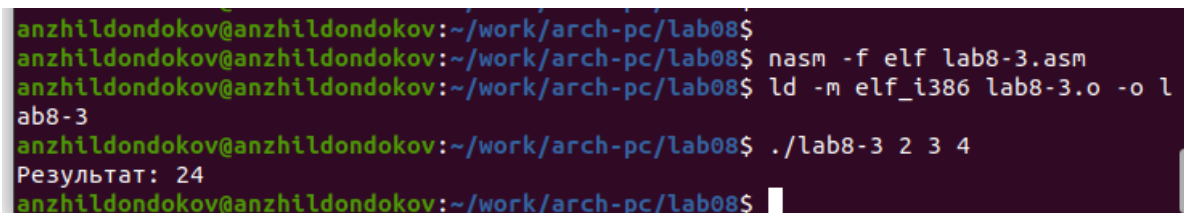
7. Изменил текст программы из листинга 8.3 для вычисления произведения

аргументов командной строки.



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 1 ; Используем `esi` для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку `_end`)
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 mov ebx,eax
22 mov eax,esi
23 mul ebx
24 mov esi,eax ; добавляем к промежуточной сумме
25 ; след. аргумент `esi=esi+eax`
26 loop next ; переход к обработке следующего аргумента
27 _end:
28 mov eax, msg ; вывод сообщения "Результат: "
29 call sprint
30 mov eax, esi ; записываем сумму в регистр `eax`
31 call iprintLF ; печать результата
32 call quit ; завершение программы
```

Рис. 2.11: Программа lab8-3.asm

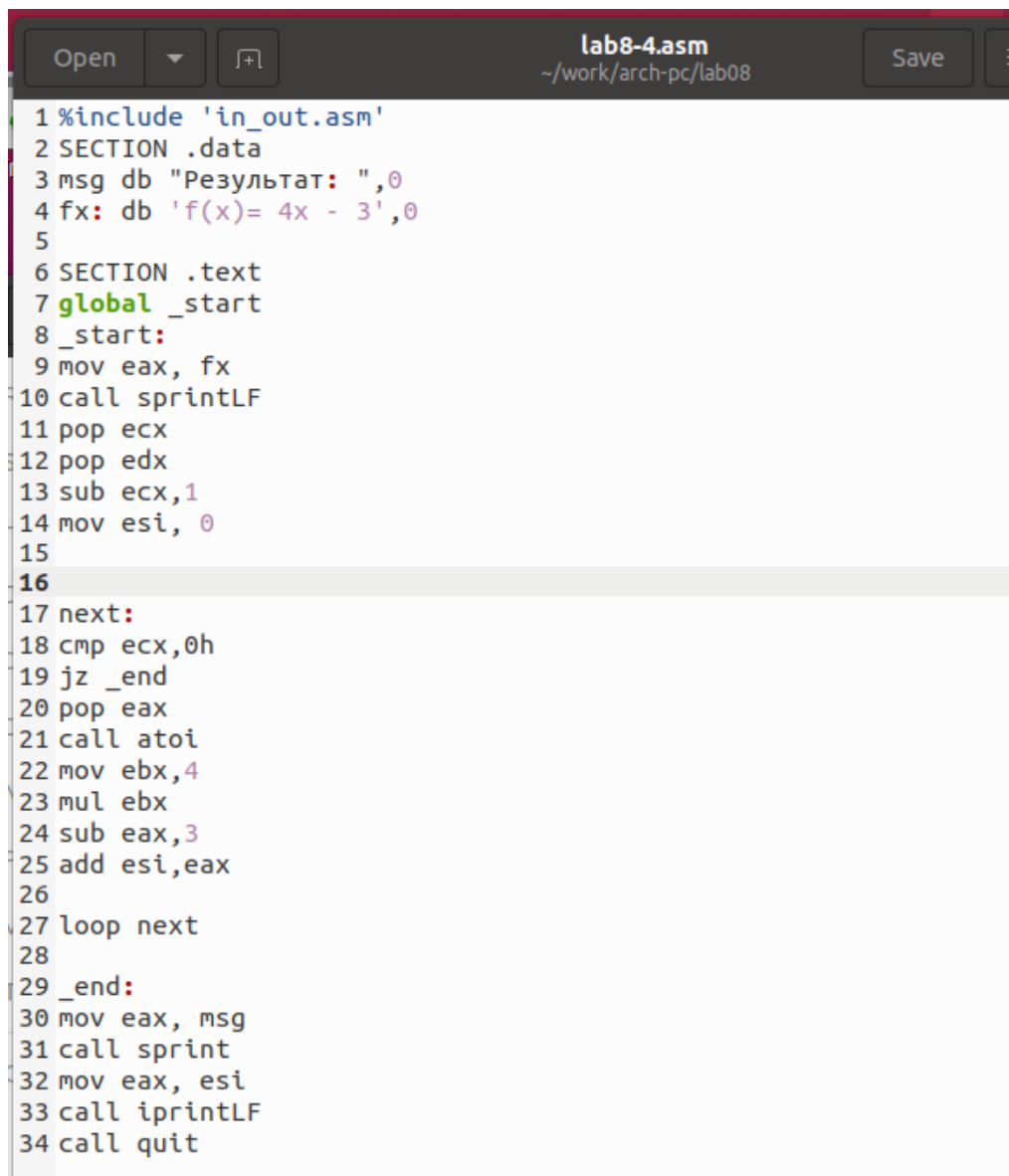


```
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ./lab8-3 2 3 4
Результат: 24
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$
```

Рис. 2.12: Запуск программы lab8-3.asm

8. Напишите программу, которая находит сумму значений функции  $f(x)$  для  $x = x_1, x_2, \dots, x_n$ , т.е. программа должна выводить значение  $f(x_1) + f(x_2) + \dots + f(x_n)$ . Значения  $x$  передаются как аргументы. Вид функции  $f(x)$  выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах  $x$ .

для варианта 6  $f(x) = 4x - 3$



```
lab8-4.asm
~/work/arch-pc/lab08

1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)= 4x - 3',0
5
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintf
11 pop ecx
12 pop edx
13 sub ecx,1
14 mov esi, 0
15
16
17 next:
18 cmp ecx,0h
19 jz _end
20 pop eax
21 call atoi
22 mov ebx,4
23 mul ebx
24 sub eax,3
25 add esi,eax
26
27 loop next
28
29 _end:
30 mov eax, msg
31 call sprintf
32 mov eax, esi
33 call iprintLF
34 call quit
```

Рис. 2.13: Программа lab8-4.asm

```
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-4.o -o lab8-4
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ./lab8-4
f(x)= 4x - 3
Результат: 0
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ./lab8-4 3
f(x)= 4x - 3
Результат: 9
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$ ./lab8-4 3 6 4 9 7
f(x)= 4x - 3
Результат: 101
anzhildondokov@anzhildondokov:~/work/arch-pc/lab08$
```

Рис. 2.14: Запуск программы lab8-4.asm



## 3 Выводы

Освоили работы со стеком, циклом и аргументами на ассемблере `naasm`.