

# **Отчёт по лабораторной работе № 9**

**Архитектура компьютера**

**Егор Рыжов**

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Реализация циклов в NASM . . . . .	6
3.2	Обработка аргументов командной строки . . . . .	11
3.3	Задание для самостоятельной работы . . . . .	14
<b>4</b>	<b>Выводы</b>	<b>16</b>

## Список иллюстраций

3.1	lab9-1.asm . . . . .	6
3.2	Текст программы . . . . .	7
3.3	Исполняемый файл . . . . .	7
3.4	Текст программы . . . . .	8
3.5	Исполняемый файл . . . . .	8
3.6	Значения в цикле . . . . .	9
3.7	Текст программы . . . . .	10
3.8	Исполняемый файл . . . . .	10
3.9	lab9-2.asm . . . . .	11
3.10	Текст программы . . . . .	12
3.11	Исполняемый файл . . . . .	12
3.12	lab9-3.asm . . . . .	12
3.13	Текст программы . . . . .	13
3.14	Исполняемый файл . . . . .	13
3.15	Текст программы . . . . .	14
3.16	Исполняемый файл . . . . .	14
3.17	Текст программы . . . . .	15
3.18	Исполняемый файл . . . . .	15

# 1 Цель работы

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

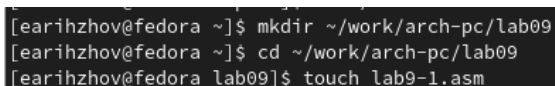
## 2 Задание

1. Реализовать циклы в NASM
2. Выполнить обработку аргументов командной строки
3. Выполнить задание для самостоятельной работы

## 3 Выполнение лабораторной работы

### 3.1 Реализация циклов в NASM

Создали каталог для программ лабораторной работы № 9, перейшли в него и создали файл lab9-1.asm: (рис. 3.1)

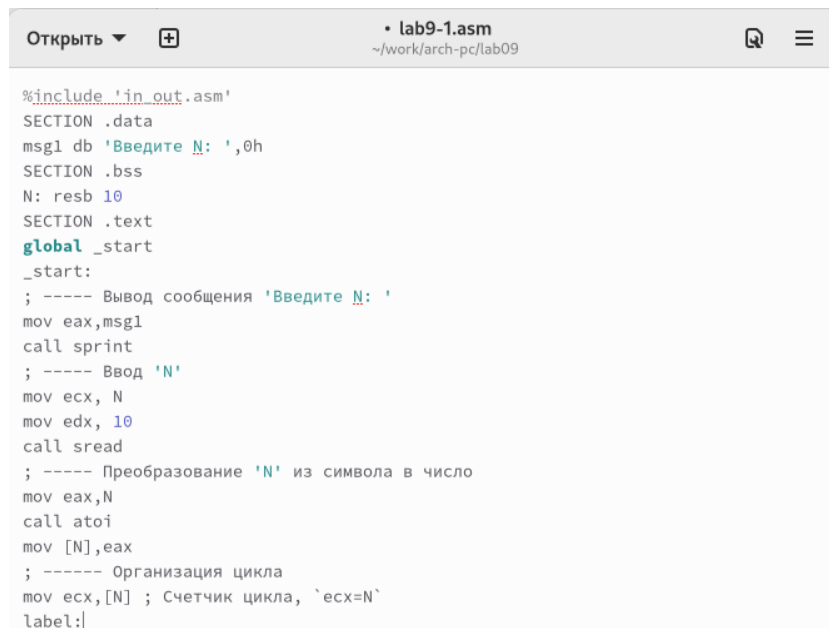


```
[earihzhov@fedora ~]$ mkdir ~/work/arch-pc/lab09  
[earihzhov@fedora ~]$ cd ~/work/arch-pc/lab09  
[earihzhov@fedora lab09]$ touch lab9-1.asm
```

Рис. 3.1: lab9-1.asm

При реализации циклов в NASM с использованием инструкции `loop` необходимо помнить о том, что эта инструкция использует регистр `ecx` в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрели программу, которая выводит значение регистра `ecx`. Внимательно изучили текст программы (Листинг 9.1).

