

Отчёта по лабораторной работе 9

Программирование цикла. Обработка аргументов командной строки.

Одеджими Олуваколаде Осемудиам

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	19

Список иллюстраций

2.1	Файл lab9-1.asm	7
2.2	Работа программы lab9-1.asm	8
2.3	Файл lab9-1.asm	9
2.4	Работа программы lab9-1.asm	10
2.5	Файл lab9-1.asm	11
2.6	Работа программы lab9-1.asm	12
2.7	Файл lab9-2.asm	13
2.8	Работа программы lab9-2.asm	13
2.9	Файл lab9-3.asm	14
2.10	Работа программы lab9-3.asm	14
2.11	Файл lab9-3.asm	15
2.12	Работа программы lab9-3.asm	15
2.13	Файл lab9-4.asm	17
2.14	Работа программы lab9-4.asm	18

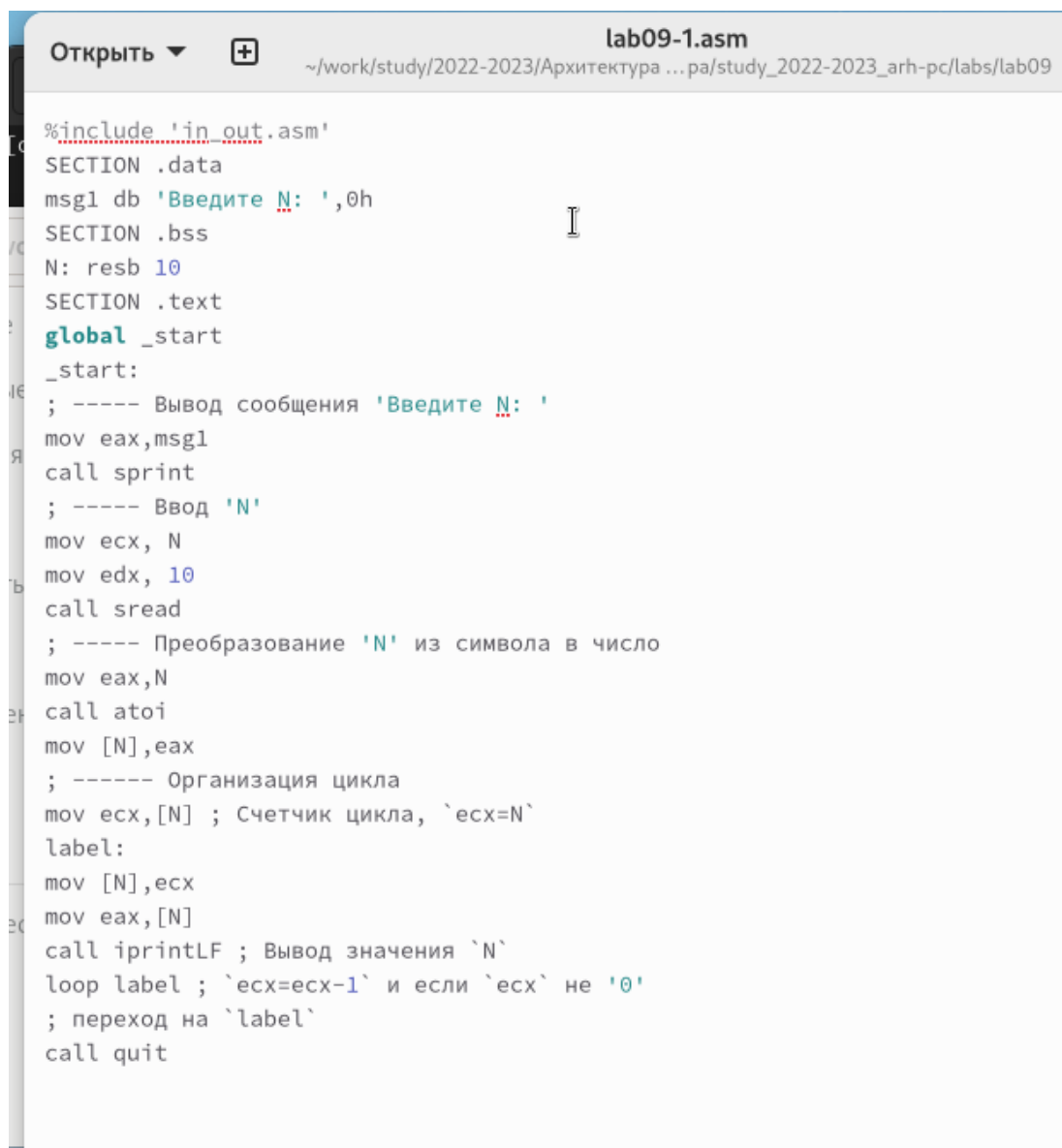
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

