

Лабораторная работа №9

Программирование цикла. Обработка аргументов командной строки.

Хрусталеv Влад Николаевич

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Выполнение самостоятельной работы	19
4	Выводы	21

Список иллюстраций

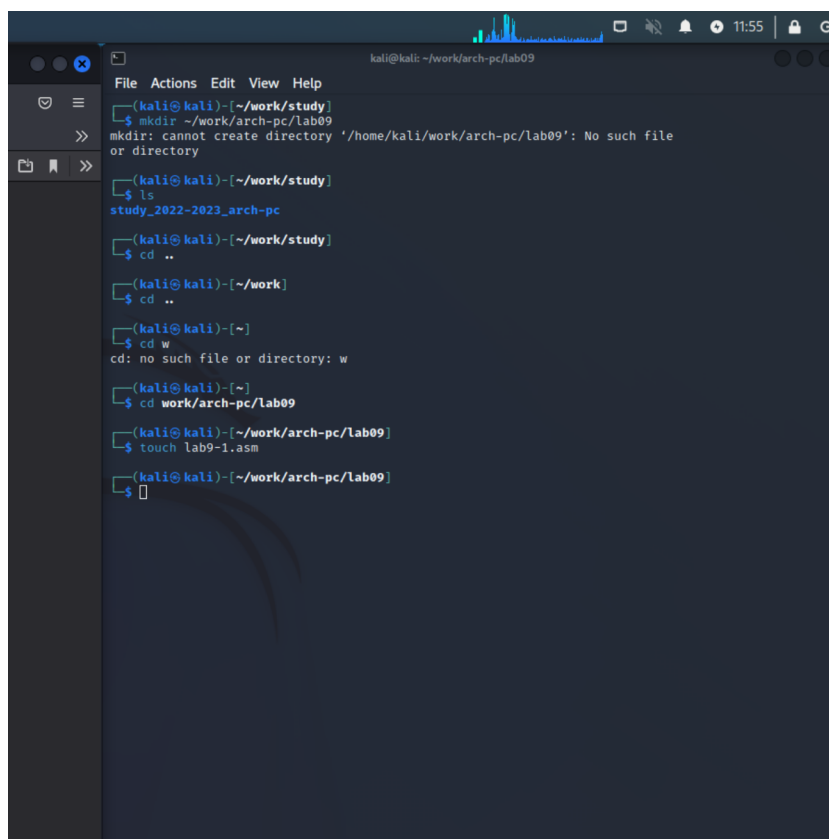
2.1	Название рисунка	5
2.2	Название рисунка	6
2.3	Название рисунка	7
2.4	Название рисунка	8
2.5	Название рисунка	9
2.6	Название рисунка	10
2.7	Название рисунка	11
2.8	Название рисунка	12
2.9	Название рисунка	13
2.10	Название рисунка	14
2.11	Название рисунка	15
2.12	Название рисунка	16
2.13	Название рисунка	17
2.14	Название рисунка	18
3.1	Название рисунка	19
3.2	Название рисунка	20

1 Цель работы

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

2 Выполнение лабораторной работы

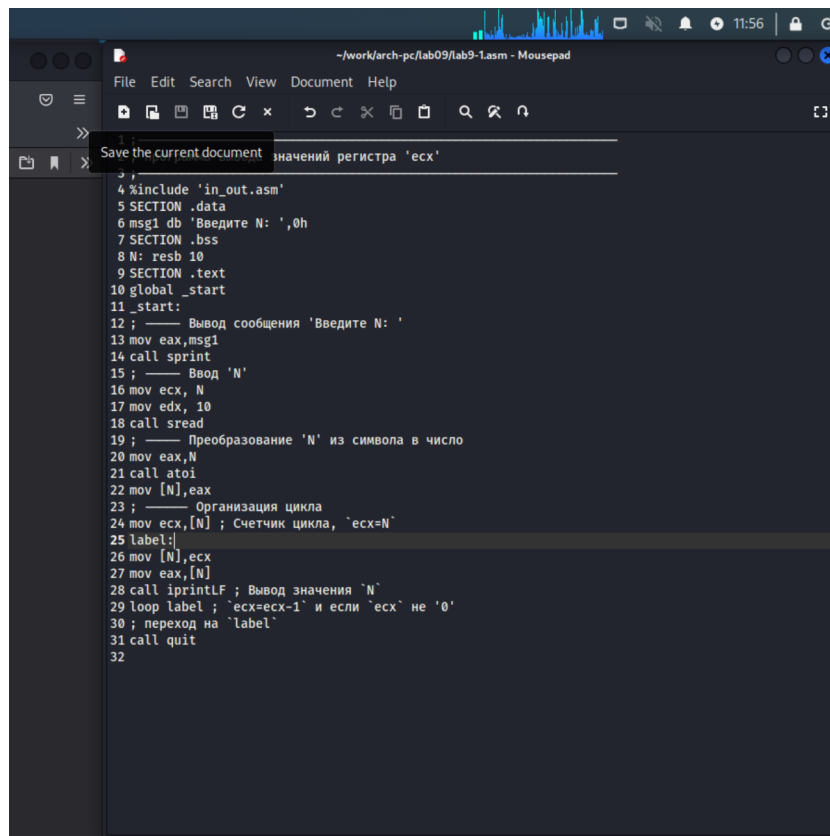
Создадим файл lab9-1.asm (рис. 2.1)



```
kali@kali: ~/work/arch-pc/lab09
File Actions Edit View Help
(kali@kali)-[~/work/study]
$ mkdir ~/work/arch-pc/lab09
mkdir: cannot create directory '/home/kali/work/arch-pc/lab09': No such file
or directory
(kali@kali)-[~/work/study]
$ ls
study_2022-2023_arch-pc
(kali@kali)-[~/work/study]
$ cd ..
(kali@kali)-[~/work]
$ cd ..
(kali@kali)-[~]
$ cd w
cd: no such file or directory: w
(kali@kali)-[~]
$ cd work/arch-pc/lab09
(kali@kali)-[~/work/arch-pc/lab09]
$ touch lab9-1.asm
(kali@kali)-[~/work/arch-pc/lab09]
$
```

