

# **Отчёта по лабораторной работе 9**

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

Элина Майзингер НММбд-02-22

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>21</b>
	<b>Список литературы</b>	<b>22</b>

## Список иллюстраций

4.1	Файл lab9-1.asm . . . . .	9
4.2	Работа программы lab9-1.asm . . . . .	10
4.3	Файл lab9-1.asm . . . . .	11
4.4	Работа программы lab9-1.asm . . . . .	12
4.5	Файл lab9-1.asm . . . . .	13
4.6	Работа программы lab9-1.asm . . . . .	14
4.7	Файл lab9-2.asm . . . . .	15
4.8	Работа программы lab9-2.asm . . . . .	15
4.9	Файл lab9-3.asm . . . . .	16
4.10	Работа программы lab9-3.asm . . . . .	17
4.11	Файл lab9-3.asm . . . . .	18
4.12	Работа программы lab9-3.asm . . . . .	18
4.13	Файл lab9-4.asm . . . . .	19
4.14	Работа программы lab9-4.asm . . . . .	20

## Список таблиц

# 1 Цель работы

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

## 2 Задание

1. Изучите примеры программ
2. 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  $x$ .
3. Загрузите файлы на GitHub.

### 3 Теоретическое введение

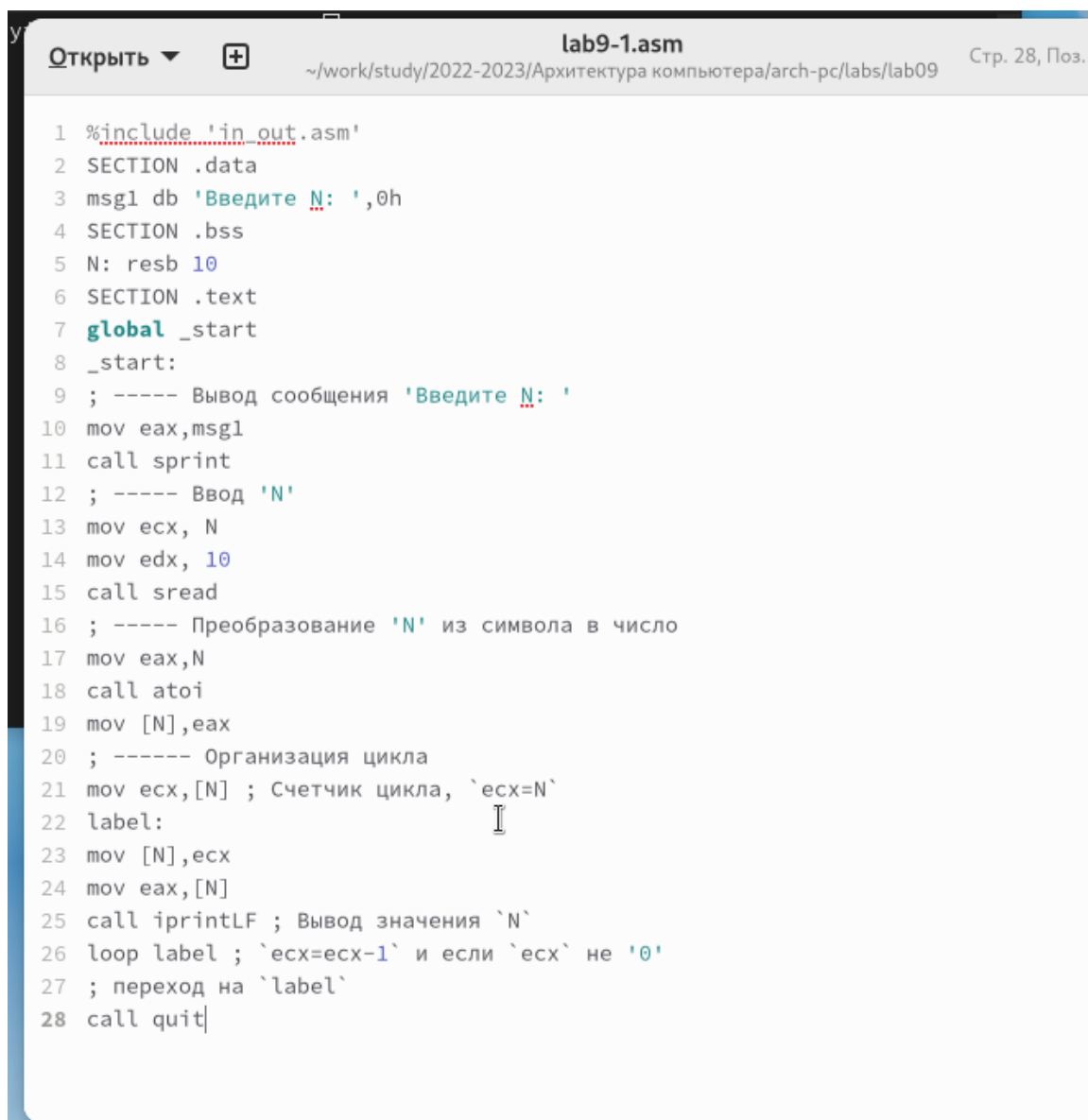
Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре esx. Наиболее простой является инструкция loor. Инструкция loor выполняется в два этапа. Сначала из регистра esx вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды loor.