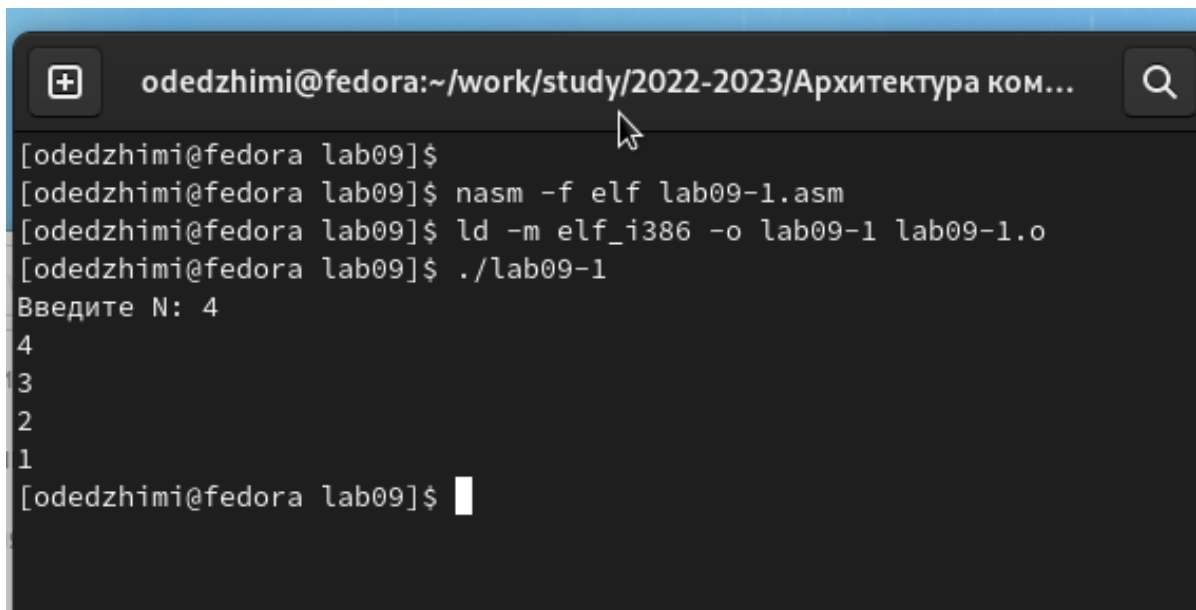
1. Создайте каталог для программ лабораторной работы № 9, перейдите в него и создайте файл lab9-1.asm
2. Введите в файл lab9-1.asm текст программы из листинга 9.1. Создайте исполняемый файл и проверьте его работу. (рис. [2.1], [2.2])



```
lab09-1.asm
~/work/study/2022-2023/Архитектура ...pa/study_2022-2023_arh-pc/labs/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:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не `0`
; переход на `label`
call quit
```

