

Отчёт по лабораторной работе 9

Программирование цикла. Обработка аргументов командной строки.

Чесноков Артемий Павлович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение самостоятельной работы	20
4	Выводы	23

Список иллюстраций

2.1	Созадние файла lab9-1.asm	7
2.2	Листинг lab9-1.asm	8
2.3	Компиляция и проверка lab9-1.asm	9
2.4	Измененный lab9-1.asm	10
2.5	Компилирование и проверка измененного lab9-1.asm	11
2.6	Код lab9-1.asm с push и pop	12
2.7	Проверка lab9-1.asm с push и pop	13
2.8	Код lab9-2.asm	14
2.9	Проверка lab9-2.asm	15
2.10	Код lab9-3.asm	16
2.11	Проверка lab9-3.asm	17
2.12	Измененный код lab9-3.asm	18
2.13	Проверка измененного кода lab9-3.asm	19
3.1	Код программы для самостоятельной	21
3.2	Проверка самостоятельной	22

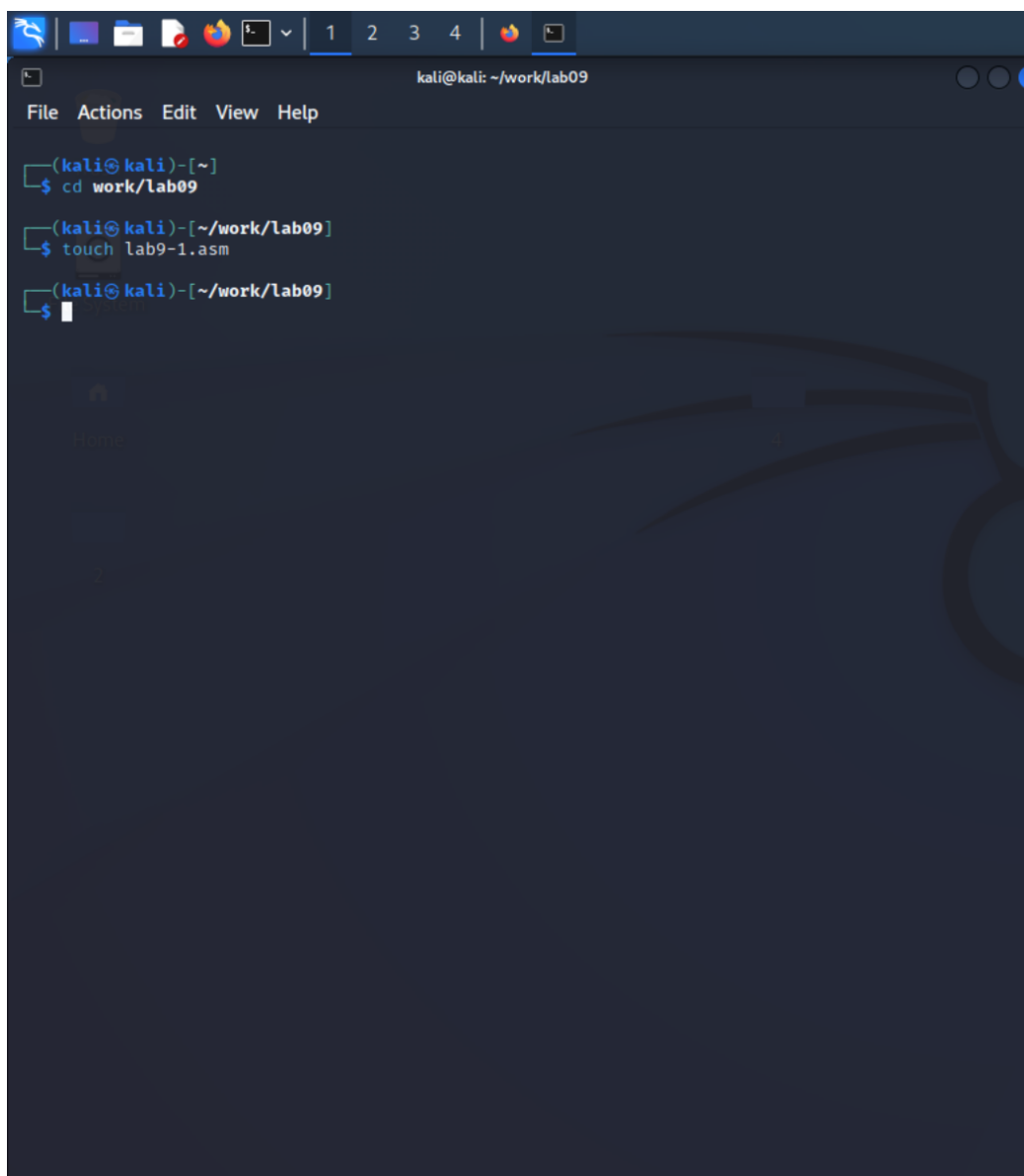
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1. Создадим каталог для лабораторной работы и файл lab9-1.asm (рис. 2.1)