## 4 Выполнение лабораторной работы

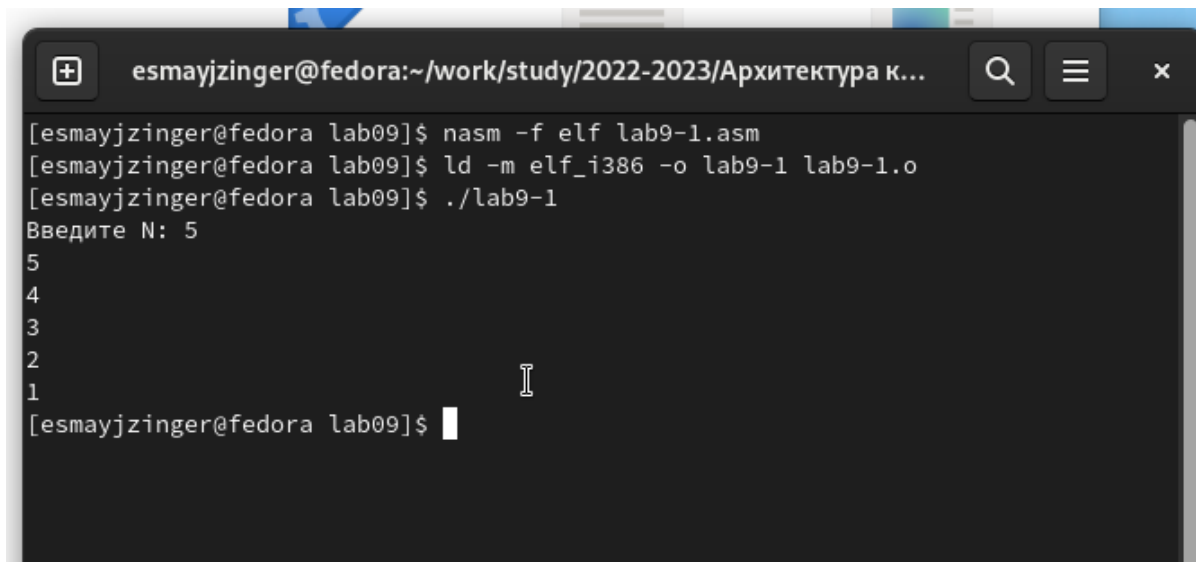
1. Создайте каталог для программ лабораторной работы № 9, перейдите в него и создайте файл lab9-1.asm
2. Введите в файл lab9-1.asm текст