Рис. 2.1: Название рисунка

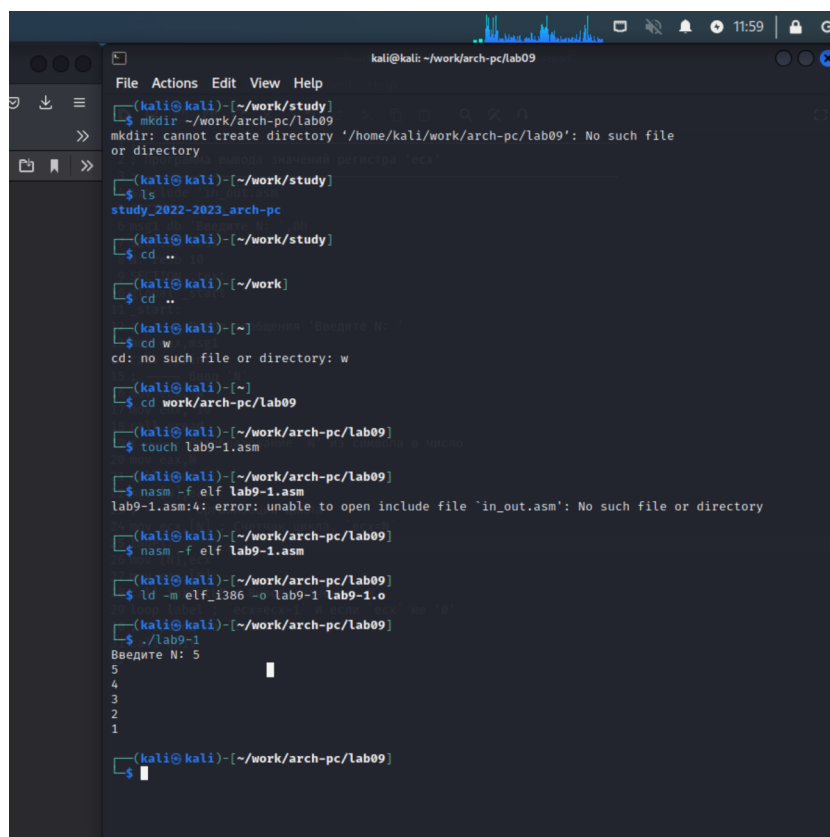
Введём программу из листинга 9.1 (рис. 2.2)



```
1 ;
2 ;
3 ;
4 %include 'in_out.asm'
5 SECTION .data
6 msg1 db 'Введите N: ',0h
7 SECTION .bss
8 N: resb 10
9 SECTION .text
10 global _start
11 _start:
12 ; ----- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call sprint
15 ; ----- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call sread
19 ; ----- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
25 label:
26 mov [N],ecx
27 mov eax,[N]
28 call iprintf ; Вывод значения 'N'
29 loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
30 ; переход на 'label'
31 call quit
32
```