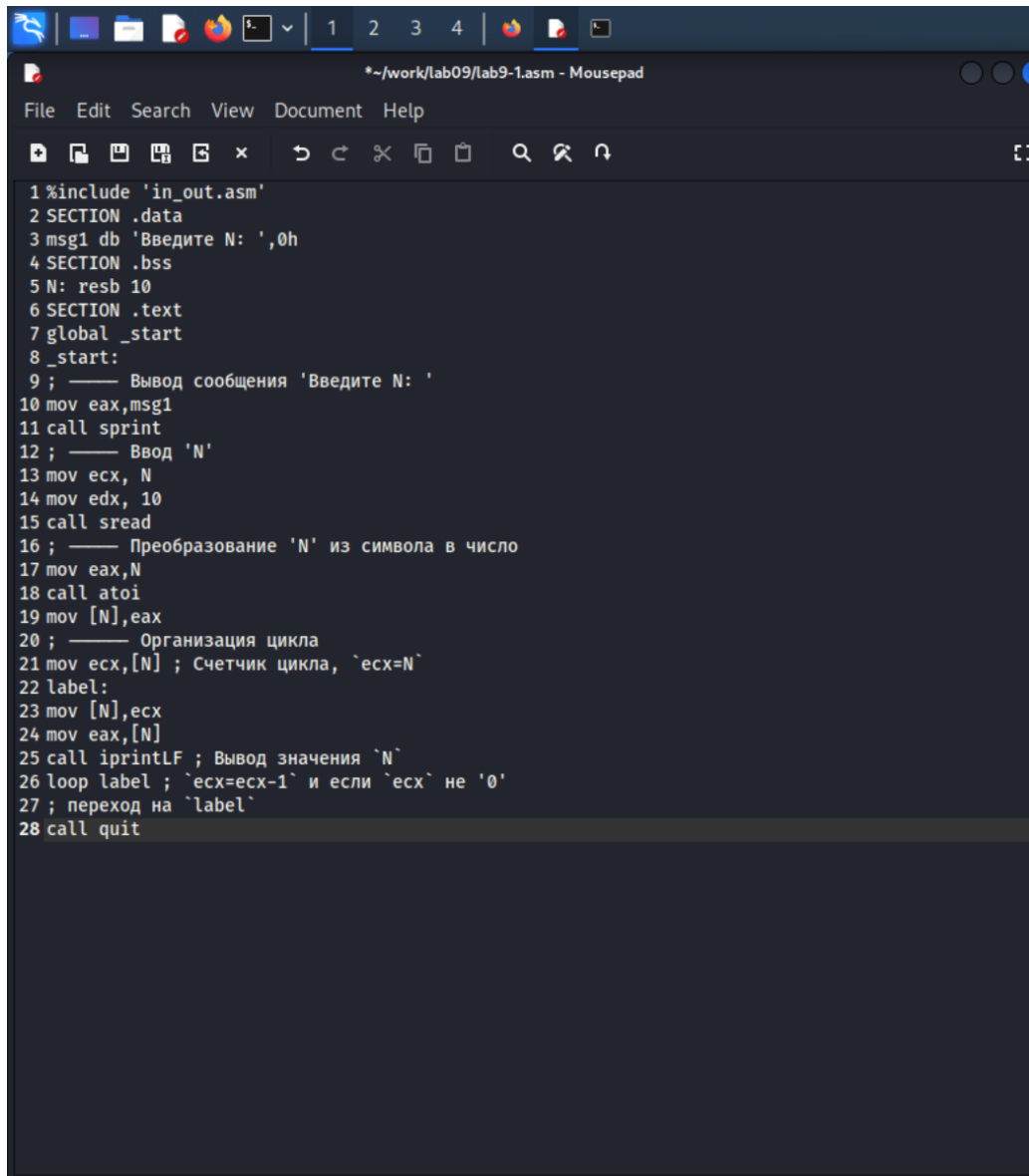
The image shows a terminal window with a dark blue background and light blue text. The window title is "kali@kali: ~/work/lab09". The terminal shows three commands being executed in sequence:

```
(kali@kali)-[~]  
$ cd work/lab09  
  
(kali@kali)-[~/work/lab09]  
$ touch lab9-1.asm  
  
(kali@kali)-[~/work/lab09]  
$
```

The prompt changes from "(kali@kali)-[~]" to "(kali@kali)-[~/work/lab09]" after the first command. The second command "touch lab9-1.asm" creates the file. The third command shows the prompt with a cursor, indicating the terminal is ready for further input.

Рис. 2.1: Создание файла lab9-1.asm

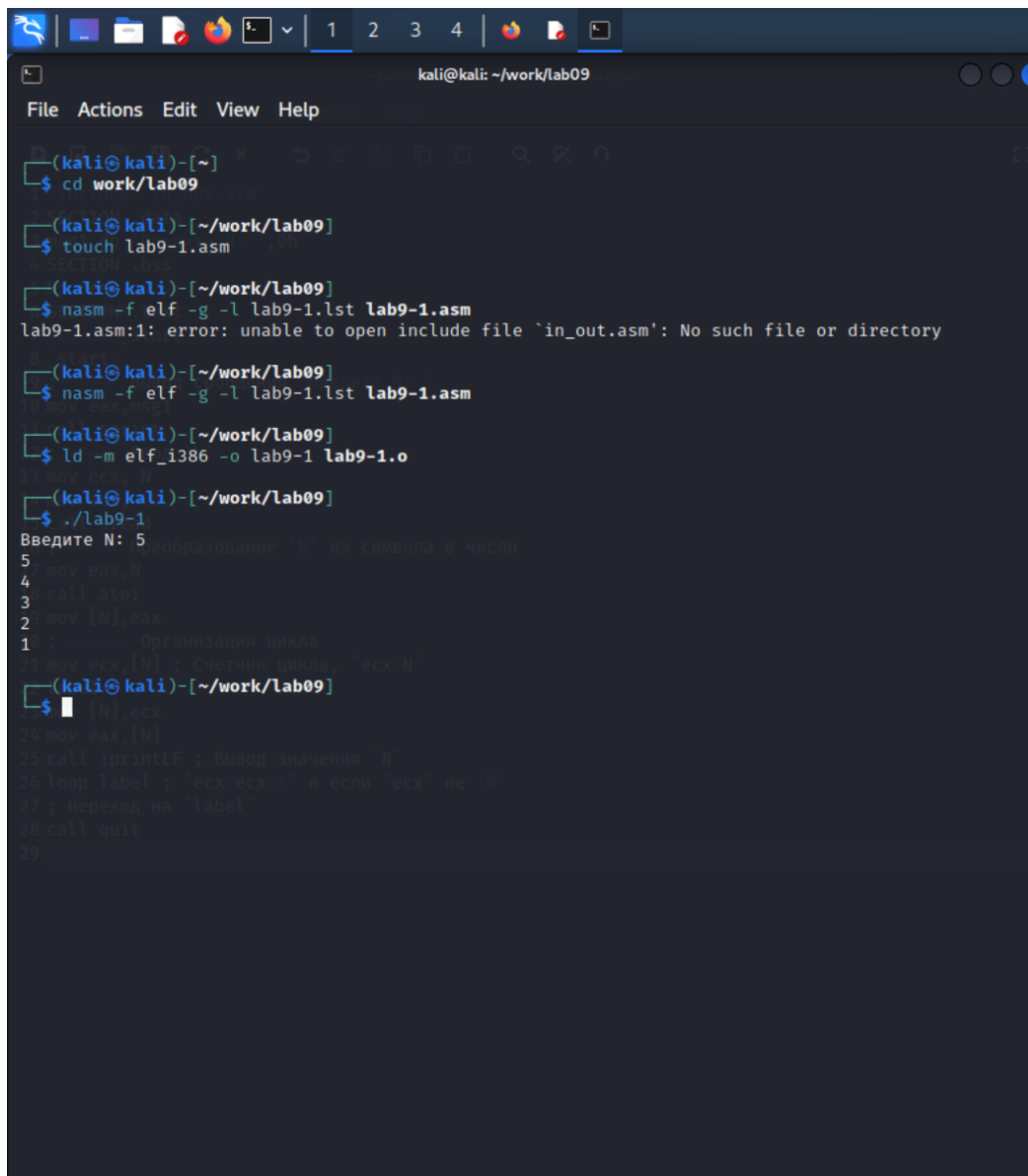
2. Запишем в него код из листинга 9.1(рис. 2.2)



```
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. 2.2: Листинг lab9-1.asm

3. Создадим исполняемый файл и проверим его работу(рис. 2.3)