программы из листинга 9.1. Создайте исполняемый файл и проверьте его работу. (рис. 4.1, 4.2)





```
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. 4.1: Файл lab9-1.asm

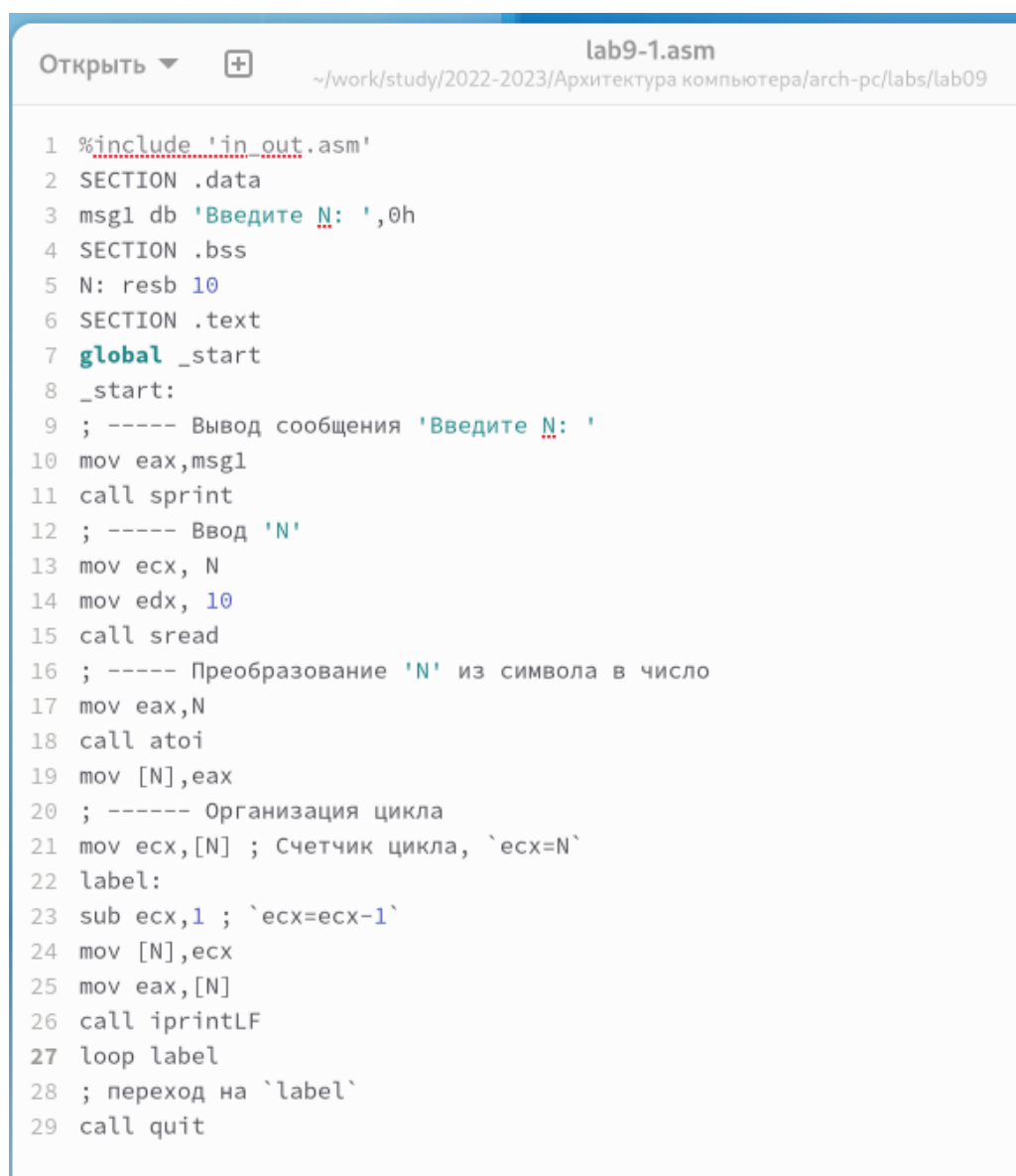