Рис. 2.2: Название рисунка

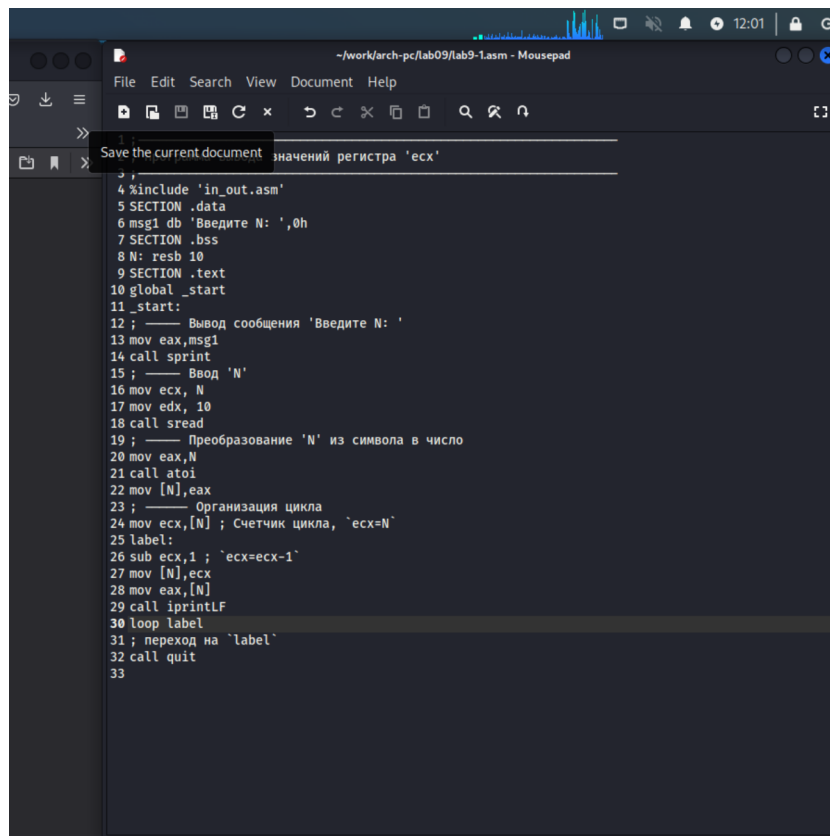
Проверим его работу (рис. 2.3)



```
kali@kali: ~/work/arch-pc/lab09
File Actions Edit View Help
(kali@kali)-[~/work/study]
$ mkdir ~/work/arch-pc/lab09
mkdir: cannot create directory '/home/kali/work/arch-pc/lab09': No such file
or directory
(kali@kali)-[~/work/study]
$ ls
study_2022-2023_arch-pc
(kali@kali)-[~/work/study]
$ cd ..
(kali@kali)-[~/work]
$ cd ..
(kali@kali)-[~]
$ cd w
cd: no such file or directory: w
(kali@kali)-[~]
$ cd work/arch-pc/lab09
(kali@kali)-[~/work/arch-pc/lab09]
$ touch lab9-1.asm
(kali@kali)-[~/work/arch-pc/lab09]
$ nasm -f elf lab9-1.asm
lab9-1.asm:4: error: unable to open include file 'in_out.asm': No such file or directory
(kali@kali)-[~/work/arch-pc/lab09]
$ nasm -f elf lab9-1.asm
(kali@kali)-[~/work/arch-pc/lab09]
$ ld -m elf_i386 -o lab9-1 lab9-1.o
(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-1
Введите N: 5
5
4
3
2
1
(kali@kali)-[~/work/arch-pc/lab09]
$
```

Рис. 2.3: Название рисунка

Изменим текст программы по инструкции в лабораторной добавив изменение регистра есх (рис. 2.4)



```
1 ;
2 ;
3 ;
4 %include 'in_out.asm'
5 SECTION .data
6 msg1 db 'Введите N: ',0h
7 SECTION .bss
8 N: resb 10
9 SECTION .text
10 global _start
11 _start:
12 ; ----- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call sprint
15 ; ----- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call sread
19 ; ----- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
25 label:
26 sub ecx,1 ; 'ecx=ecx-1'
27 mov [N],ecx
28 mov eax,[N]
29 call iprintLF
30 loop label
31 ; переход на 'label'
32 call quit
33
```

Рис. 2.4: Название рисунка

Проверим его работу и выясним, что из-за изменения регистра ecx число подходов цикла не соответствует числу введенному в программу (рис. 2.5)