Рис. 2.1: Файл lab9-1.asm

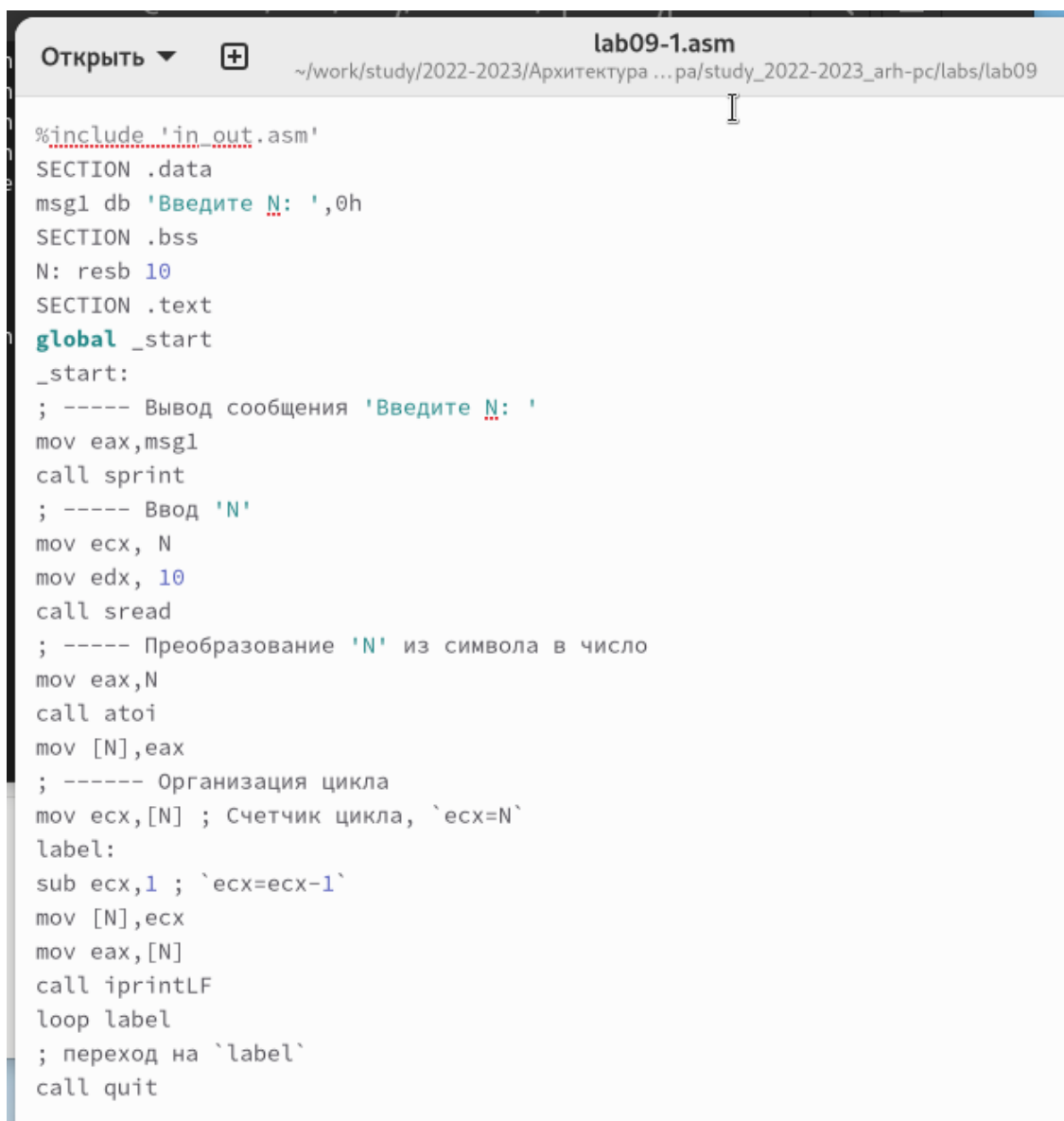


```
odedzhimi@fedora:~/work/study/2022-2023/Архитектура ком...
[odedzhimi@fedora lab09]$
[odedzhimi@fedora lab09]$ nasm -f elf lab09-1.asm
[odedzhimi@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[odedzhimi@fedora lab09]$ ./lab09-1
Введите N: 4
4
3
2
1
[odedzhimi@fedora lab09]$
```

Рис. 2.2: Работа программы lab9-1.asm

3. Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Измените 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? (рис. [2.3], [2.4])

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



```
lab09-1.asm
~/work/study/2022-2023/Архитектура ...pa/study_2022-2023_arh-pc/labs/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:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
; переход на `label`
call quit
```