Ввели в файл `lab9-1.asm` текст программы из листинга 9.1. (рис. 3.2) Создали исполняемый файл и проверили его работу. (рис. 3.3)




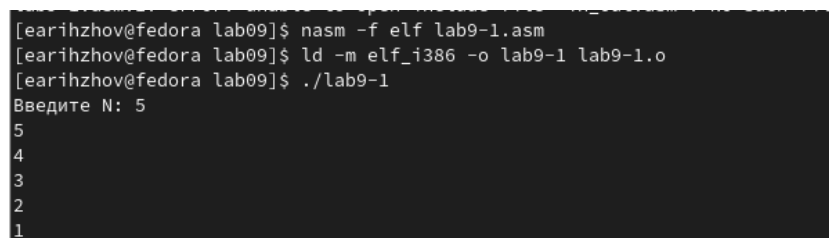
```
Открыть ▾  • lab9-1.asm  
~/work/arch-pc/lab09  
  
%include 'in_out.asm'  
SECTION .data  
msg1 db 'Введите N: ',0h  
SECTION .bss  
N: resb 10  
SECTION .text  
global _start  
_start:  
; ----- Вывод сообщения 'Введите N: '  
mov eax,msg1  
call sprint  
; ----- Ввод 'N'  
mov ecx, N  
mov edx, 10  
call sread  
; ----- Преобразование 'N' из символа в число  
mov eax,N  
call atoi  
mov [N],eax  
; ----- Организация цикла  
mov ecx,[N] ; Счетчик цикла, `ecx=N`  
label:|
```

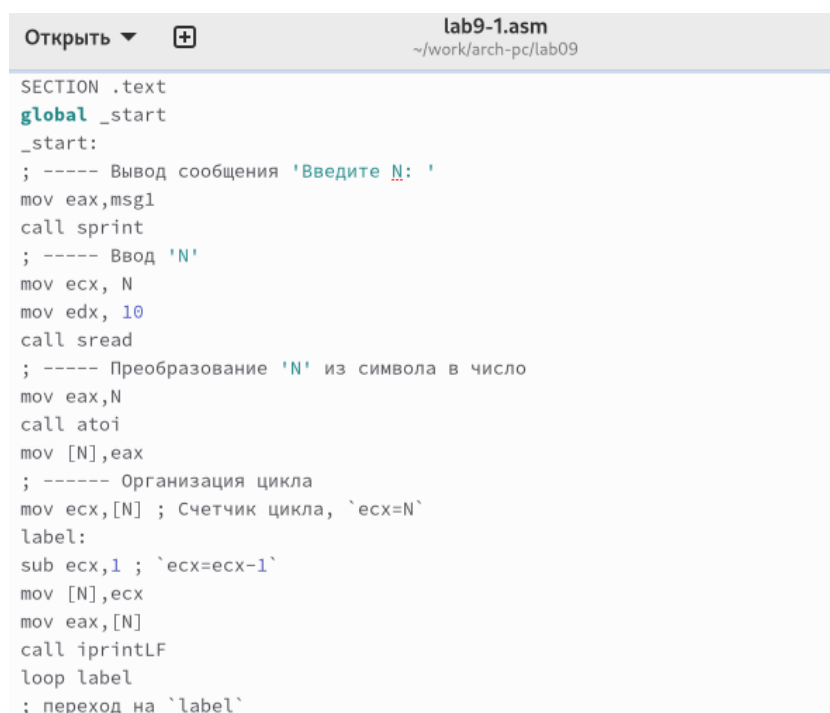
Рис. 3.2: Текст программы



```
[earihzhov@fedora lab09]$ nasm -f elf lab9-1.asm  
[earihzhov@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o  
[earihzhov@fedora lab09]$ ./lab9-1  
Введите N: 5  
5  
4  
3  
2  
1
```