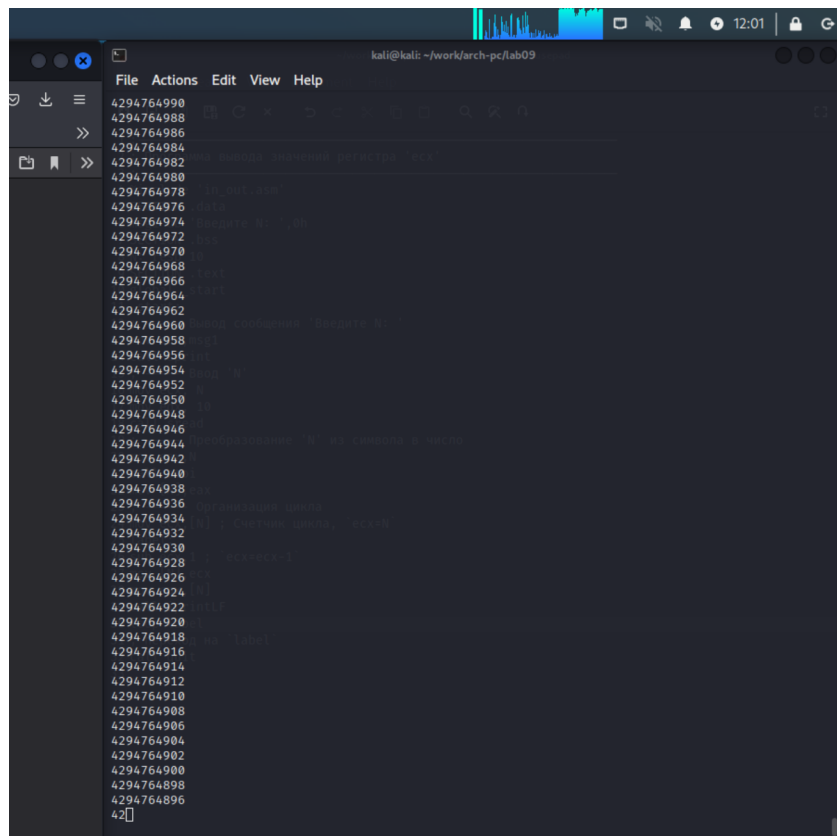
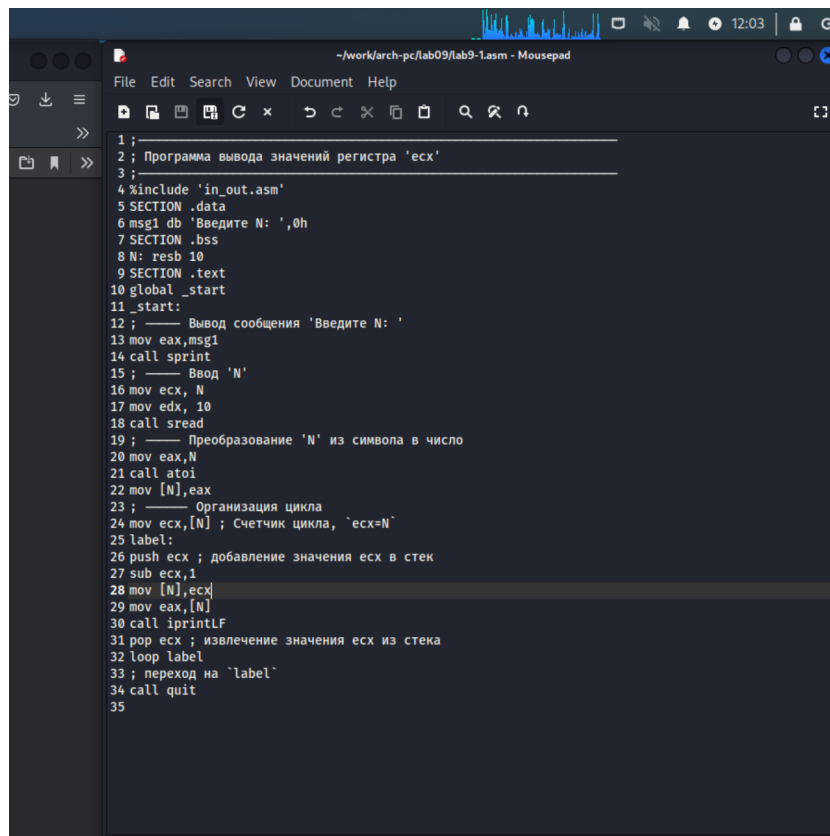


Рис. 2.5: Название рисунка

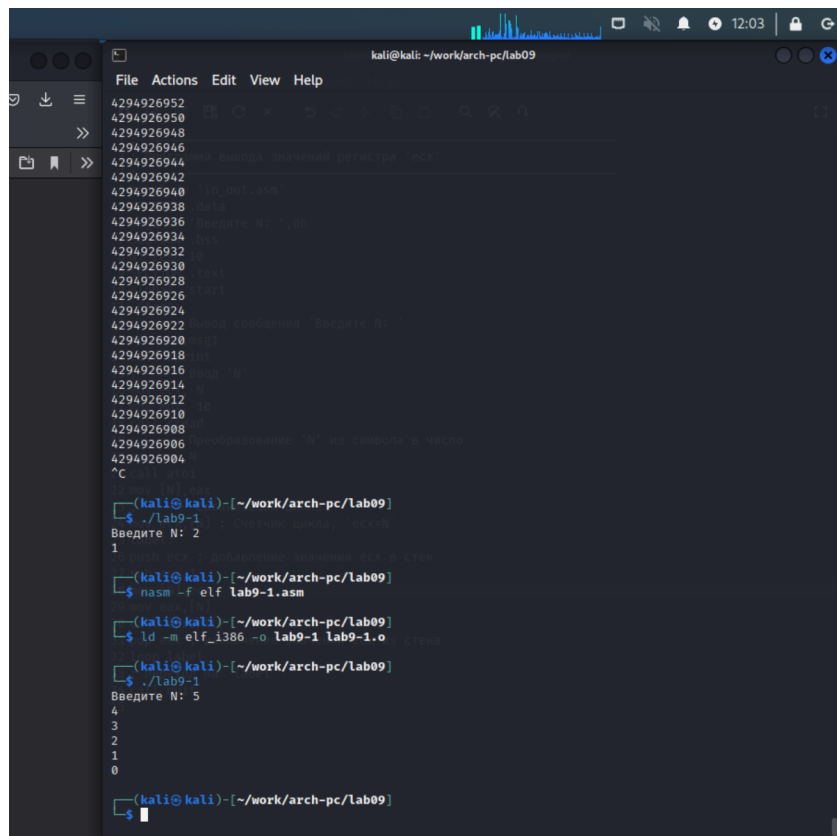
Изменим программу добавив команды push и pop (рис. 2.6)



```
1 ;
2 ; Программа вывода значений регистра 'ecx'
3 ;
4 %include 'in_out.asm'
5 SECTION .data
6 msg1 db 'Введите N: ',0h
7 SECTION .bss
8 N: resb 10
9 SECTION .text
10 global _start
11 _start:
12 ; ----- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call sprint
15 ; ----- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call sread
19 ; ----- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
25 label:
26 push ecx ; добавление значения ecx в стек
27 sub ecx,1
28 mov [N],ecx
29 mov eax,[N]
30 call iprintf
31 pop ecx ; извлечение значения ecx из стека
32 loop label
33 ; переход на 'label'
34 call quit
35
```