```
kali@kali: ~/work/lab09
File Actions Edit View Help

(kali@kali)-[~]
$ cd work/lab09

(kali@kali)-[~/work/lab09]
$ touch lab9-1.asm

(kali@kali)-[~/work/lab09]
$ nasm -f elf -g -l lab9-1.lst lab9-1.asm
lab9-1.asm:1: error: unable to open include file `in_out.asm': No such file or directory

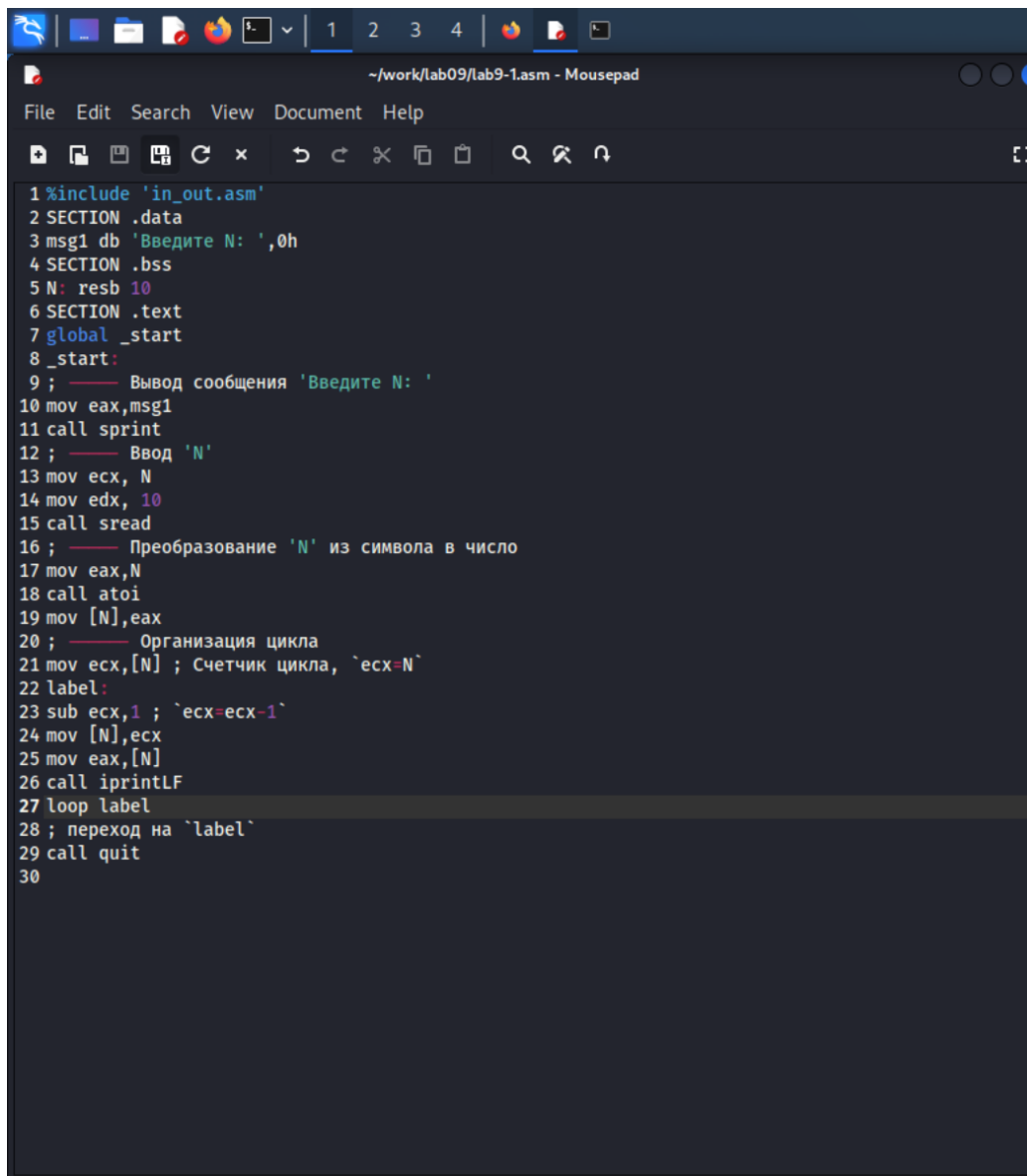
(kali@kali)-[~/work/lab09]
$ nasm -f elf -g -l lab9-1.lst lab9-1.asm

(kali@kali)-[~/work/lab09]
$ ld -m elf_i386 -o lab9-1 lab9-1.o

(kali@kali)-[~/work/lab09]
$ ./lab9-1
Введите N: 5
5      ; преобразование N из символа в число
4      mov eax,N
3      call atoi
2      mov [N],eax
1      ; Организация цикла
10     mov ecx,[N] ; Установка цикла на ecx N
11     loop label ; ecx-ecx и если ecx не 0
26     ; переход на label
27     call quit
28
29
```

Рис. 2.3: Компиляция и проверка lab9-1.asm

4. Изменим текст программы по инструкции в лабораторной(рис. 2.4)



```
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
30
```

Рис. 2.4: Измененный lab9-1.asm

5. Скомпилируем снова это файл. То как меняется ecx мы можем посмотреть по выводу программы, каждое выведенное число - это ecx. Из-за изменения регистра ecx число подходов цикла не соответствует числу введенному в программу(рис. 2.5)

```
kali@kali: ~/work/lab09
File Actions Edit View Help

(kali@kali)-[~]
$ cd work/lab09

(kali@kali)-[~/work/lab09]
$ touch lab9-1.asm

(kali@kali)-[~/work/lab09]
$ nasm -f elf -g -l lab9-1.lst lab9-1.asm
lab9-1.asm:1: error: unable to open include file 'in_out.asm': No such file or directory

