

# **Отчёт по лабораторной работе 8**

**Программирование цикла. Обработка аргументов командной строки.**

Субанов Юсуф Жура угли НПМбв-01-21

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Выводы</b>	<b>19</b>

## Список иллюстраций

2.1	Файл lab8-1.asm . . . . .	7
2.2	Работа программы lab8-1.asm . . . . .	8
2.3	Файл lab8-1.asm . . . . .	9
2.4	Работа программы lab8-1.asm . . . . .	10
2.5	Файл lab8-1.asm . . . . .	11
2.6	Работа программы lab8-1.asm . . . . .	12
2.7	Файл lab8-2.asm . . . . .	12
2.8	Работа программы lab8-2.asm . . . . .	13
2.9	Файл lab8-3.asm . . . . .	14
2.10	Работа программы lab8-3.asm . . . . .	14
2.11	Файл lab8-3.asm . . . . .	15
2.12	Работа программы lab8-3.asm . . . . .	16
2.13	Файл lab8-4.asm . . . . .	17
2.14	Работа программы lab8-4.asm . . . . .	18

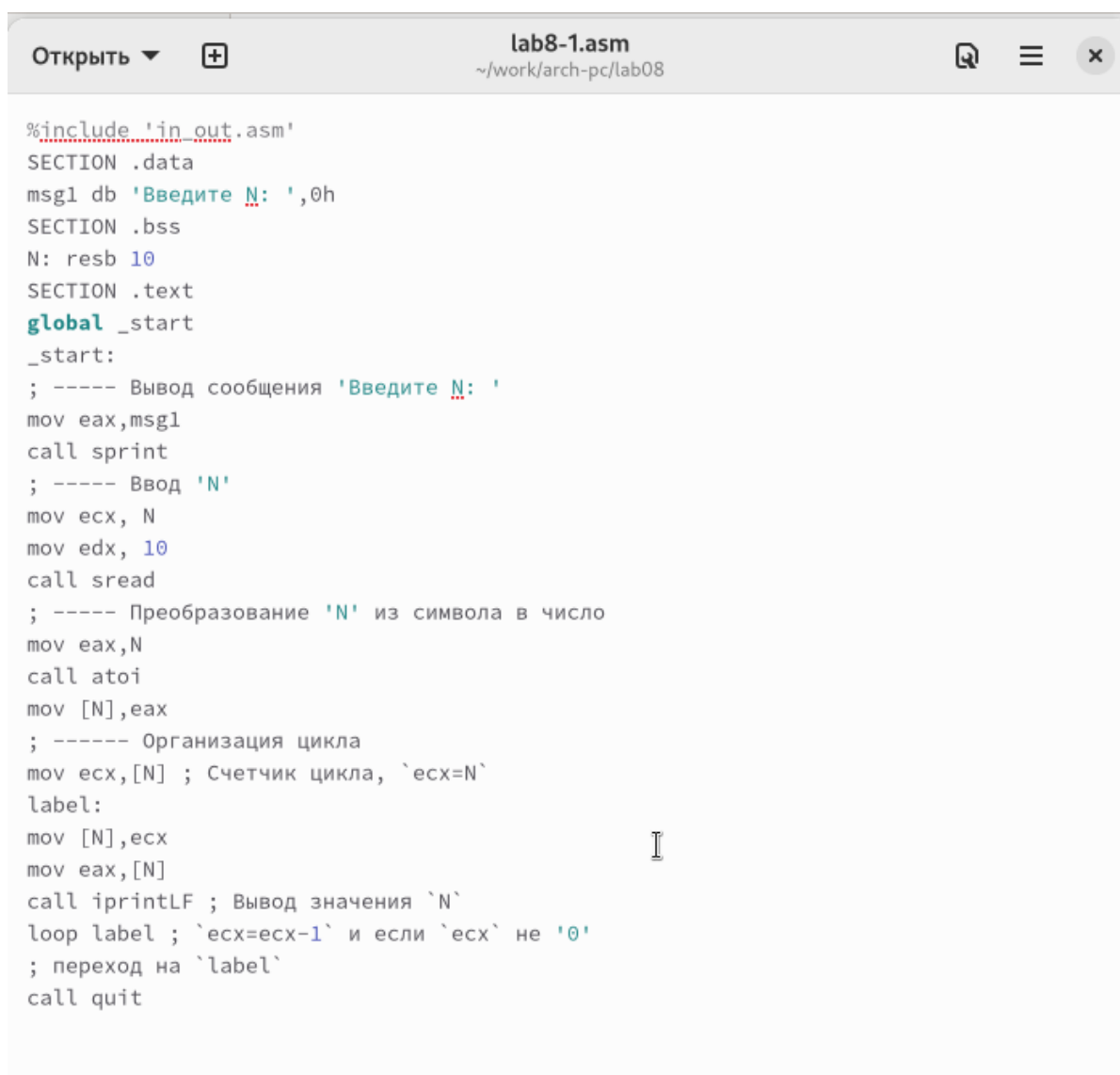
## **Список таблиц**

# 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 № 8, перейдите в него и создайте файл lab8-1.asm
2. Введите в файл lab8-1.asm текст программы из листинга 8.1. Создайте исполняемый файл и проверьте его работу.



```
%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 lab8-1.asm

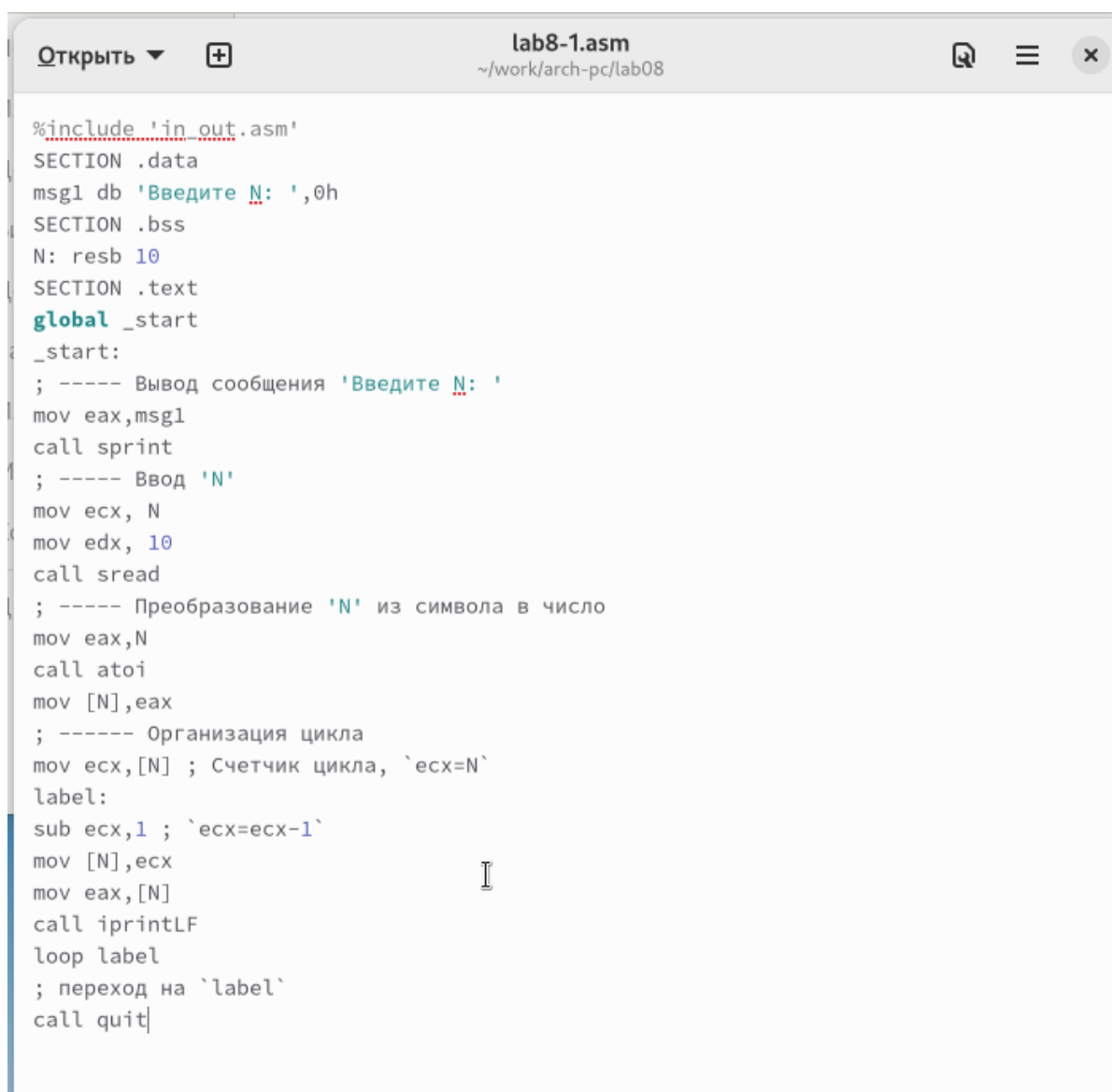
```
[yusufsubanov@fedora lab08]$ nasm -f elf lab8-1.asm
[yusufsubanov@fedora lab08]$ ld -m elf_i386 lab8-1.o -o lab8-1
[yusufsubanov@fedora lab08]$ ./lab8-1
Введите N: 4
4
3
2
1
[yusufsubanov@fedora lab08]$
```

Рис. 2.2: Работа программы lab8-1.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?

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