Рис. 2.6: Название рисунка

Проверим работоспособность и выясним что число циклов соответствует заданному(рис. 2.7)

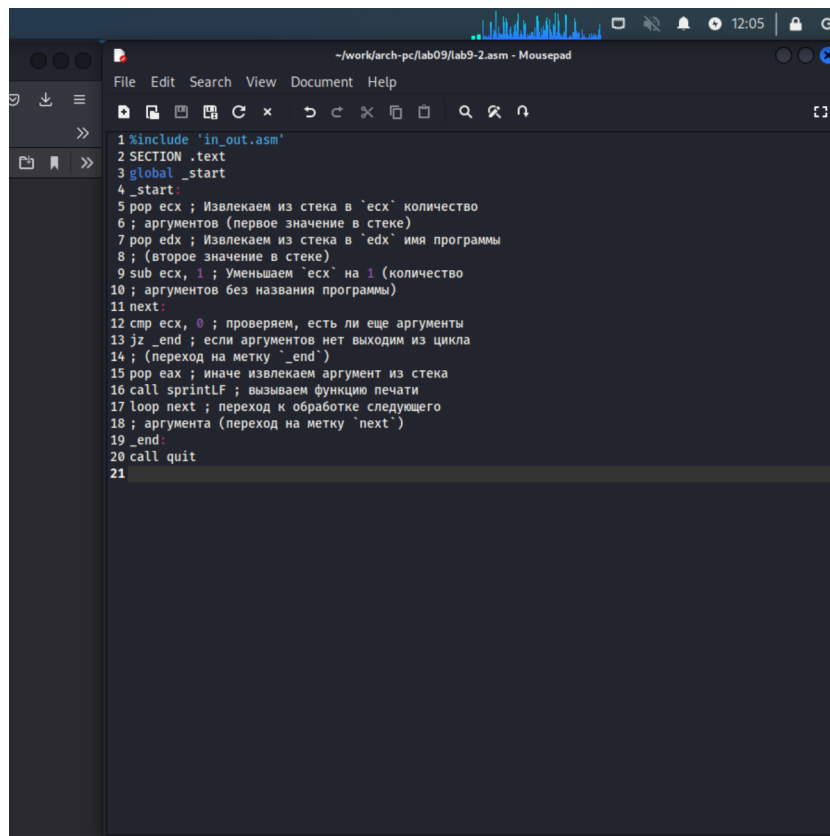


```
kali@kali: ~/work/arch-pc/lab09
File Actions Edit View Help
4294926952
4294926950
4294926948
4294926946
4294926944
4294926942
4294926940
4294926938
4294926936
4294926934
4294926932
4294926930
4294926928
4294926926
4294926924
4294926922
4294926920
4294926918
4294926916
4294926914
4294926912
4294926910
4294926908
4294926906
4294926904
^C

(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-1
Введите N: 2
1
(kali@kali)-[~/work/arch-pc/lab09]
$ nasm -f elf lab9-1.asm
(kali@kali)-[~/work/arch-pc/lab09]
$ ld -m elf_i386 -o lab9-1 lab9-1.o
(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-1
Введите N: 5
4
3
2
1
0
(kali@kali)-[~/work/arch-pc/lab09]
$
```