```
esmayjzinger@fedora:~/work/study/2022-2023/Архитектура к...
[esmayjzinger@fedora lab09]$ nasm -f elf lab9-1.asm
[esmayjzinger@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[esmayjzinger@fedora lab09]$ ./lab9-1
Введите N: 5
5
4
3
2
1
[esmayjzinger@fedora lab09]$
```

Рис. 4.2: Работа программы lab9-1.asm

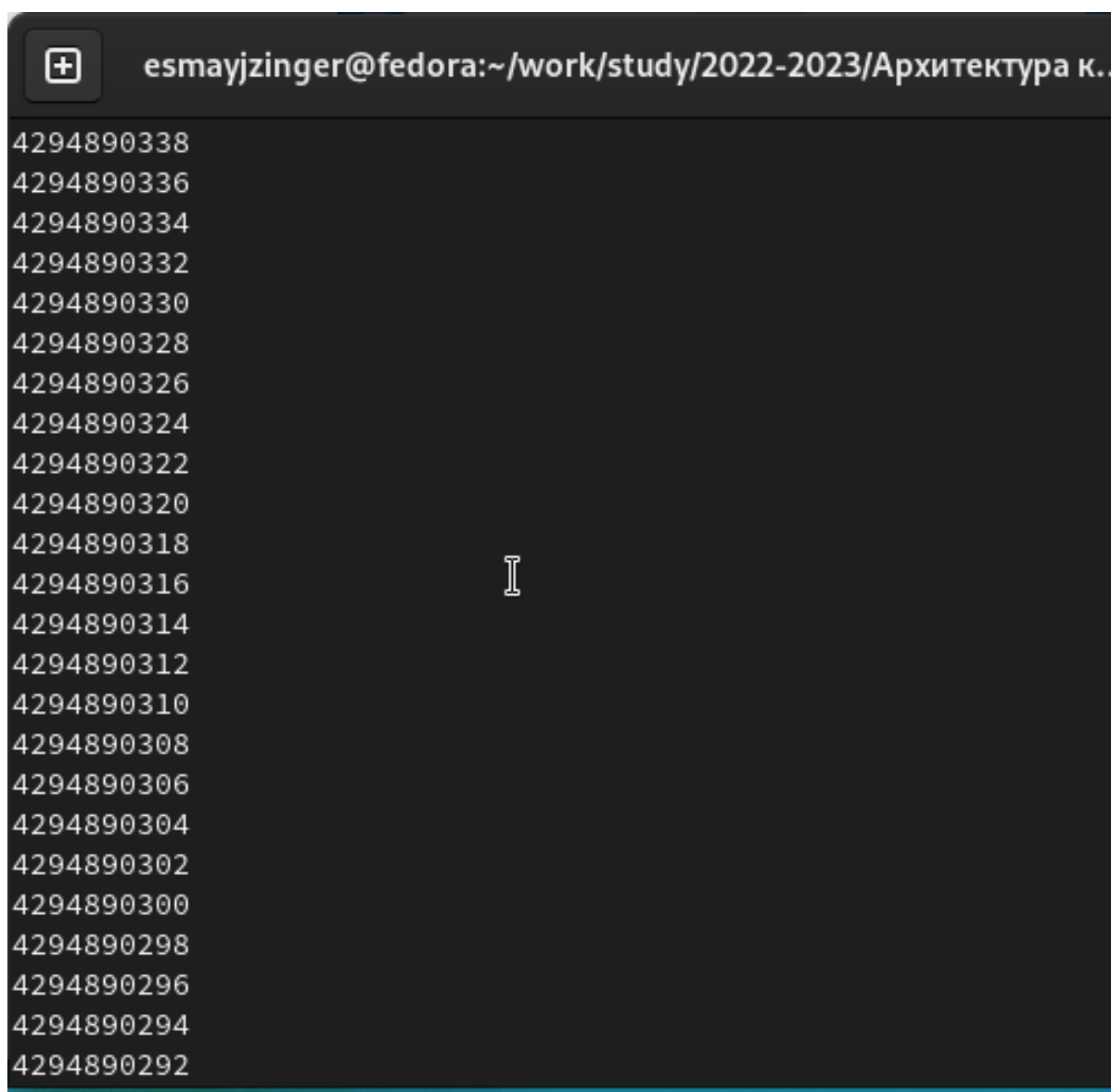
3. Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Измените текст программы добавив изменение значение регистра `ecx` в цикле: Создайте исполняемый файл и проверьте его работу. Какие значения принимает регистр `ecx` в цикле? Соответствует ли число проходов цикла значению `N`, введенному с клавиатуры? (рис. 4.3, 4.4)

Программа запускает бесконечный цикл.