```
%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
```

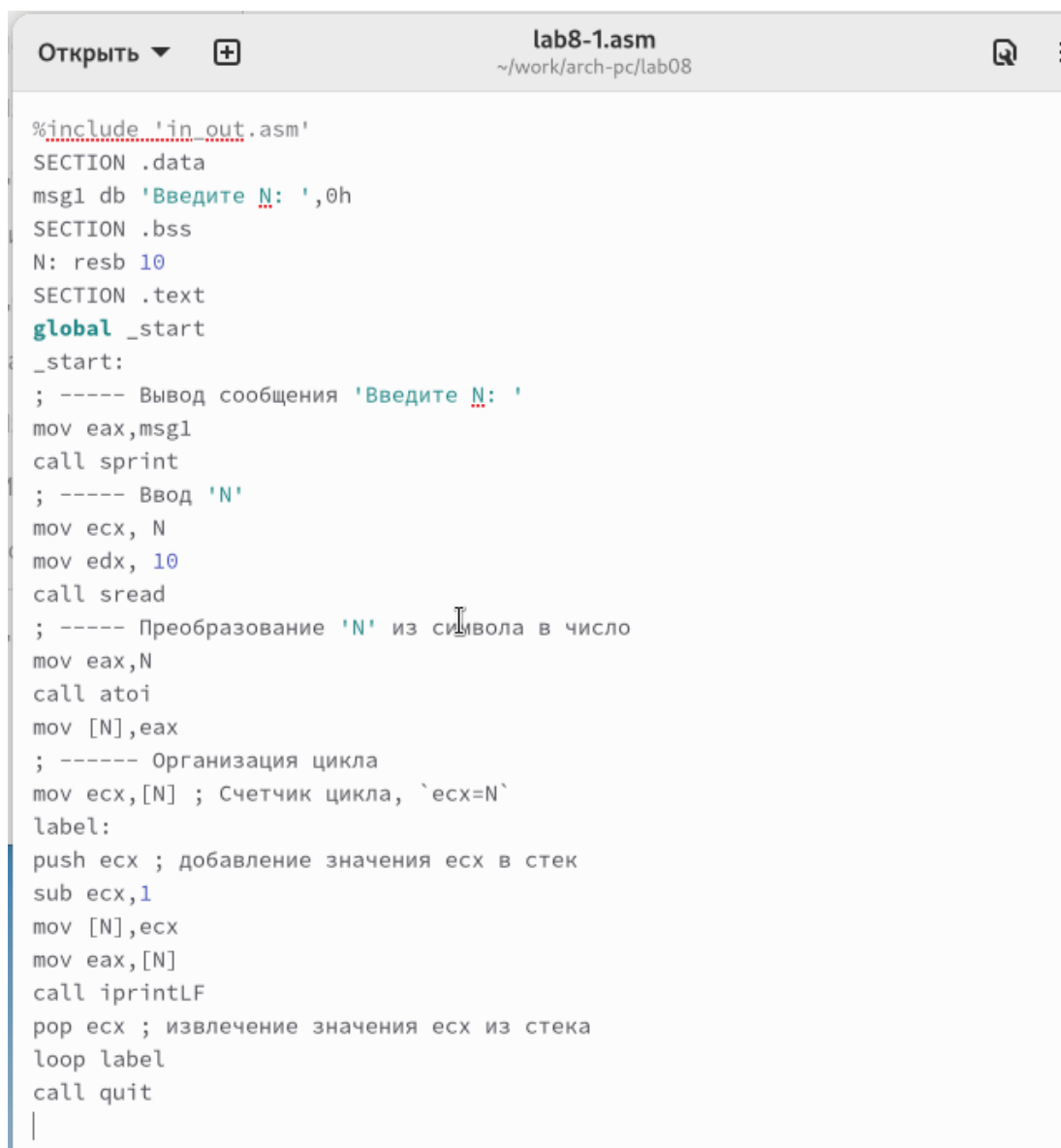
Рис. 2.3: Файл lab8-1.asm