Рис. 2.7: Название рисунка

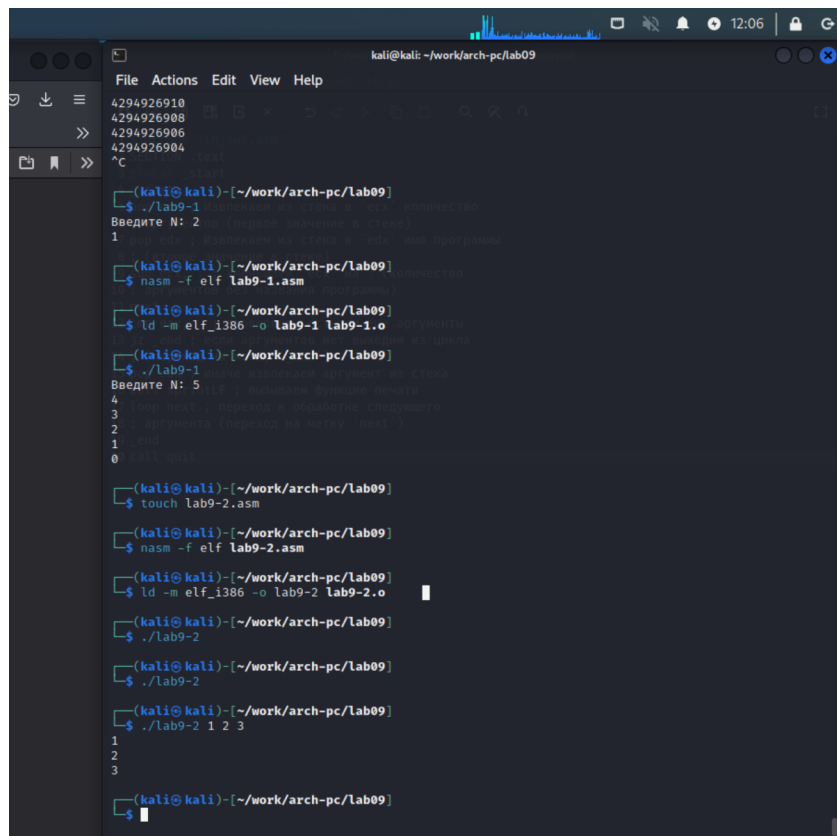
Создадим файл lab9-2.asm и введём в него текст программы из лситинга 9.2 (рис. 2.8)



```
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: Название рисунка

Проверим его работоспособность все аргументы были обработаны (рис. 2.9)



```
kali@kali: ~/work/arch-pc/lab09
File Actions Edit View Help
4294926910
4294926908
4294926906
4294926904
^C
(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-1
Введите N: 2
1
(kali@kali)-[~/work/arch-pc/lab09]
$ nasm -f elf lab9-1.asm
(kali@kali)-[~/work/arch-pc/lab09]
$ ld -m elf_i386 -o lab9-1 lab9-1.o
(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-1
Введите N: 5
4
3
2
1
0
(kali@kali)-[~/work/arch-pc/lab09]
$ touch lab9-2.asm
(kali@kali)-[~/work/arch-pc/lab09]
$ nasm -f elf lab9-2.asm
(kali@kali)-[~/work/arch-pc/lab09]
$ ld -m elf_i386 -o lab9-2 lab9-2.o
(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-2
(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-2
(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-2 1 2 3
1
2
3
(kali@kali)-[~/work/arch-pc/lab09]
$
```

Рис. 2.9: Название рисунка

Создадим файл lab9-3.asm (рис. 2.10)

```
kali@kali: ~/work/arch-pc/lab09
File Actions Edit View Help
4294926904
^C
(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-1
Введите N: 2
1

(kali@kali)-[~/work/arch-pc/lab09]
$ nasm -f elf lab9-1.asm

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

(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-1
Введите N: 5
4
3
2
1
0

(kali@kali)-[~/work/arch-pc/lab09]
$ touch lab9-2.asm

(kali@kali)-[~/work/arch-pc/lab09]
$ nasm -f elf lab9-2.asm

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

(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-2

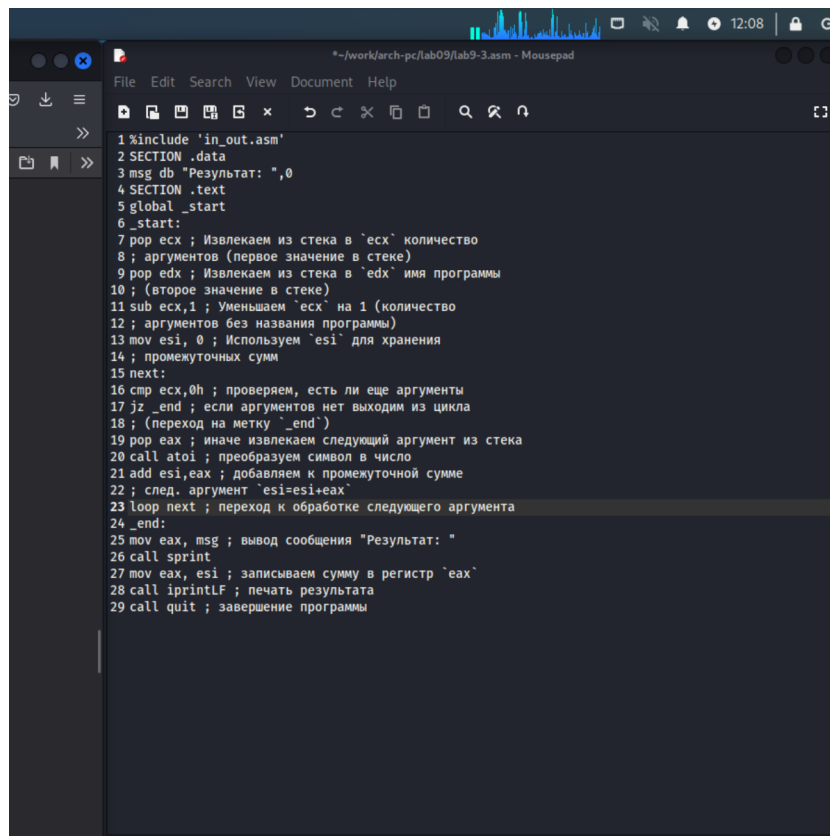
(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-2 1 2 3
1
2
3