Рис. 3.3: Исполняемый файл

Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Изменили текст программы добавив изменение значения регистра `ecx` в цикле: (рис. 3.4)



```

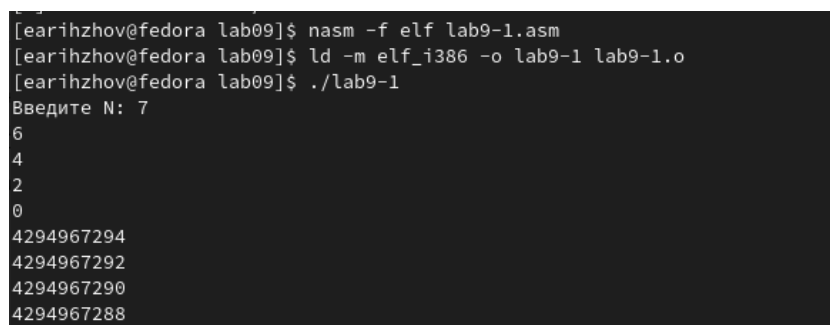
Открыть ▾ + lab9-1.asm
~/work/arch-pc/lab09

SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
; переход на `label`

```

Рис. 3.4: Текст программы

Создали исполняемый файл и проверили его работу. (рис. 3.5) Регистр `ecx` принимает следующие значения в цикле: (рис. 3.6). Число проходов цикла не соответствует значению `N` введенному с клавиатуры.



```

[earihzhov@fedora lab09]$ nasm -f elf lab9-1.asm
[earihzhov@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[earihzhov@fedora lab09]$ ./lab9-1
Введите N: 7
6
4
2
0
4294967294
4294967292
4294967290
4294967288

```

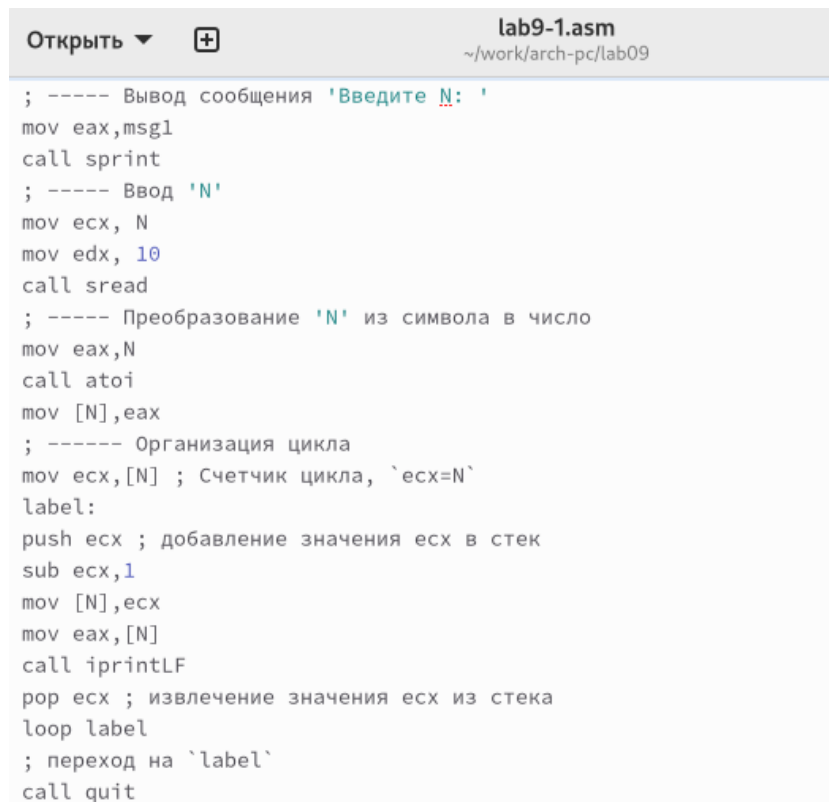
Рис. 3.5: Исполняемый файл



```
4294915286
4294915284
4294915282
4294915280
4294915278
4294915276
4294915274
4294915272
4294915270
4294915268
4294915266
4294915264
4294915262
4294915260
4294915258
4294915256
4294915254
4294915252
4294915250
4294915248
4294915246
4294915244
4294915242
4294915240
4294915238
4294915236
4294915234
4294915232
```

Рис. 3.6: Значения в цикле

Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Внесли изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`: (рис. 3.7)