(kali@kali)-[~/work/lab09]
$ nasm -f elf -g -l lab9-1.lst lab9-1.asm

(kali@kali)-[~/work/lab09]
$ ld -m elf_i386 -o lab9-1 lab9-1.o

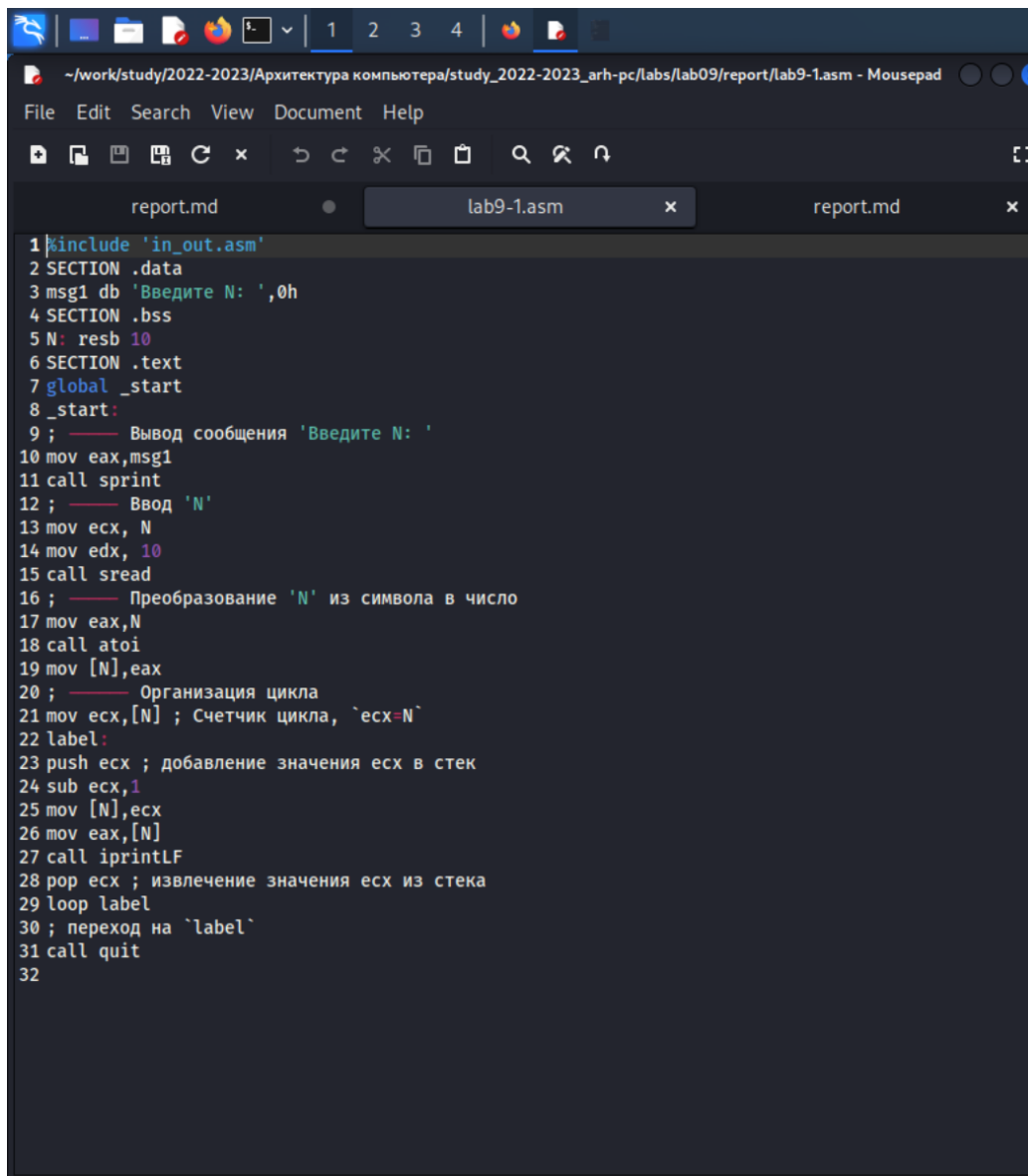
(kali@kali)-[~/work/lab09]
$ ./lab9-1
Введите N: 5
5      преобразование N из символа в число
4      mov eax,N
3      call atoi
2      mov [N],eax
1      Организация цикла
1      mov ecx,[N] - факторы цикла - ecx N
(kali@kali)-[~/work/lab09]
$ nasm -f elf -g -l lab9-1.lst lab9-1.asm

(kali@kali)-[~/work/lab09]
$ ld -m elf_i386 -o lab9-1 lab9-1.o

(kali@kali)-[~/work/lab09]
$ ./lab9-1
Введите N: 5
4
2
0
4294967294
4294967292
4294967290
4294967288
4294967286
4294967284
4294967282
4294967280
4294967278
4294967276
4294967274
```

Рис. 2.5: Компилирование и проверка измененного lab9-1.asm

6. Снова изменим программу, добавив push и pop.(рис. 2.6)



```
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 ; переход на `label`
31 call quit
32
```

Рис. 2.6: Код lab9-1.asm с push и pop

7. Скомпилируем и проверим. Проход циклов соответствует введёному с клавиатуры(рис. 2.7)

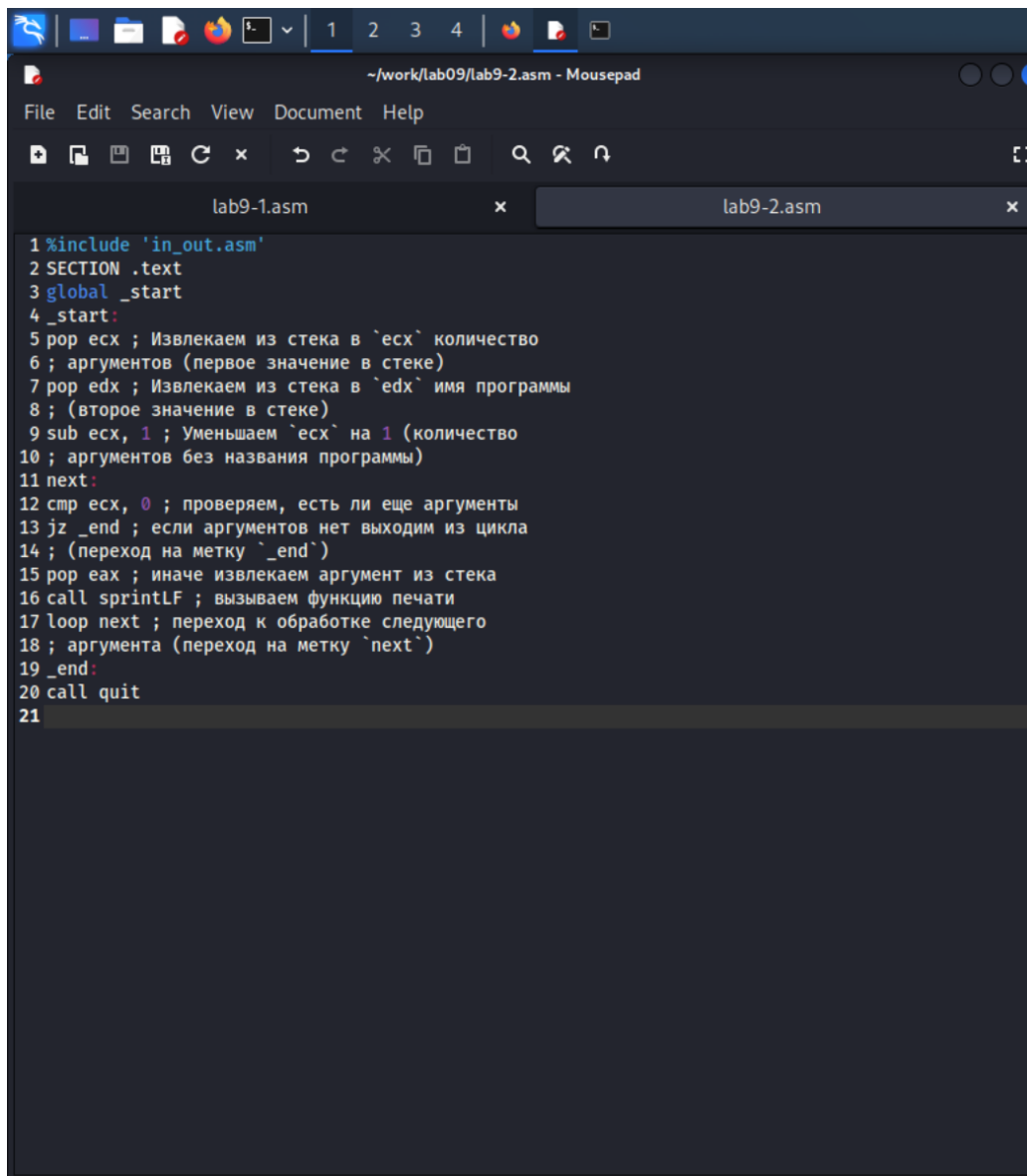
```
(kali@kali)-[~/work/lab09]
$ nasm -f elf -g -l lab9-1.lst lab9-1.asm