```
[yusufsubanov@fedora lab08]$ nasm -f elf lab8-1.asm
[yusufsubanov@fedora lab08]$ ld -m elf_i386 lab8-1.o -o lab8-1
[yusufsubanov@fedora lab08]$ ./lab8-1
Введите N: 4
3
1
[yusufsubanov@fedora lab08]$
```

Рис. 2.4: Работа программы lab8-1.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?

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



```
%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 lab8-1.asm

```

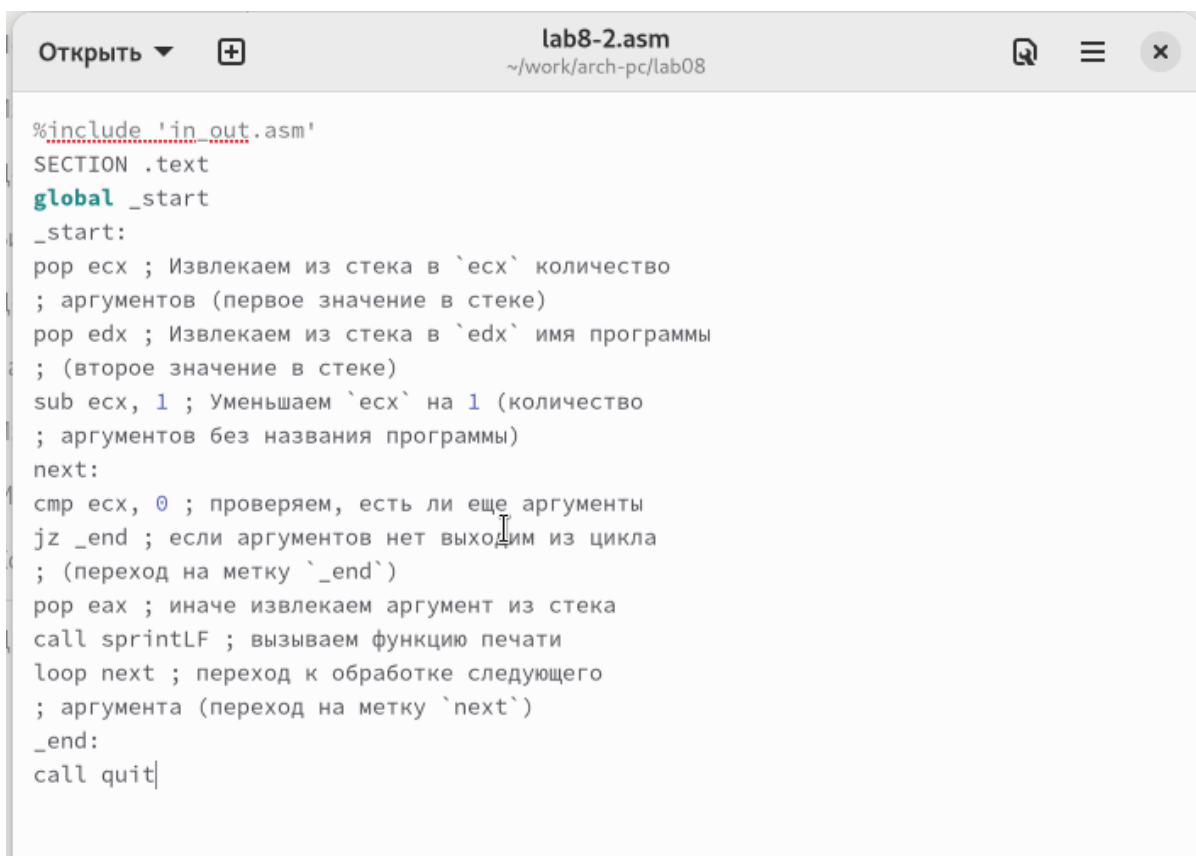
[yusufsubanov@fedora lab08]$ nasm -f elf lab8-1.asm
[yusufsubanov@fedora lab08]$ ld -m elf_i386 lab8-1.o -o lab8-1
[yusufsubanov@fedora lab08]$ ./lab8-1
Введите N: 4
3
2
1
0
[yusufsubanov@fedora lab08]$