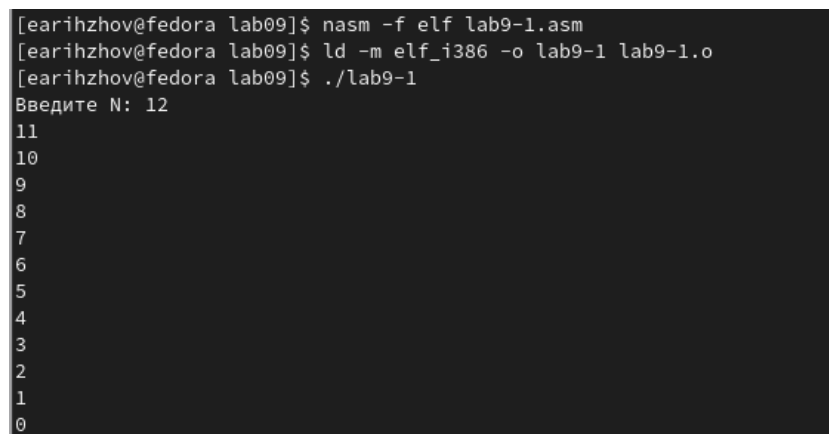
```
Открыть ▾  lab9-1.asm  
~/work/arch-pc/lab09  
; ----- Вывод сообщения 'Введите N: '  
mov eax,msg1  
call sprint  
; ----- Ввод 'N'  
mov ecx, N  
mov edx, 10  
call sread  
; ----- Преобразование 'N' из символа в число  
mov eax,N  
call atoi  
mov [N],eax  
; ----- Организация цикла  
mov ecx,[N] ; Счетчик цикла, `ecx=N`  
label:  
push ecx ; добавление значения ecx в стек  
sub ecx,1  
mov [N],ecx  
mov eax,[N]  
call iprintLF  
pop ecx ; извлечение значения ecx из стека  
loop label  
; переход на `label`  
call quit
```

Рис. 3.7: Текст программы

Создали исполняемый файл и проверили его работу. (рис. 3.8) В данном случае число проходов цикла соответствует значению N введенному с клавиатуры.



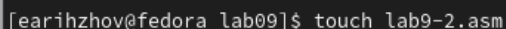
```
[earihzhov@fedora lab09]$ nasm -f elf lab9-1.asm  
[earihzhov@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o  
[earihzhov@fedora lab09]$ ./lab9-1  
Введите N: 12  
11  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1  
0
```

Рис. 3.8: Исполняемый файл

## 3.2 Обработка аргументов командной строки

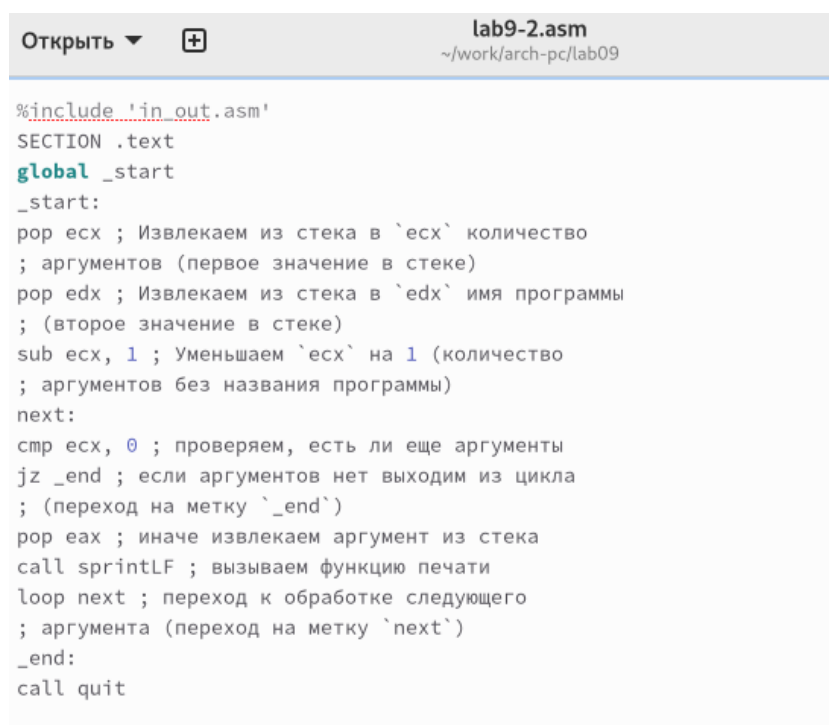
При разработке программ иногда возникает необходимость указывать аргументы, которые будут использоваться в программе, непосредственно из командной строки при запуске программы. При запуске программы в NASM аргументы командной строки загружаются в стек в обратном порядке, кроме того в стек записывается имя программы и общее количество аргументов. Последние два элемента стека для программы, скомпилированной NASM, – это всегда имя программы и количество переданных аргументов. Таким образом, для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. Т.е. сначала нужно извлечь из стека количество аргументов, а затем циклично для каждого аргумента выполнить логику программы. В качестве примера рассмотрим программу, которая выводит на экран аргументы командной строки. Внимательно изучили текст программы (Листинг 9.2).