(kali@kali)-[~/work/lab09]
$ ld -m elf_i386 -o lab9-1 lab9-1.o

(kali@kali)-[~/work/lab09]
$ ./lab9-1.exe
Введите N: 5
4: _start
3: ; вывод сообщения "Введите N:"
2: mov eax,msg1
1: call printf
0: ; вывод N
(kali@kali)-[~/work/lab09]
$ cat lab9-1.asm
15: call $read
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 [hl],ecx
26: mov eax,[N]
27: call iprintlf
28: pop ecx ; извлечение значения ecx из стека
29: loop label
30: ; переход на label
31: call quit
32
```

Рис. 2.7: Проверка lab9-1.asm с push и pop

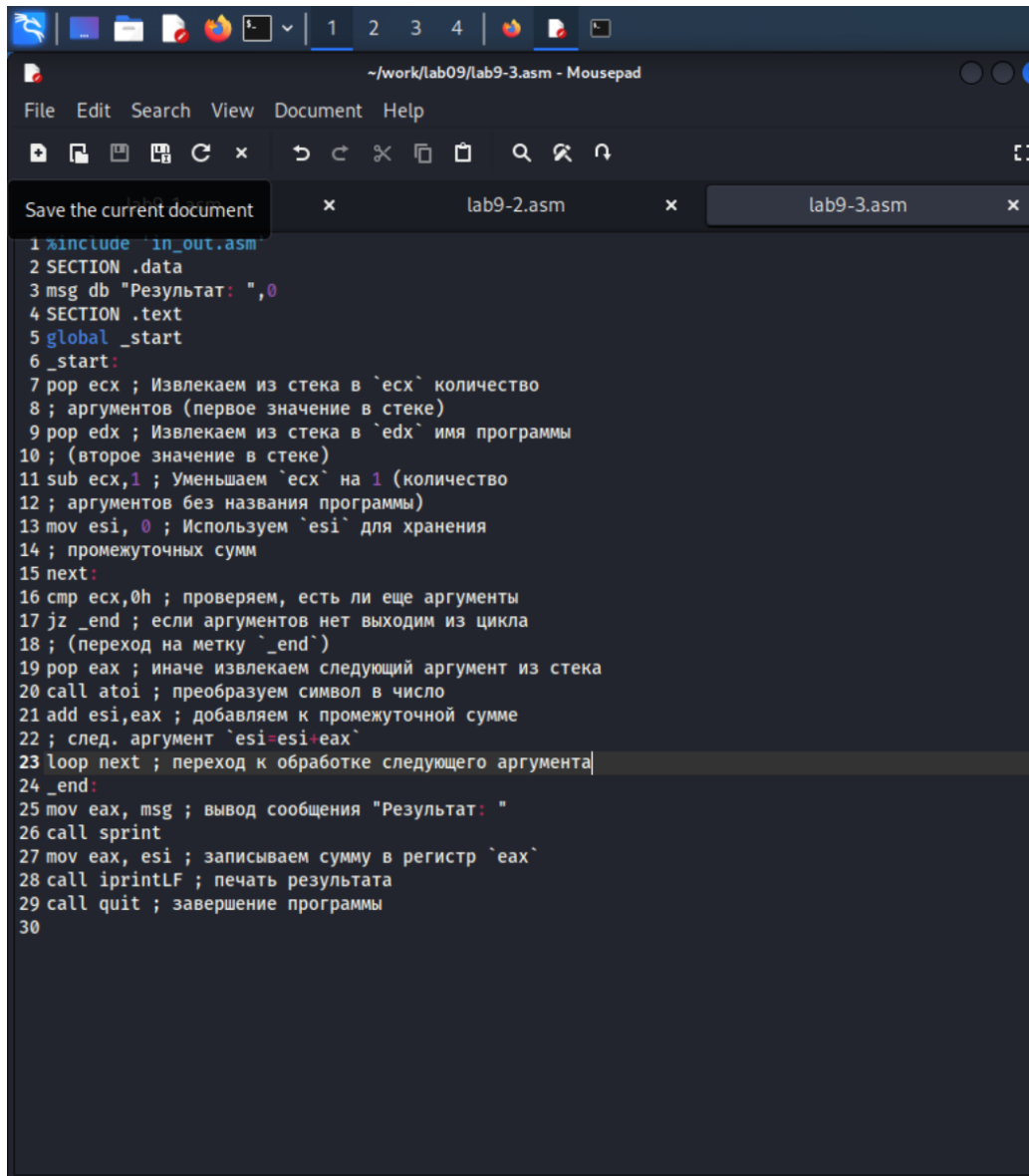
8. Создадим файл lab9-2.asm и введём код из листинга9.2 (рис. 2.8)