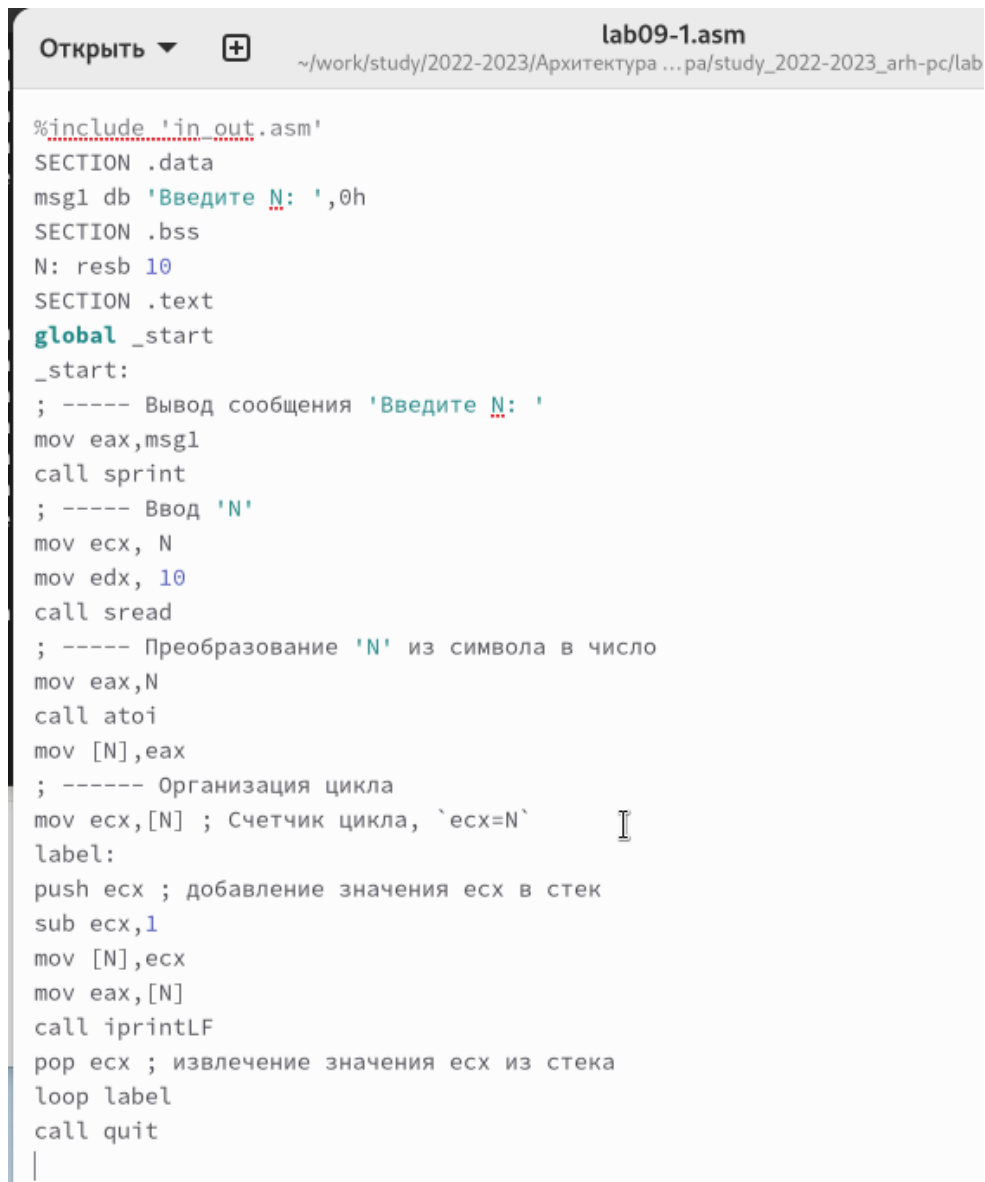
Рис. 2.3: Файл lab9-1.asm

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

Рис. 2.4: Работа программы lab9-1.asm

4. Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Внесите изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`. Создайте исполняемый файл и проверьте его работу. Соответствует ли в данном случае число проходов цикла значению `N` введенному с клавиатуры? (рис. [2.5], [2.6])

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



```
Открыть ▾ + lab09-1.asm
~/work/study/2022-2023/Архитектура ...pa/study_2022-2023_arh-пс/lab

%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:
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
call quit
|
```