Создали файл lab9-2.asm в каталоге ~/work/arch-pc/lab09 и ввели в него текст программы из листинга 9.2. (рис. 3.9), (рис. 3.10) Создали исполняемый файл и запустили его, указав аргументы: (рис. 3.11)



```
[earihzhov@fedora lab09]$ touch lab9-2.asm
```

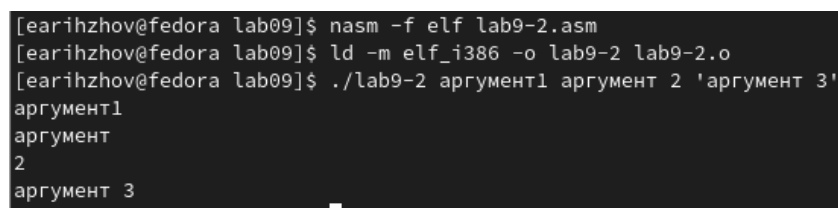
Рис. 3.9: lab9-2.asm



```
Открыть ▾ + lab9-2.asm
~/work/arch-pc/lab09

%include 'in_out.asm'
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем аргумент из стека
call sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

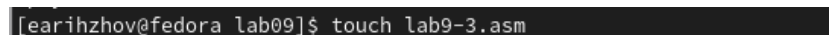
Рис. 3.10: Текст программы



```
[earihzhov@fedora lab09]$ nasm -f elf lab9-2.asm
[earihzhov@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[earihzhov@fedora lab09]$ ./lab9-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

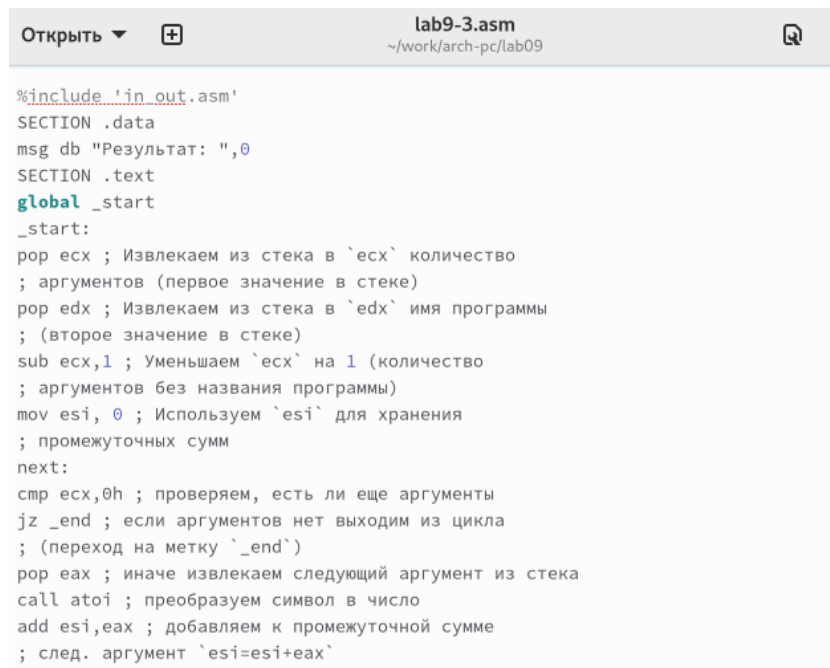
Рис. 3.11: Исполняемый файл

Четыре аргумента было обработано программой. Рассмотрели еще один пример программы, которая выводит сумму чисел, которые передаются в программу как аргументы. Создали файл lab9-3.asm в каталоге ~/work/arch-pc/lab09 и ввели в него текст программы из листинга 9.3. (рис. 3.12), (рис. 3.13)



```
[earihzhov@fedora lab09]$ touch lab9-3.asm
```