```

Рис. 2.6: Работа программы lab8-1.asm

5. Создайте файл lab9-2.asm в каталоге ~/work/arch-pc/lab08 и введите в него текст программы из листинга 8.2. Создайте исполняемый файл и запустите его, указав аргументы. Сколько аргументов было обработано программой?

Программа обработала 5 аргументов.



```

lab8-2.asm
~/work/arch-pc/lab08

%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

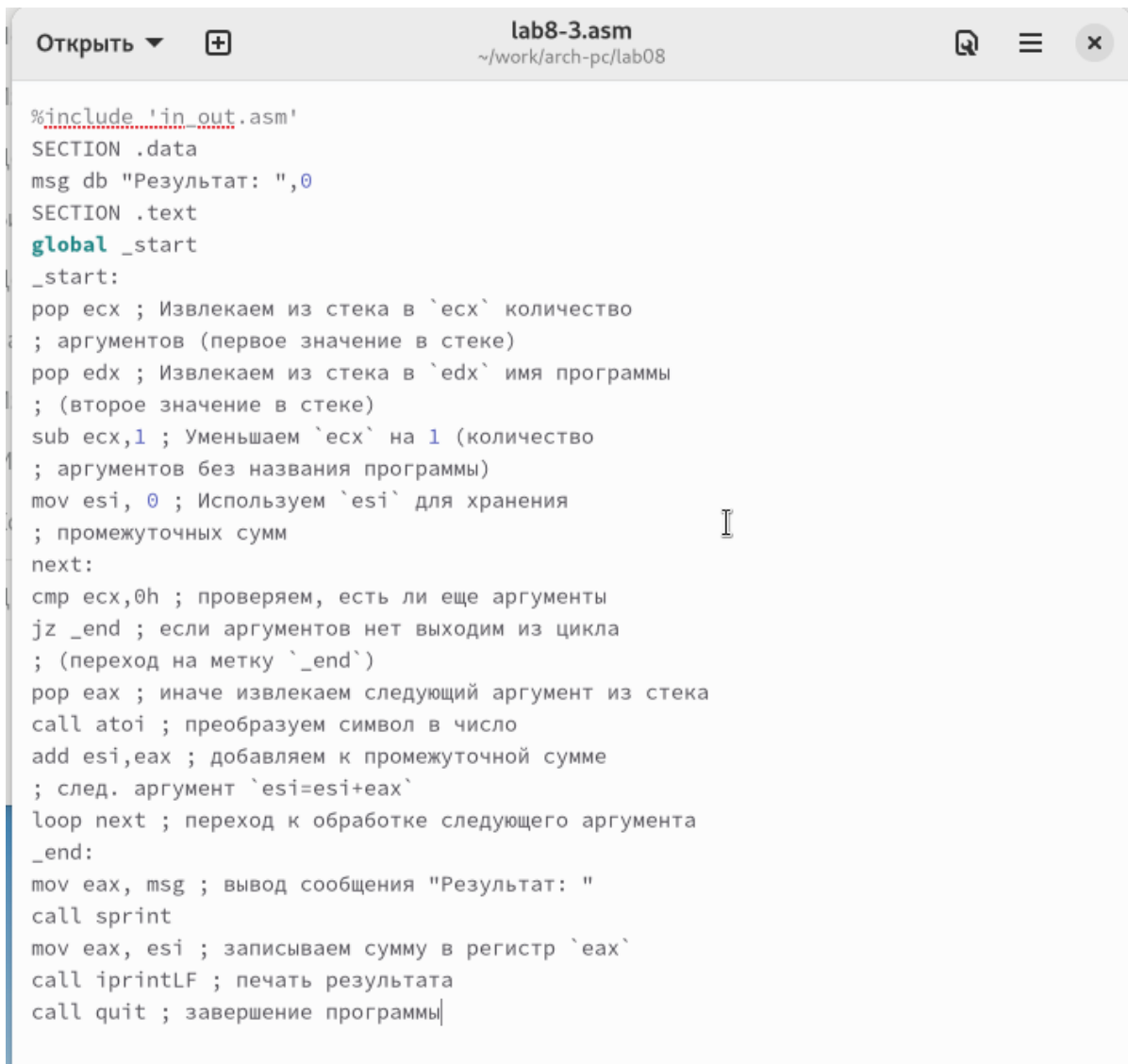
```