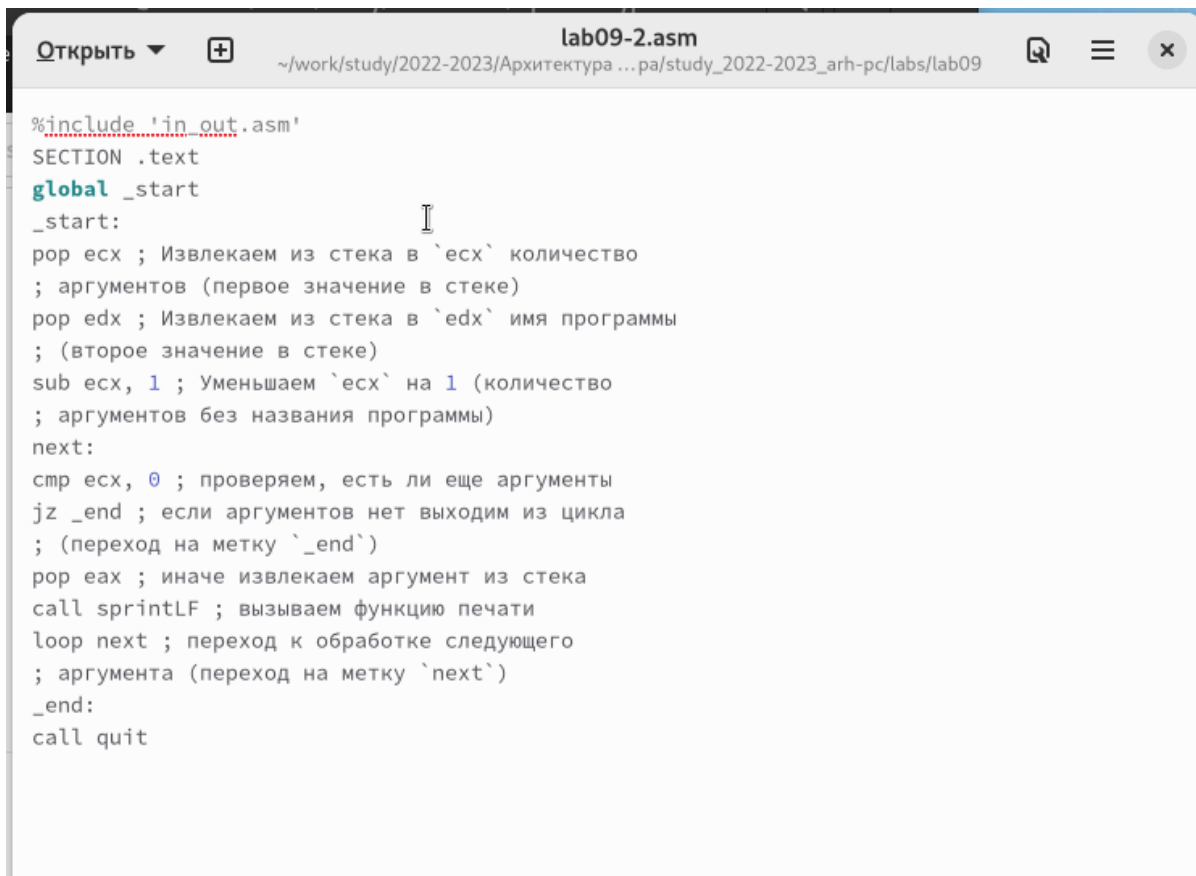
Рис. 2.5: Файл lab9-1.asm

```
[odedzhimi@fedora lab09]$  
[odedzhimi@fedora lab09]$ nasm -f elf lab09-1.asm  
[odedzhimi@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o  
[odedzhimi@fedora lab09]$ ./lab09-1  
Введите N: 4  
3  
2  
1  
0  
[odedzhimi@fedora lab09]$
```

Рис. 2.6: Работа программы lab9-1.asm

5. Создайте файл lab9-2.asm в каталоге ~/work/arch-pc/lab09 и введите в него текст программы из листинга 9.2. Создайте исполняемый файл и запустите его, указав аргументы. (рис. [2.7], [2.8]) Сколько аргументов было обработано программой?

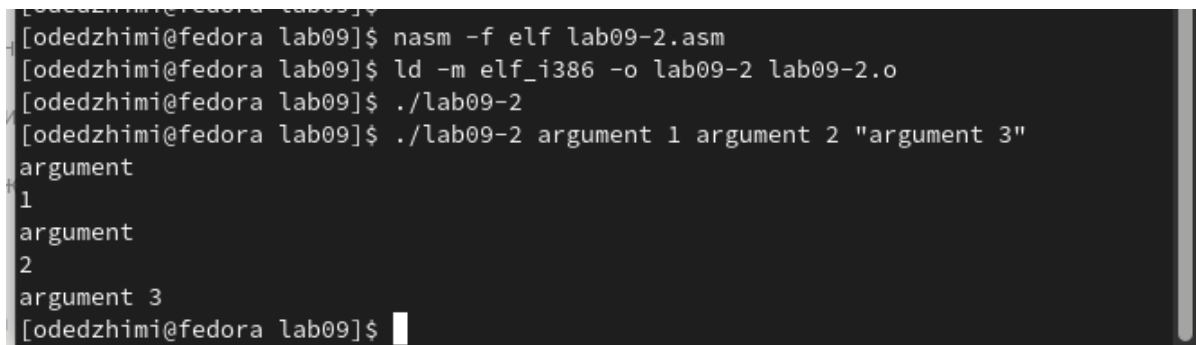
Программа обработала 5 аргументов.



```
lab09-2.asm
~/work/study/2022-2023/Архитектура ...pa/study_2022-2023_arh-pc/labs/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 printf ; вызываем функцию печати
    loop next ; переход к обработке следующего
             ; аргумента (переход на метку `next`)
_end:
    call quit
```