(kali@kali)-[~/work/arch-pc/lab09]
$ touch lab9-3.asm

(kali@kali)-[~/work/arch-pc/lab09]
$
```

Рис. 2.10: Название рисунка

Введём программу из листинга 9.3 (рис. 2.11)



```
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 sprintf
27 mov eax, esi ; записываем сумму в регистр `eax`
28 call iprintlf ; печать результата
29 call quit ; завершение программы
```

Рис. 2.11: Название рисунка

Проверим как работает программа на данных из лабораторной (рис. 2.12)

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

(kali@kali)-[~/work/arch-pc/lab09]
$ nasm -f elf lab9-3.asm

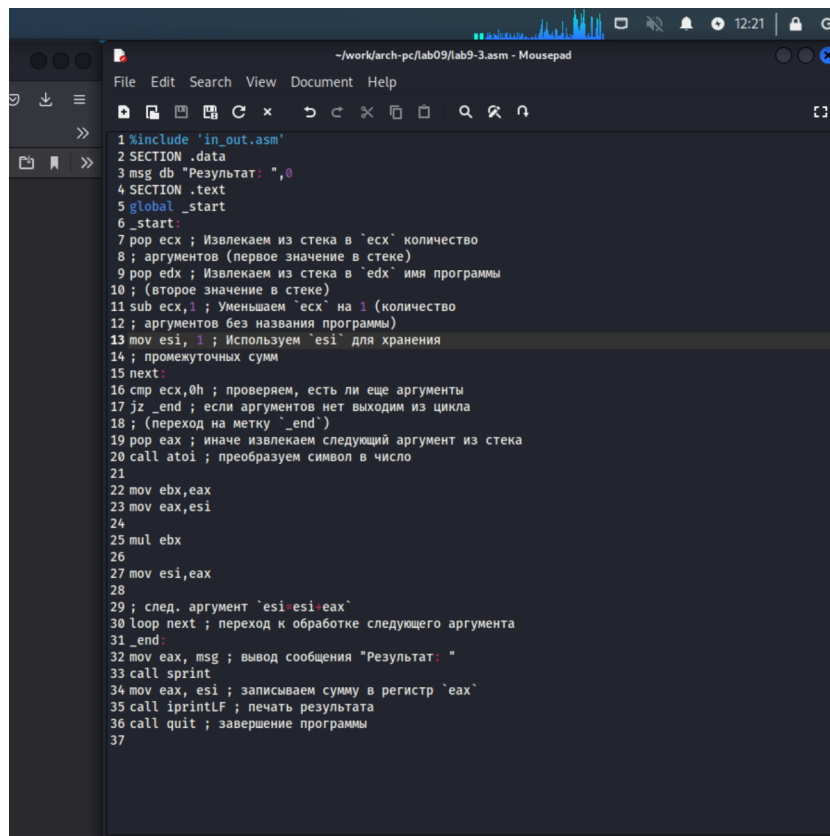
(kali@kali)-[~/work/arch-pc/lab09]
$ ld -m elf_i386 -o lab9-3 lab9-3.o

(kali@kali)-[~/work/arch-pc/lab09]
$ ./lab9-3 12 13 7 10 5
Результат: 47

(kali@kali)-[~/work/arch-pc/lab09]
$
```

Рис. 2.12: Название рисунка

Изменим эту программу так, чтобы она вычисляла произведение аргументов (рис. 2.13)



```
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 ; преобразуем символ в число
21
22 mov ebx,eax
23 mov eax,esi
24
25 mul ebx
26
27 mov esi,eax
28
29 ; след. аргумент `esi-esi-eax`
30 loop next ; переход к обработке следующего аргумента
31 _end:
32 mov eax,msg ; вывод сообщения "Результат: "
33 call sprint
34 mov eax,esi ; записываем сумму в регистр `eax`
35 call iprintf ; печать результата
36 call quit ; завершение программы
37
```

Рис. 2.13: Название рисунка

Проверим его работоспособность (рис. 2.14)