```
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
21
```

Рис. 2.8: Код lab9-2.asm

9. Проверим работу программы. Все аргументы были обработаны программой. (рис. 2.9)



```
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
30
```

Рис. 2.10: Код lab9-3.asm

11. Скомпилируем и проверим (рис. 2.11)


```
kali@kali: ~/work/lab09
File Actions Edit View Help

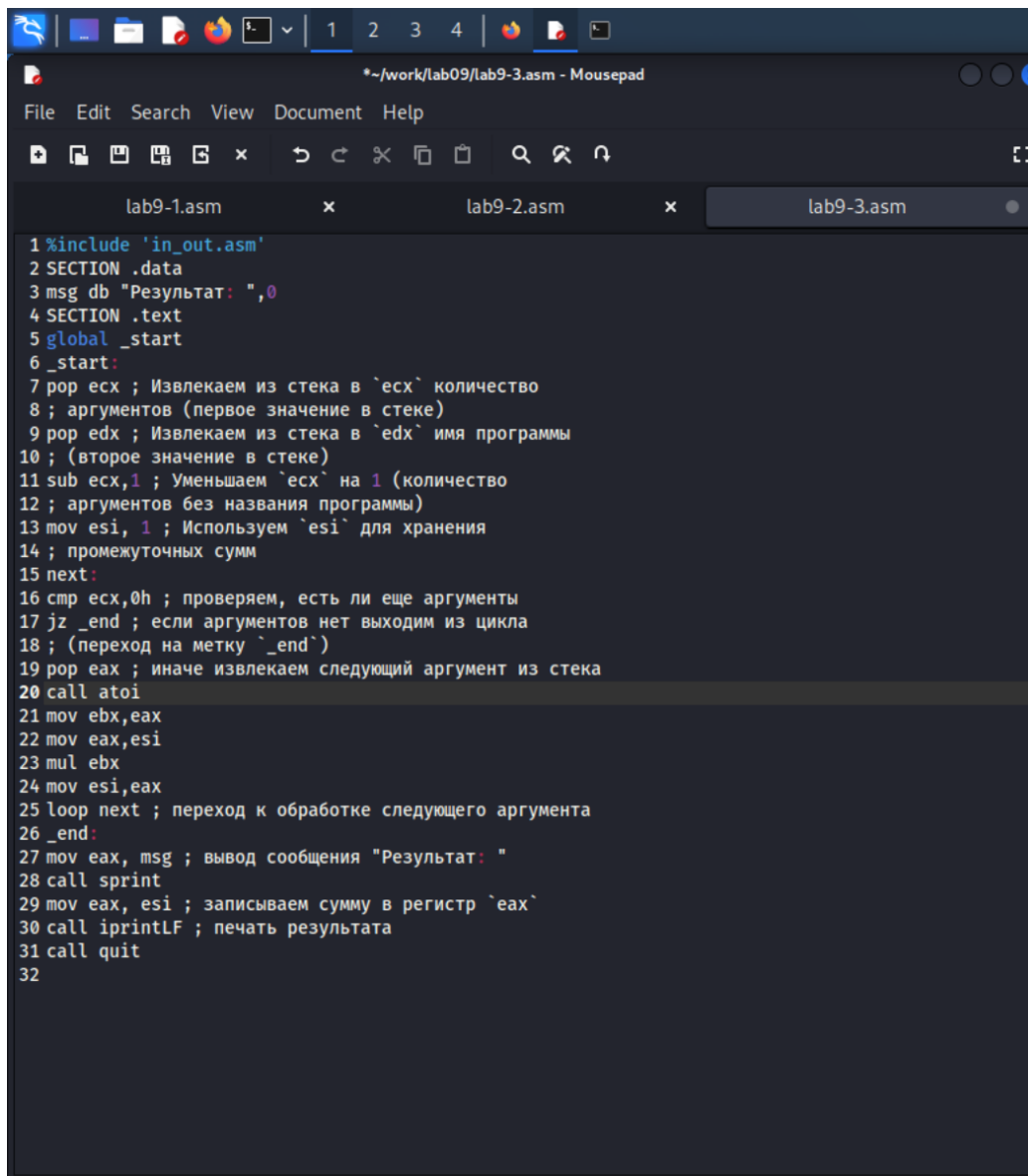
(kali@kali)~/work/lab09
$ nasm -f elf -g -l lab9-2.lst lab9-2.asm