```
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. 4.3: Файл lab9-1.asm




```
esmayjzinger@fedora:~/work/study/2022-2023/Архитектура к...
4294890338
4294890336
4294890334
4294890332
4294890330
4294890328
4294890326
4294890324
4294890322
4294890320
4294890318
4294890316
4294890314
4294890312
4294890310
4294890308
4294890306
4294890304
4294890302
4294890300
4294890298
4294890296
4294890294
4294890292
```

Рис. 4.4: 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. 4.5, 4.6)

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

Открыть ▾  lab9-1.asm Стр. 21

~\work\study\2022-2023\Архитектура компьютера\arch-pc\labs\lab09

```
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. 4.5: Файл lab9-1.asm

```
[esmayjzinger@fedora lab09]$  
[esmayjzinger@fedora lab09]$ nasm -f elf lab9-1.asm  
[esmayjzinger@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o  
[esmayjzinger@fedora lab09]$ ./lab9-1  
Введите N: 5  
4  
3  
2  
1  
0  
[esmayjzinger@fedora lab09]$
```

Рис. 4.6: 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. 4.7, 4.8) Сколько аргументов было обработано программой?

Программа обработала 5 аргументов.

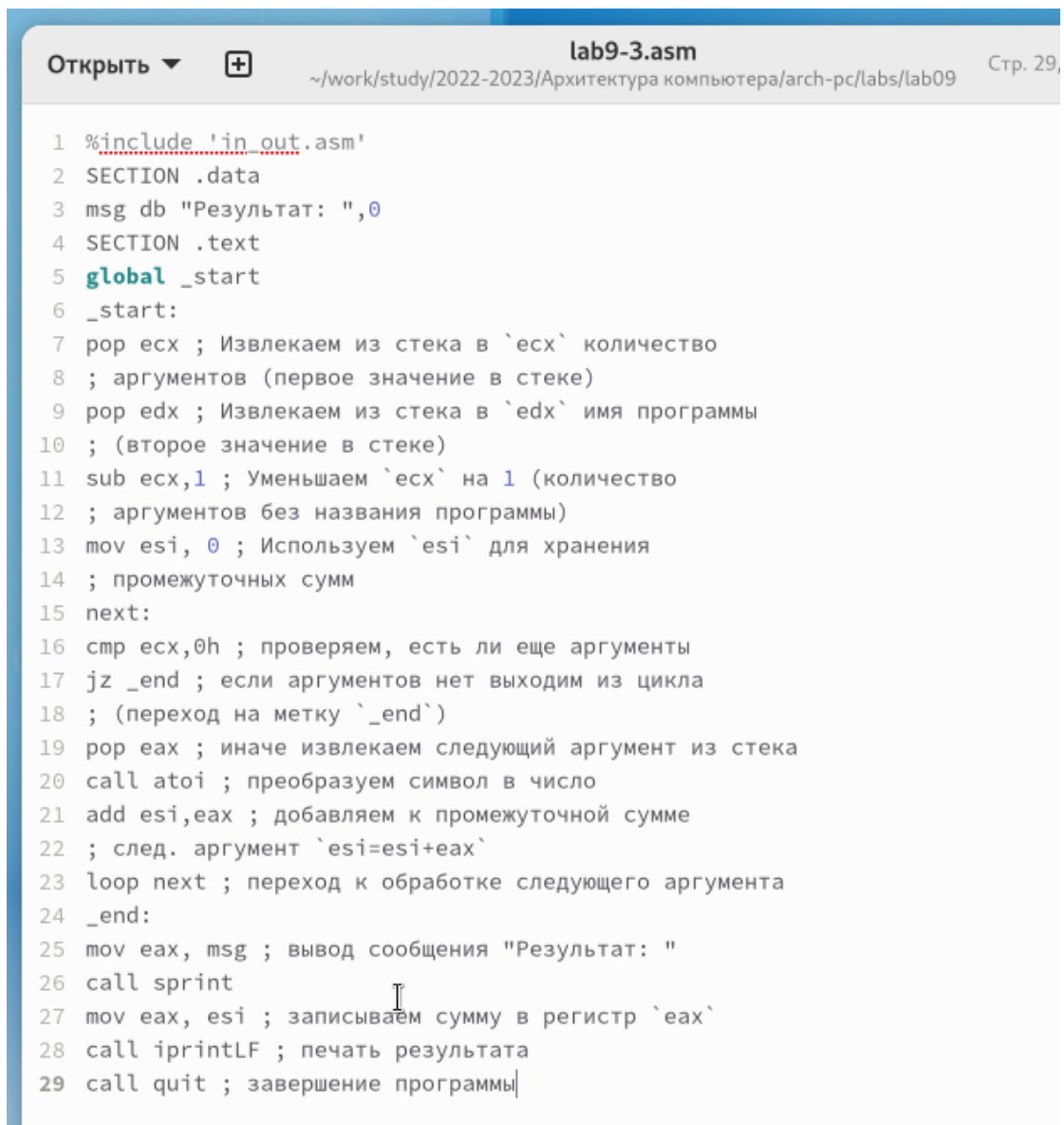
```
1 %include 'in_out.asm'
2 SECTION .text
3 global _start
4 _start:
5 pop ecx ; Извлекаем из стека в `ecx` количество
6 ; аргументов (первое значение в стеке)
7 pop edx ; Извлекаем из стека в `edx` имя программы
8 ; (второе значение в стеке)
9 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
10 ; аргументов 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. 4.7: Файл lab9-2.asm

```
[esmayjzinger@fedora lab09]$ nasm -f elf lab9-2.asm
[esmayjzinger@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[esmayjzinger@fedora lab09]$ ./lab9-2
[esmayjzinger@fedora lab09]$ ./lab9-2 argument 1 argument 2 'argument 3'
argument
1
argument
2
argument 3
[esmayjzinger@fedora lab09]$
```