Рис. 2.7: Файл lab8-2.asm

```
[yusufsubanov@fedora lab08]$  
[yusufsubanov@fedora lab08]$ nasm -f elf lab8-2.asm  
[yusufsubanov@fedora lab08]$ ld -m elf_i386 lab8-2.o -o lab8-2  
[yusufsubanov@fedora lab08]$ ./lab8-2 argument 1 argiment 2 'argument 3'  
argument  
1  
argiment  
2  
argument 3  
[yusufsubanov@fedora lab08]$
```

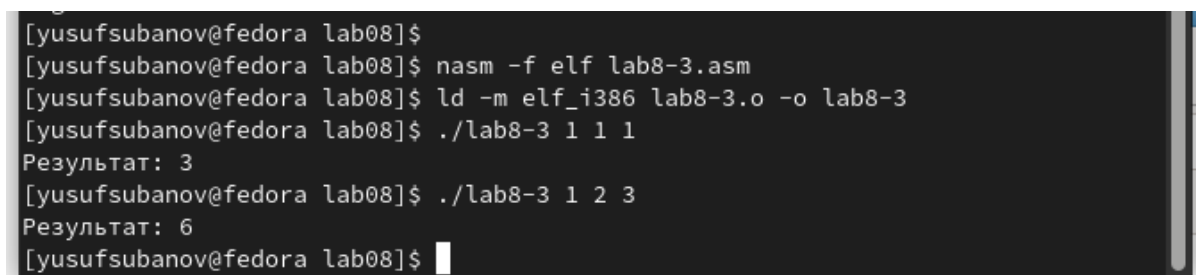
Рис. 2.8: Работа программы lab8-2.asm

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



```
%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
```