(kali@kali)~/work/lab09
$ ld -m elf_i386 -o lab9-2 lab9-2.o

(kali@kali)~/work/lab09
$ ./lab9-2
./lab9-2 1 2 3
1 pop eax ; Извлекаем из стека в "eax" количество
2 ; аргументов (первое значение в стеке)
3 pop edx ; Извлекаем из стека в "edx" имя программы
10 ; Извлекаем значение в стек
(kali@kali)~/work/lab09
$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
; для хранения
$ ld -m elf_i386 -o lab9-3 lab9-3.o
(kali@kali)~/work/lab09
$ ./lab9-3
Результат: 0
(kali@kali)~/work/lab09
$ ./lab9-3 12 13 7 10 5
Результат: 47
(kali@kali)~/work/lab09
$
```

Рис. 2.11: Проверка lab9-3.asm

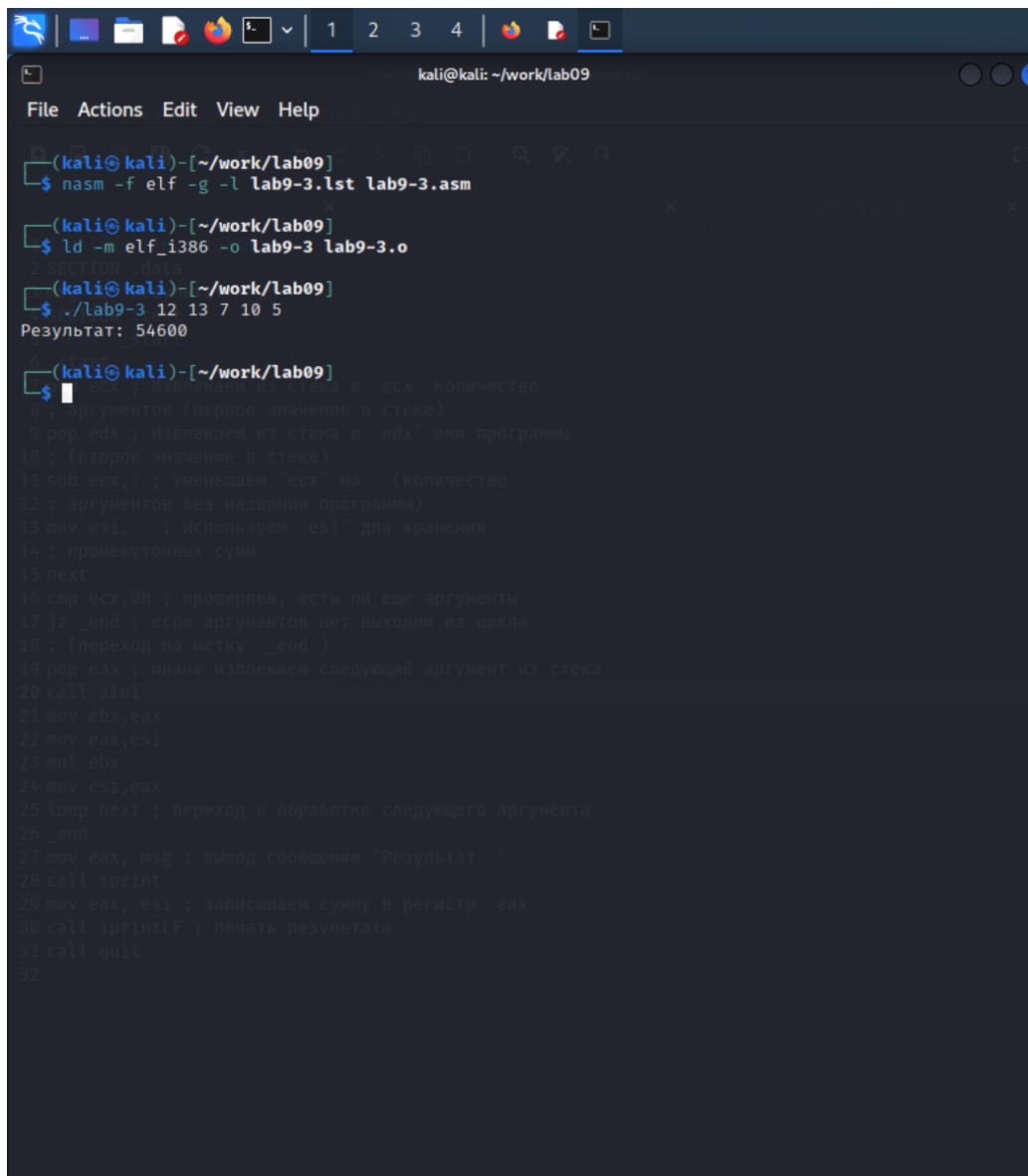
12. Изменим код программы, для вычисления произведения аргументов (рис. 2.12)



```
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
21 mov ebx,eax
22 mov eax,esi
23 mul ebx
24 mov esi,eax
25 loop next ; переход к обработке следующего аргумента
26 _end:
27 mov eax,msg ; вывод сообщения "Результат: "
28 call sprint
29 mov eax,esi ; записываем сумму в регистр `eax`
30 call iprintLF ; печать результата
31 call quit
32
```

Рис. 2.12: Измененный код lab9-3.asm

13. Проверим работу программы. Ответ при тех же аргументах должен быть 54600. И мы убедились, что всё работает хорошо. (рис. 2.13)



```
kali@kali: ~/work/lab09
File Actions Edit View Help

(kali@kali)~/work/lab09
$ nasm -f elf -g -l lab9-3.lst lab9-3.asm

(kali@kali)~/work/lab09
$ ld -m elf_i386 -o lab9-3 lab9-3.o

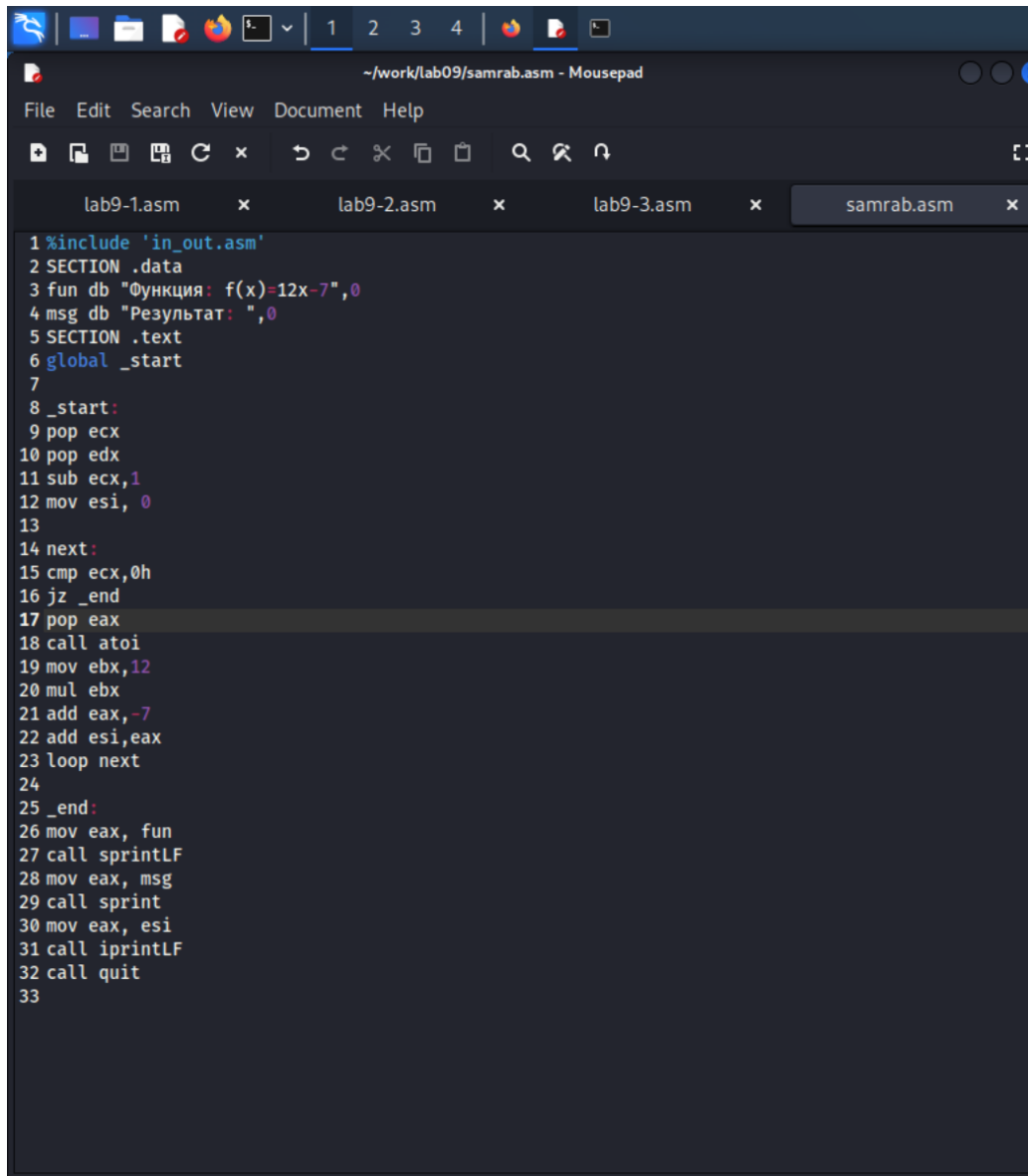
(kali@kali)~/work/lab09
$ ./lab9-3 12 13 7 10 5
Результат: 54600