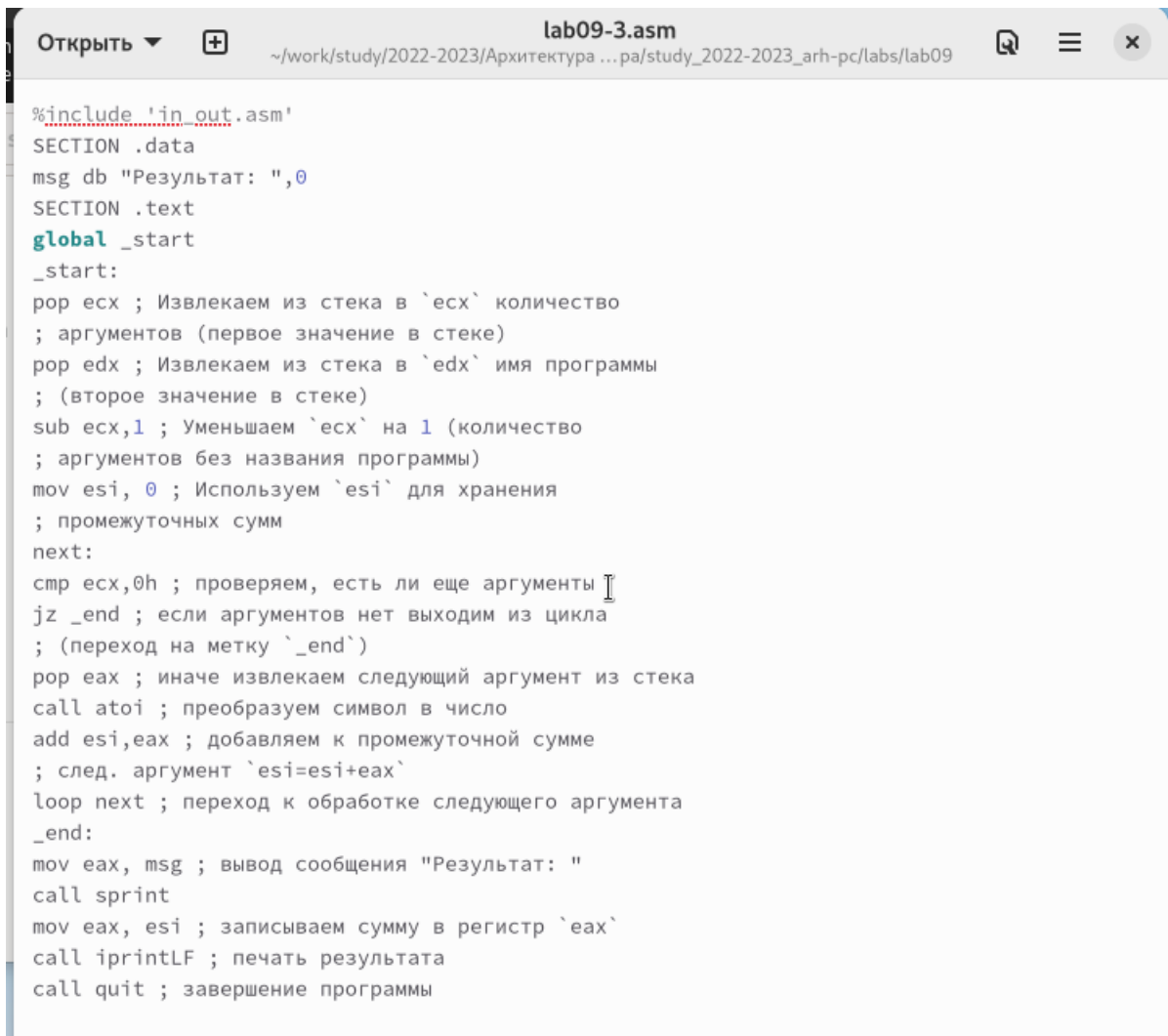
Рис. 2.7: Файл lab9-2.asm



```
[odedzhimi@fedora lab09]$ nasm -f elf lab09-2.asm
[odedzhimi@fedora lab09]$ ld -m elf_i386 -o lab09-2 lab09-2.o
[odedzhimi@fedora lab09]$ ./lab09-2
[odedzhimi@fedora lab09]$ ./lab09-2 argument 1 argument 2 "argument 3"
argument
1
argument
2
argument 3
[odedzhimi@fedora lab09]$
```