Рис. 3.12: lab9-3.asm

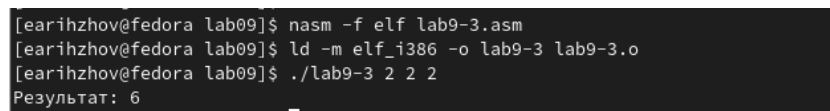


```
Открыть + lab9-3.asm
~/work/arch-pc/lab09

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 0 ; Используем `esi` для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`
; ...
```

Рис. 3.13: Текст программы

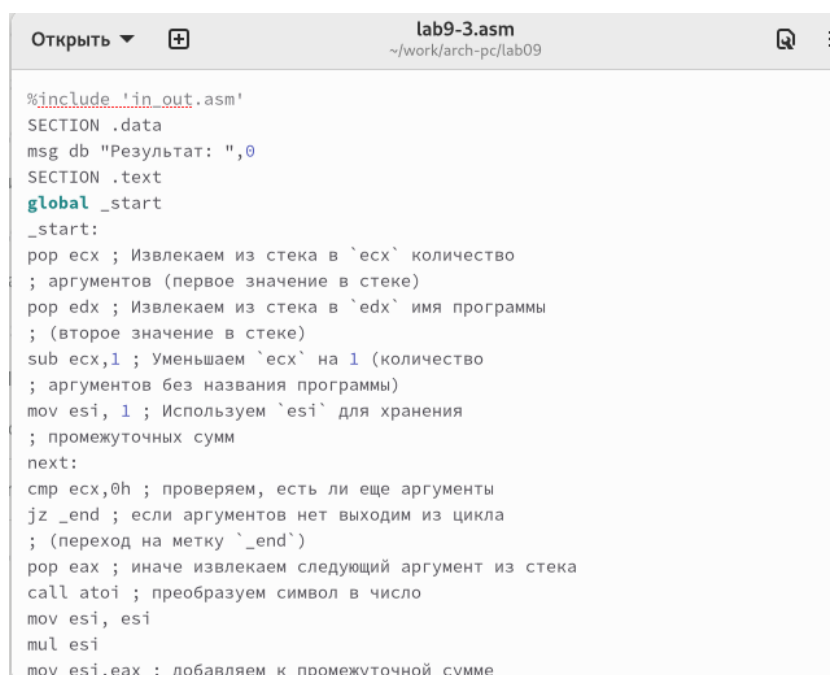
Создали исполняемый файл и запустили его, указав аргументы. (рис. 3.14)



```
[earihzhov@fedora lab09]$ nasm -f elf lab9-3.asm
[earihzhov@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[earihzhov@fedora lab09]$ ./lab9-3 2 2 2
Результат: 6
```

Рис. 3.14: Исполняемый файл

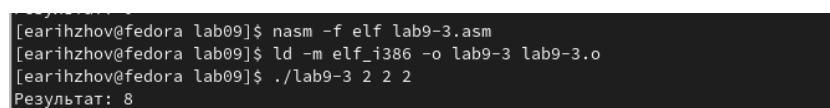
Изменили текст программы из листинга 9.3 для вычисления произведения аргументов командной строки. (рис. 3.15), (рис. 3.16)



```
Открыть + lab9-3.asm
~/work/arch-pc/lab09

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 1 ; Используем `esi` для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
mov esi, esi
mul esi
mov esi,eax ; добавляем к промежуточной сумме
```

Рис. 3.15: Текст программы



```
[earihzhov@fedora lab09]$ nasm -f elf lab9-3.asm
[earihzhov@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[earihzhov@fedora lab09]$ ./lab9-3 2 2 2
Результат: 8
```

Рис. 3.16: Исполняемый файл

### 3.3 Задание для самостоятельной работы

Написали 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  $x_i$  передаются как аргументы. Вид функции  $f(x)$  выбранной из таблицы 9.1 вариантов заданий в соответствии с вариантом 7, полученным при выполнении лабораторной работы № 7. Создали исполняемый файл и проверили его работу на нескольких наборах. (рис. 3.17), (рис. 3.18)



## 4 Выводы

В ходе выполнения лабораторной работы были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.