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

(kali@kali)-[~/work/arch-pc/lab09]
$ nasm -f elf lab9-3.asm

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

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

(kali@kali)-[~/work/arch-pc/lab09]
$
```

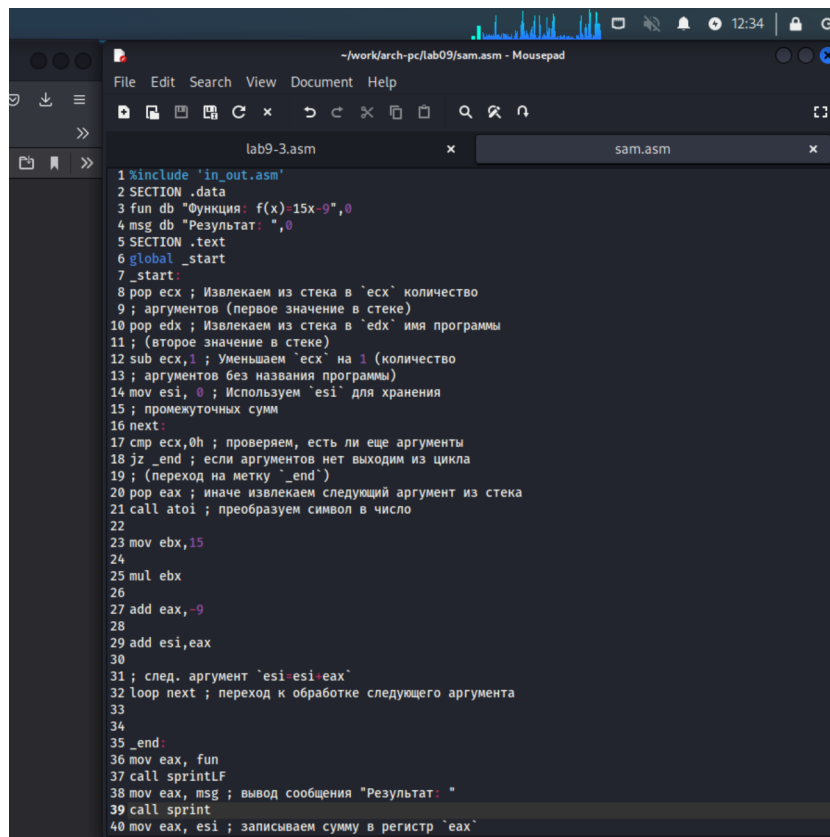
10. `push ecx` - сохраняем ecx на стек
11. `add ecx, 4` - увеличиваем ecx на 4 байта
12. `pop ecx` - берем ecx из стека
13. `mov esi, 0` - инициализируем esi
14. `push esi` - сохраняем esi на стек
15. `next`
16. `cmp ecx, 0` - проверяем, есть ли еще аргументы
17. `je .end` - если аргументов нет, выходим из цикла
18. `inc ecx` - переходим на следующий аргумент
19. `jmp ecx` - переходим к началу цикла
20. `call atoi` - преобразуем строку в число
21.
22. `mov ebx, eax`
23. `mov esi, ebx`
24. `mul esi`
25.
26. `mov esi, eax`
27. `add esi, ebx`
28. `mov ebx, esi`
29. `loop next` - переход к обработке следующего аргумента
30. `pop esi`
31. `mov ebx, ebx` - вывод окончательного результата
32. `call printf`
33. `mov ebx, ebx` - записываем сумму в регистр ebx
34. `call printf` - вывод результата
35. `call exit` - завершение программы
36.

Рис. 2.14: Название рисунка

3 Выполнение самостоятельной работы

Мой вариант 12, то есть самостоятельная работа: сумма всех $f(x)$, где x аргументы, а $f(x)=15x-9$.

Напишем программу чтобы выполняла эту задачу(рис. 3.1)



```
1 %include 'in_out.asm'
2 SECTION .data
3 fun db "Функция: f(x)=15x-9",0
4 msg db "Результат: ",0
5 SECTION .text
6 global _start
7 _start:
8 pop ecx ; Извлекаем из стека в 'ecx' количество
9 ; аргументов (первое значение в стеке)
10 pop edx ; Извлекаем из стека в 'edx' имя программы
11 ; (второе значение в стеке)
12 sub ecx,1 ; Уменьшаем 'ecx' на 1 (количество
13 ; аргументов без названия программы)
14 mov esi, 0 ; Используем 'esi' для хранения
15 ; промежуточных сумм
16 next:
17 cmp ecx,0h ; проверяем, есть ли еще аргументы
18 jz _end ; если аргументов нет выходим из цикла
19 ; (переход на метку '_end')
20 pop eax ; иначе извлекаем следующий аргумент из стека
21 call atoi ; преобразуем символ в число
22
23 mov ebx,15
24
25 mul ebx
26
27 add eax,-9
28
29 add esi,eax
30
31 ; след. аргумент 'esi-ecx'
32 loop next ; переход к обработке следующего аргумента
33
34
35 _end:
36 mov eax, fun
37 call sprintf
38 mov eax, msg ; вывод сообщения "Результат: "
39 call sprintf
40 mov eax, esi ; записываем сумму в регистр 'eax'
```

Рис. 3.1: Название рисунка

Проверим его работоспособность на паре примеров и убедимся в работоспособности (рис. 3.2)

4 Выводы

На данной лабораторной я приобрёл навыки написания программ ассемблера NASM с использованием циклов и обработкой аргументов командной строки.

https://github.com/bezura/study_2022-2023_arch-pc