Рис. 2.8: Работа программы lab9-2.asm

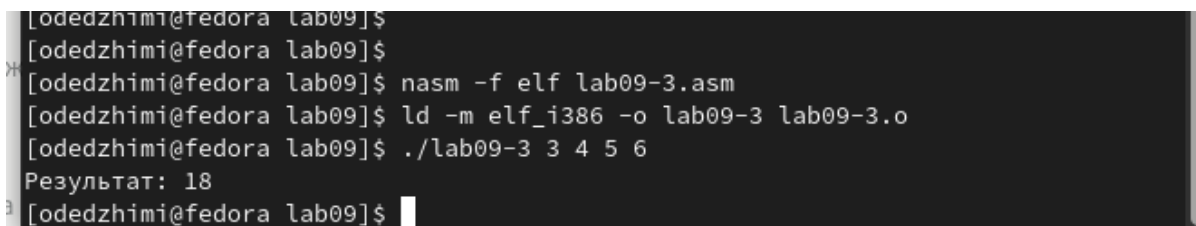
6. Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. (рис. [2.9], [2.10])



```
Открыть + lab09-3.asm ~/work/study/2022-2023/Архитектура ...pa/study_2022-2023_arh-pc/labs/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`
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы
```

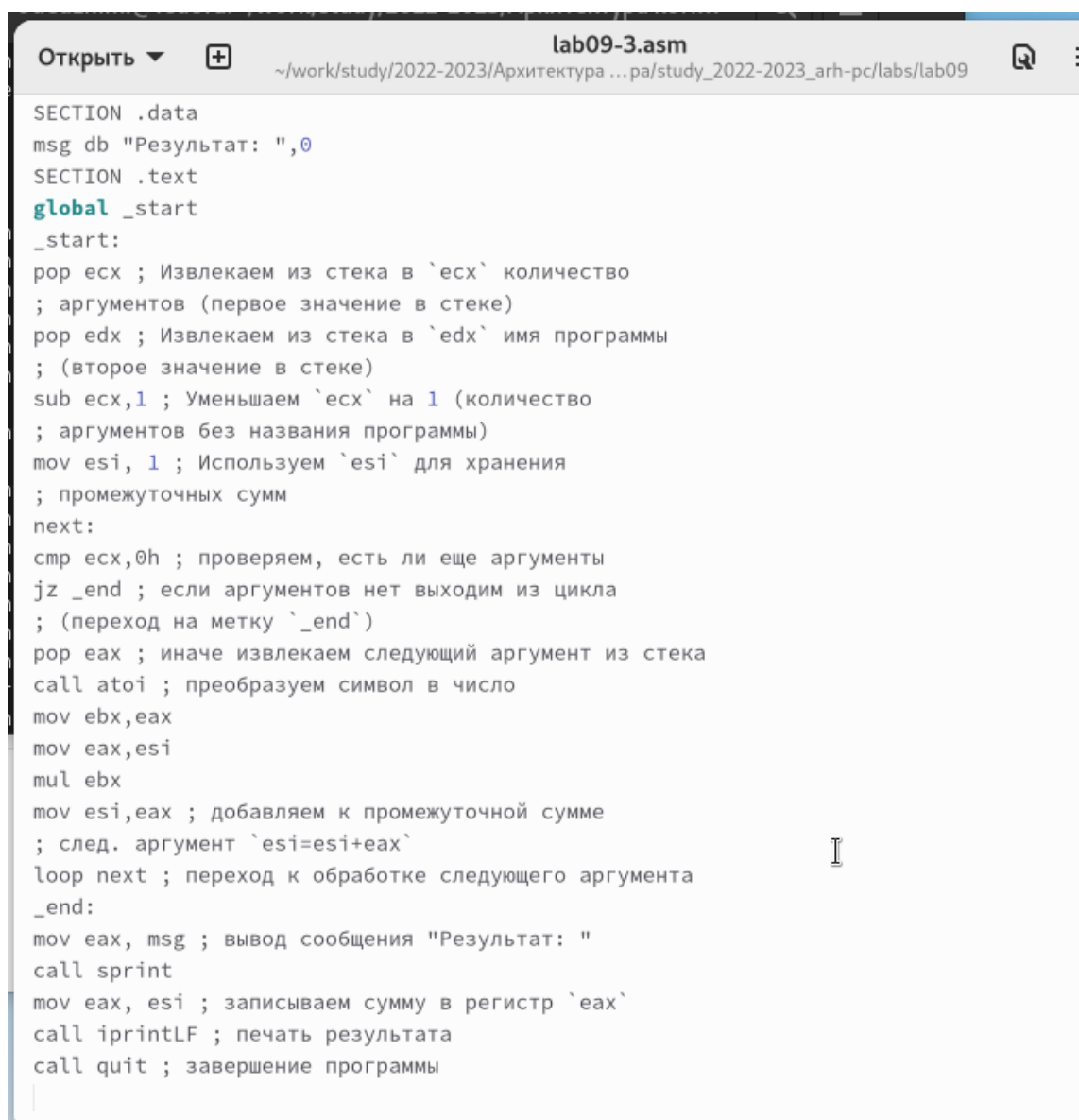
Рис. 2.9: Файл lab9-3.asm



```
[odedzhimi@fedora lab09]$
[odedzhimi@fedora lab09]$
[odedzhimi@fedora lab09]$ nasm -f elf lab09-3.asm
[odedzhimi@fedora lab09]$ ld -m elf_i386 -o lab09-3 lab09-3.o
[odedzhimi@fedora lab09]$ ./lab09-3 3 4 5 6
Результат: 18
[odedzhimi@fedora lab09]$
```