Рис. 4.8: Работа программы lab9-2.asm

6. Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. (рис. 4.9, 4.10)



```
Открыть ▾ + lab9-3.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab09 Стр. 29,

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
```

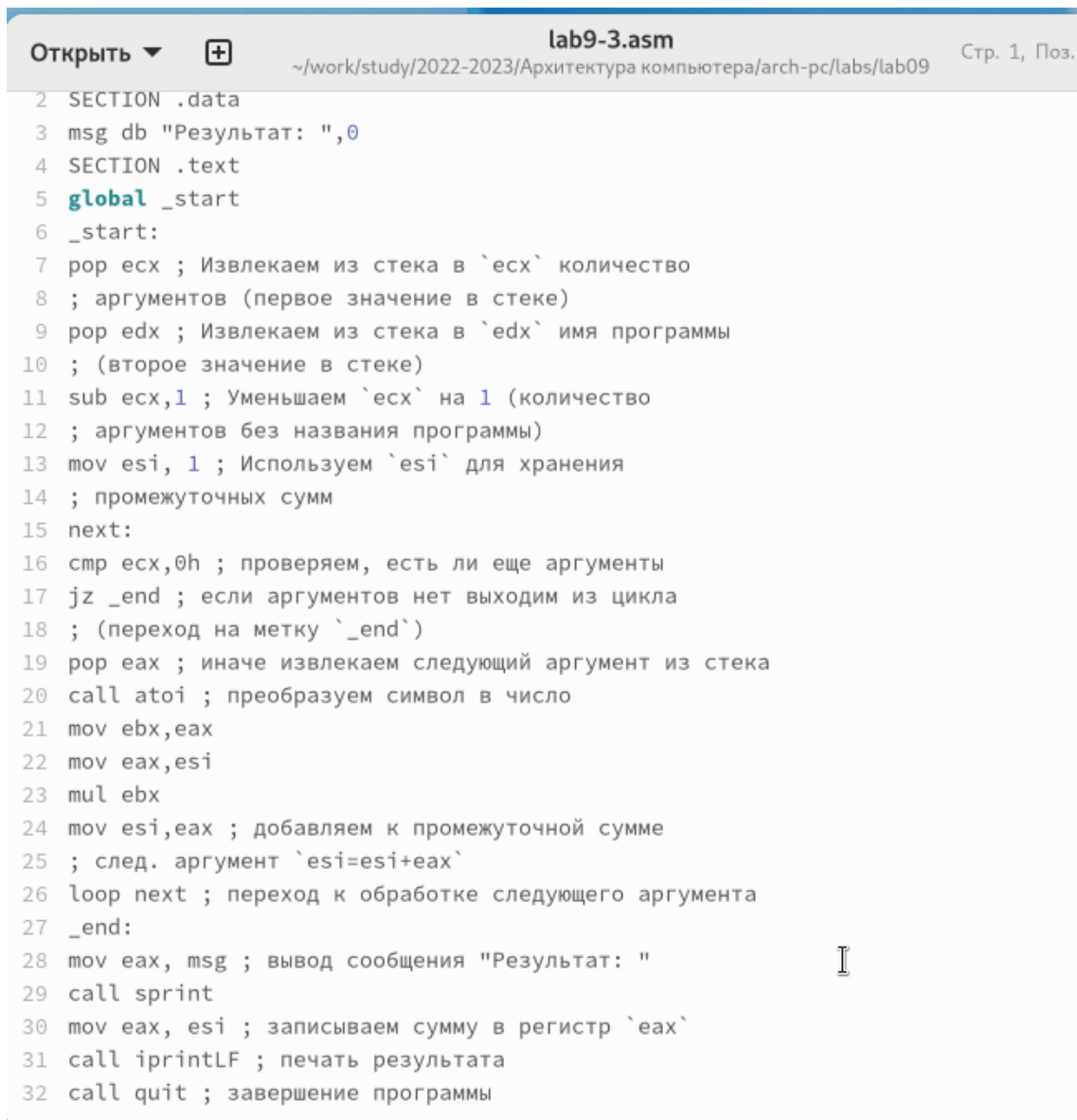
Рис. 4.9: Файл lab9-3.asm



```
argument 3
[esmayjzinger@fedora lab09]$
[esmayjzinger@fedora lab09]$
[esmayjzinger@fedora lab09]$ nasm -f elf lab9-3.asm
[esmayjzinger@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[esmayjzinger@fedora lab09]$ ./lab9-3 1 2 3 4 5
Результат: 15
[esmayjzinger@fedora lab09]$
```