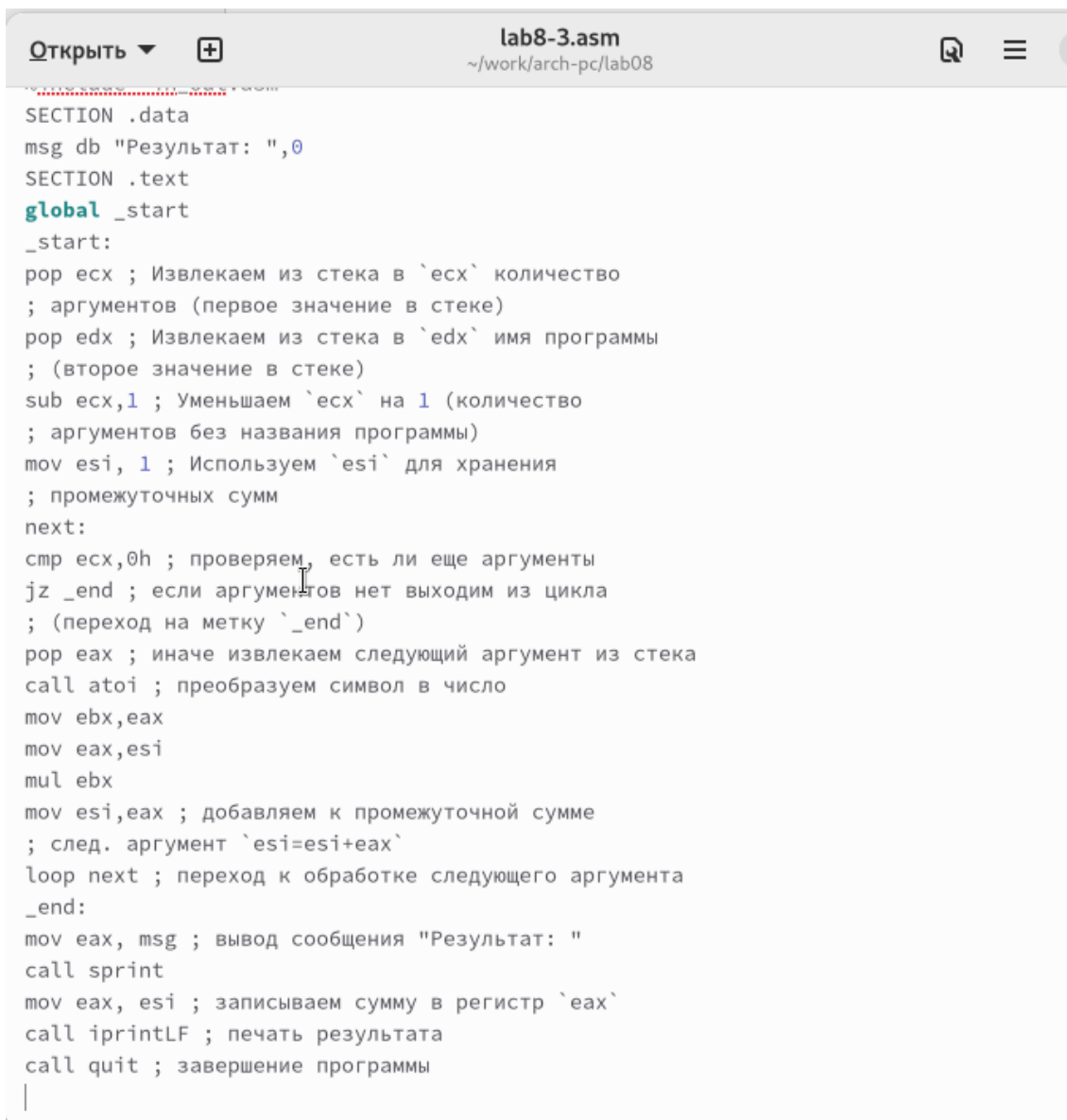
Рис. 2.9: Файл lab8-3.asm



```
[yusufsubanov@fedora lab08]$
[yusufsubanov@fedora lab08]$ nasm -f elf lab8-3.asm
[yusufsubanov@fedora lab08]$ ld -m elf_i386 lab8-3.o -o lab8-3
[yusufsubanov@fedora lab08]$ ./lab8-3 1 1 1
Результат: 3
[yusufsubanov@fedora lab08]$ ./lab8-3 1 2 3
Результат: 6
[yusufsubanov@fedora lab08]$
```

Рис. 2.10: Работа программы lab8-3.asm

7. Измените текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.



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

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 lab8-3.asm

```
[yusufsubanov@fedora lab08]$ nasm -f elf lab8-3.asm
[yusufsubanov@fedora lab08]$ ld -m elf_i386 lab8-3.o -o lab8-3
[yusufsubanov@fedora lab08]$ ./lab8-3 1 1 1
Результат: 3
[yusufsubanov@fedora lab08]$ ./lab8-3 1 2 3
Результат: 6
[yusufsubanov@fedora lab08]$ ld -m elf_i386 lab8-3.o -o lab8-3
[yusufsubanov@fedora lab08]$ nasm -f elf lab8-3.asm
[yusufsubanov@fedora lab08]$ ld -m elf_i386 lab8-3.o -o lab8-3
[yusufsubanov@fedora lab08]$ ./lab8-3 1 1 1
Результат: 1
[yusufsubanov@fedora lab08]$ ./lab8-3 1 2 3
Результат: 6
[yusufsubanov@fedora lab08]$ ./lab8-3 1 2 3 4
Результат: 24
[yusufsubanov@fedora lab08]$
```

Рис. 2.12: Работа 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 1  $f(x) = 2x + 15$



```
Открыть ▾ + lab8-4.asm  
~/work/arch-pc/lab08  
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  
call quit
```

Рис. 2.13: Файл lab8-4.asm

```
[yusufsubanov@fedora lab08]$  
[yusufsubanov@fedora lab08]$ nasm -f elf lab8-4.asm  
[yusufsubanov@fedora lab08]$ ld -m elf_i386 lab8-4.o -o lab8-4  
[yusufsubanov@fedora lab08]$ ./lab8-4 1  
f(x)=2x+15  
Результат: 17  
[yusufsubanov@fedora lab08]$ ./lab8-4 1 2 3 4 5  
f(x)=2x+15  
Результат: 105  
[yusufsubanov@fedora lab08]$
```

Рис. 2.14: Работа программы lab8-4.asm

## 3 Выводы

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