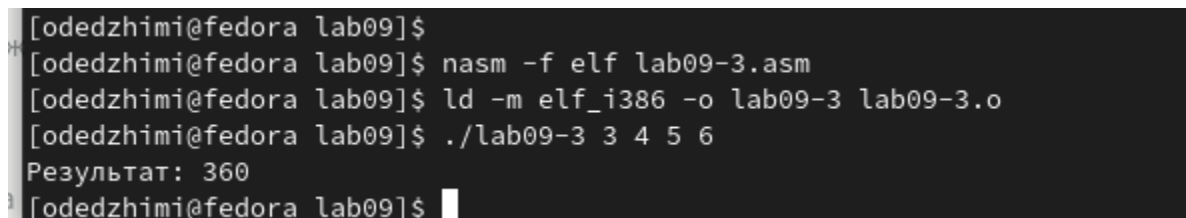
Рис. 2.10: Работа программы lab9-3.asm

7. Измените текст программы из листинга 9.3 для вычисления произведения аргументов командной строки. (рис. [2.11], [2.12])



```
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 ebx,eax
mov eax,esi
mul ebx
mov esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы
```

Рис. 2.11: Файл lab9-3.asm



```
[odedzhimi@fedora lab09]$
[odedzhimi@fedora lab09]$ nasm -f elf lab09-3.asm
[odedzhimi@fedora lab09]$ ld -m elf_i386 -o lab09-3 lab09-3.o
[odedzhimi@fedora lab09]$ ./lab09-3 3 4 5 6
Результат: 360
[odedzhimi@fedora lab09]$
```

Рис. 2.12: Работа программы lab9-3.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 9.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x . (рис. [2.13], [2.14])

для варианта 1 $f(x) = 2x + 15$



```
lab09-4.asm
~/work/study/2022-2023/Архитектура ...pa/study_2

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)=2x+15 ',0

SECTION .text
global _start
_start:
mov eax, fx
call sprintLF
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
mov ebx,2
mul ebx
add eax,15
add esi,eax

loop next

_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
```

Рис. 2.13: Файл lab9-4.asm

```
[odedzhimi@fedora lab09]$  
[odedzhimi@fedora lab09]$  
[odedzhimi@fedora lab09]$ nasm -f elf lab09-4.asm  
[odedzhimi@fedora lab09]$ ld -m elf_i386 -o lab09-4 lab09-4.o  
[odedzhimi@fedora lab09]$ ./lab09-4 1  
f(x)=2x+15  
Результат: 17  
[odedzhimi@fedora lab09]$ ./lab09-4 1 2 3 4 5 6  
f(x)=2x+15  
Результат: 132  
[odedzhimi@fedora lab09]$
```

Рис. 2.14: Работа программы lab9-4.asm

3 Выводы

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