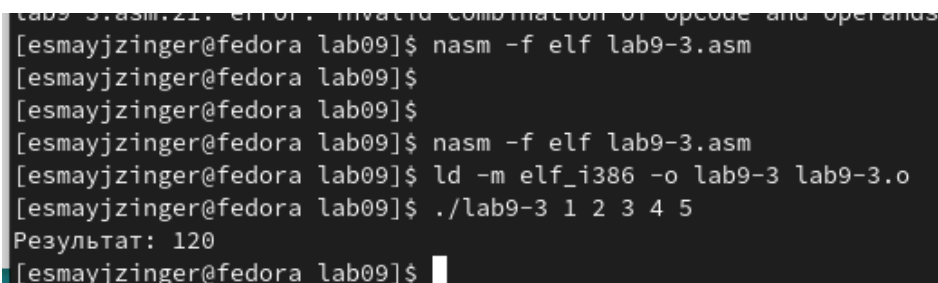
Рис. 4.10: Работа программы lab9-3.asm

7. Измените текст программы из листинга 9.3 для вычисления произведения аргументов командной строки. (рис. 4.11, 4.12)



```
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. 4.11: Файл lab9-3.asm

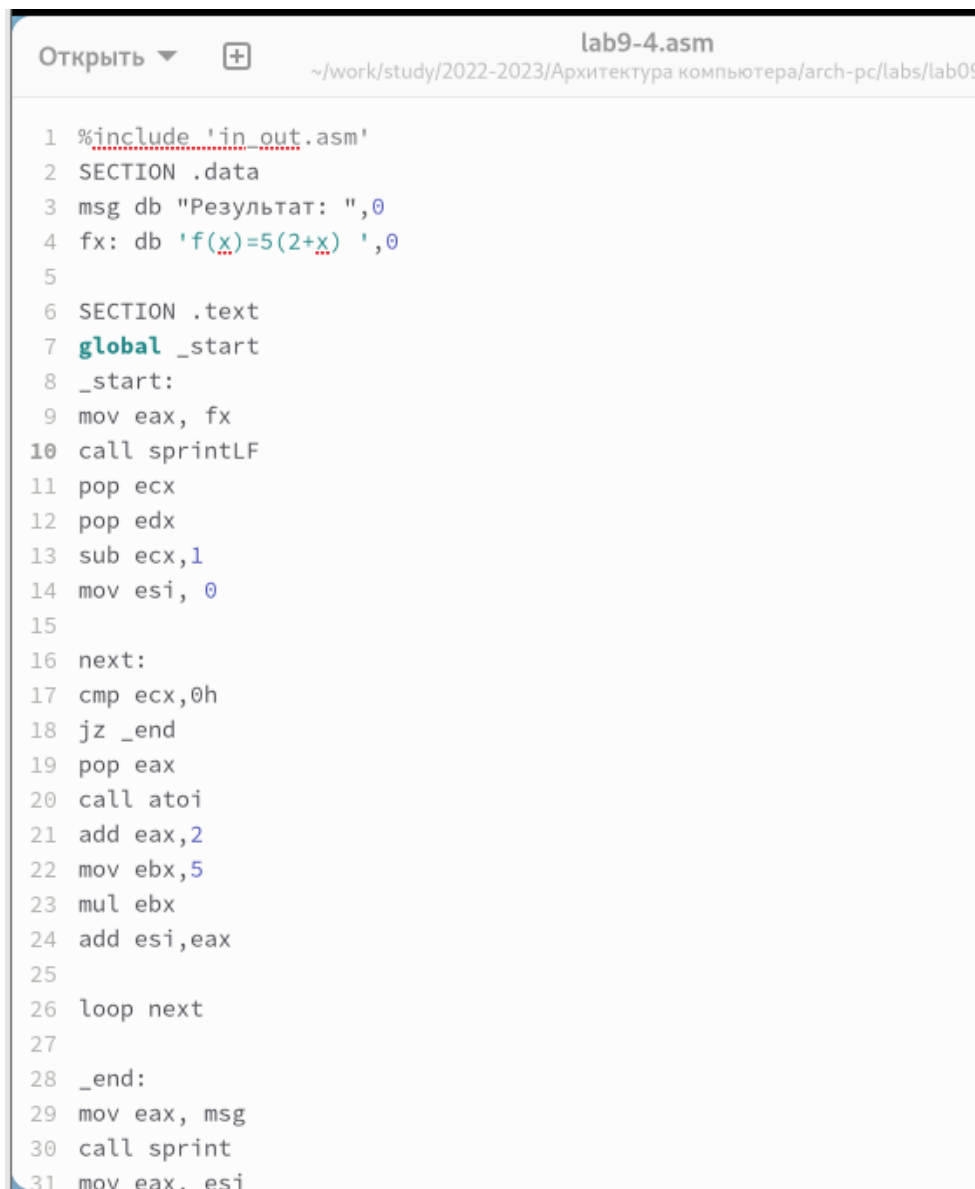


```
lab9-3.asm.21: error: invalid combination of opcode and operands
[esmayjzinger@fedora lab09]$ nasm -f elf lab9-3.asm
[esmayjzinger@fedora lab09]$
[esmayjzinger@fedora lab09]$
[esmayjzinger@fedora lab09]$ nasm -f elf lab9-3.asm
[esmayjzinger@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[esmayjzinger@fedora lab09]$ ./lab9-3 1 2 3 4 5
Результат: 120
[esmayjzinger@fedora lab09]$
```

Рис. 4.12: 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. 4.13, 4.14)

для варианта 10  $f(x) = 5(2+x)$



```
lab9-4.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab09

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

Рис. 4.13: Файл lab9-4.asm

```
[esmayjzinger@fedora lab09]$  
[esmayjzinger@fedora lab09]$ nasm -f elf lab9-4.asm  
[esmayjzinger@fedora lab09]$ ld -m elf_i386 -o lab9-4 lab9-4.o  
[esmayjzinger@fedora lab09]$ ./lab9-4 1  
f(x)=5(2+x) Результат: 15  
[esmayjzinger@fedora lab09]$ nasm -f elf lab9-4.asm  
[esmayjzinger@fedora lab09]$ ld -m elf_i386 -o lab9-4 lab9-4.o  
[esmayjzinger@fedora lab09]$ ./lab9-4 1 2 3 4 5  
f(x)=5(2+x)  
Результат: 125  
[esmayjzinger@fedora lab09]$
```

Рис. 4.14: Работа программы lab9-4.asm

## 5 Выводы

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

# Список литературы

1. Расширенный ассемблер: NASM
2. MASM, TASM, FASM, NASM под Windows и Linux