(kali@kali)~/work/lab09
$ cat lab9-3.asm
; количество
; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в "edx" имя программы
10 ; (второе значение в стеке)
11 sub ecx, 4 ; уменьшаем "ecx" на 4 (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
21 mov ebx, eax
22 mov eax, esi
23 mul ebx
24 mov esi, eax
25 loop next ; переход к обработке следующего аргумента
26 _end
27 mov eax, msg ; вывод сообщения "Результат: "
28 call sprintf
29 mov ecx, esi ; записываем сумму в регистр "ecx"
30 call iprintf ; печать результата
31 call quit
32
```

Рис. 2.13: Проверка измененного кода lab9-3.asm

3 Выполнение самостоятельной работы

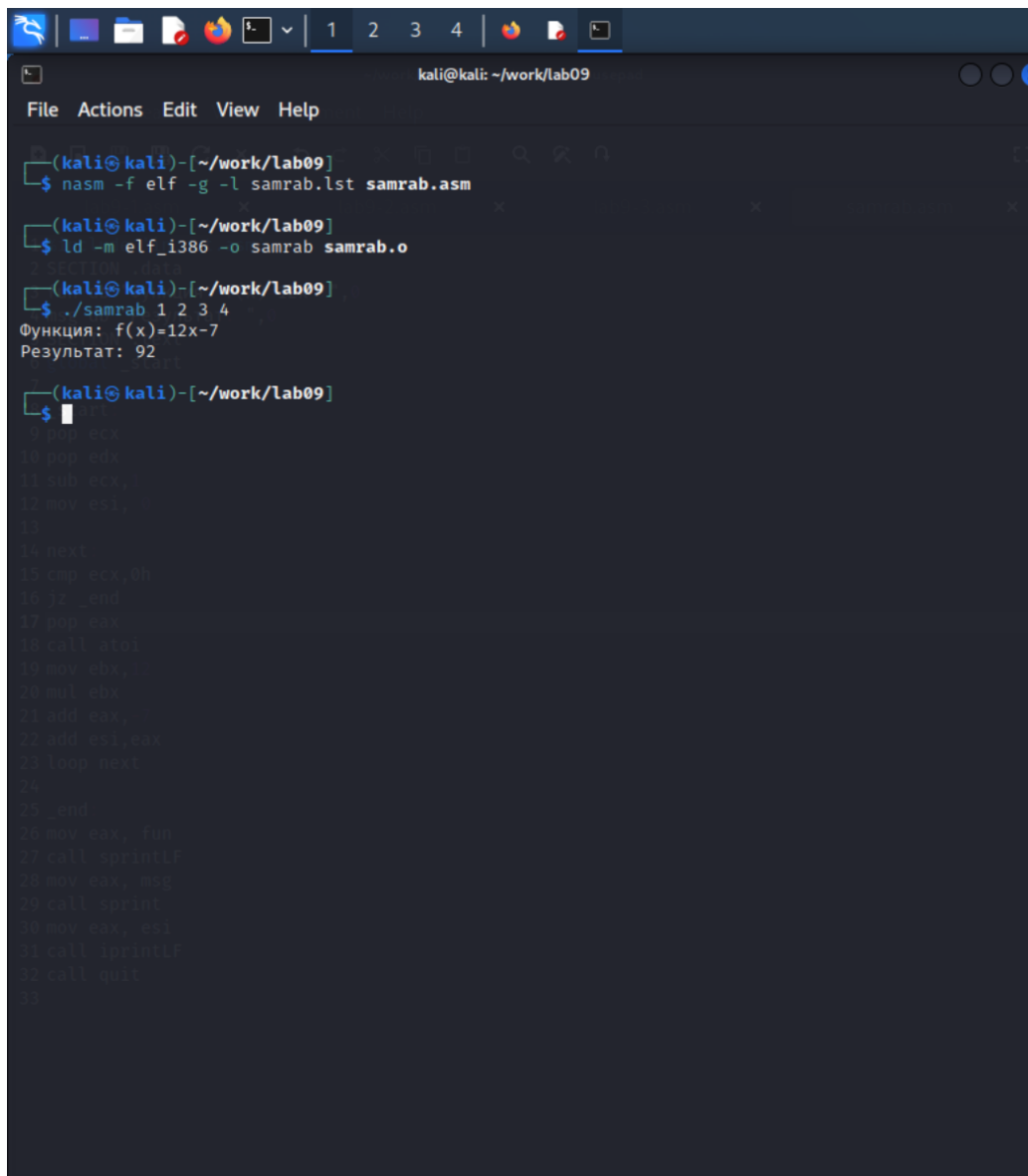
1. Прочитаем задание самостоятельной работы и напишем программу по моему 13 варианту.(рис. 3.1)



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

Рис. 3.1: Код программы для самостоятельной

2. Проверим его работу. Ответ верны, проверил калькулятором. (рис. 3.2)



```
kali@kali: ~/work/lab09
File Actions Edit View Help

(kali@kali)~/work/lab09
$ nasm -f elf -g -l samrab.lst samrab.asm

(kali@kali)~/work/lab09
$ ld -m elf_i386 -o samrab samrab.o

(kali@kali)~/work/lab09
$ ./samrab 1 2 3 4
Функция: f(x)=12x-7
Результат: 92

(kali@kali)~/work/lab09
$ cat
9 pop ecx
10 pop edx
11 sub ecx,1
12 mov esi,1
13
14 next
15 cmp ecx,0h
16 jz _end
17 pop eax
18 call atoi
19 mov ebx,1
20 mul ebx
21 add eax,1
22 add esi,eax
23 loop next
24
25 _end
26 mov eax, fun
27 call sprintf
28 mov eax, msg
29 call sprintf
30 mov eax, esi
31 call sprintf
32 call quit
33
```

Рис. 3.2: Проверка самостоятельной

4 Выводы

В данной лабораторной работе я приобрел навыки написания программ на языке NASM с использованием циклов и обработкой командной строкой.
https://github.com/Sinabon2004/study_2022-2023_arh-pc