

Шаблон отчёта по лабораторной работе

8

Талебу Тенке Франк Устон НКАбд-05-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы :	6
2.1	Реализация циклов в NASM :	6
2.2	Обработка аргументов командной строки :	10
2.3	Программа вычисления суммы аргументов командной строки : .	12
2.4	Выводы по результатам выполнения заданий :	14
3	Задание для самостоятельной работы :	15
3.1	Выводы по результатам выполнения заданий :	17
4	Выводы, согласованные с целью работы :	18
	Список литературы	19

Список иллюстраций

2.1	Ресунок	6
2.2	Ресунок	7
2.3	Ресунок	8
2.4	Ресунок	9
2.5	Ресунок	10
2.6	Ресунок	11
2.7	Ресунок	11
2.8	Ресунок	12
2.9	Ресунок	13
2.10	Ресунок	13
2.11	Ресунок	14
3.1	Ресунок	16
3.2	Ресунок	17

Список таблиц

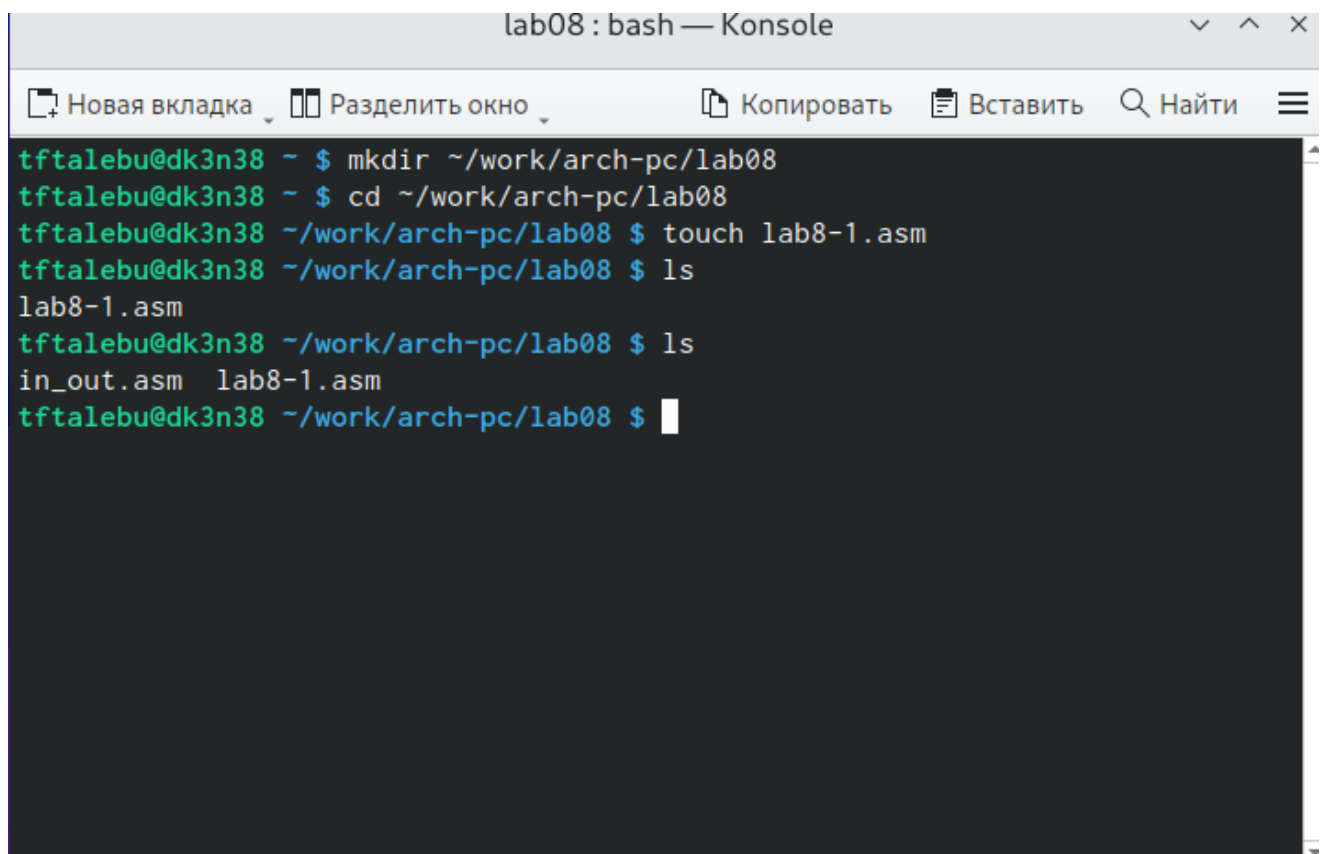
1 Цель работы

- В 8 лабораторной работе мы научимся писать программы с циклами и обработкой аргументов с помощью командной строки.

2 Выполнение лабораторной работы :

2.1 Реализация циклов в NASM :

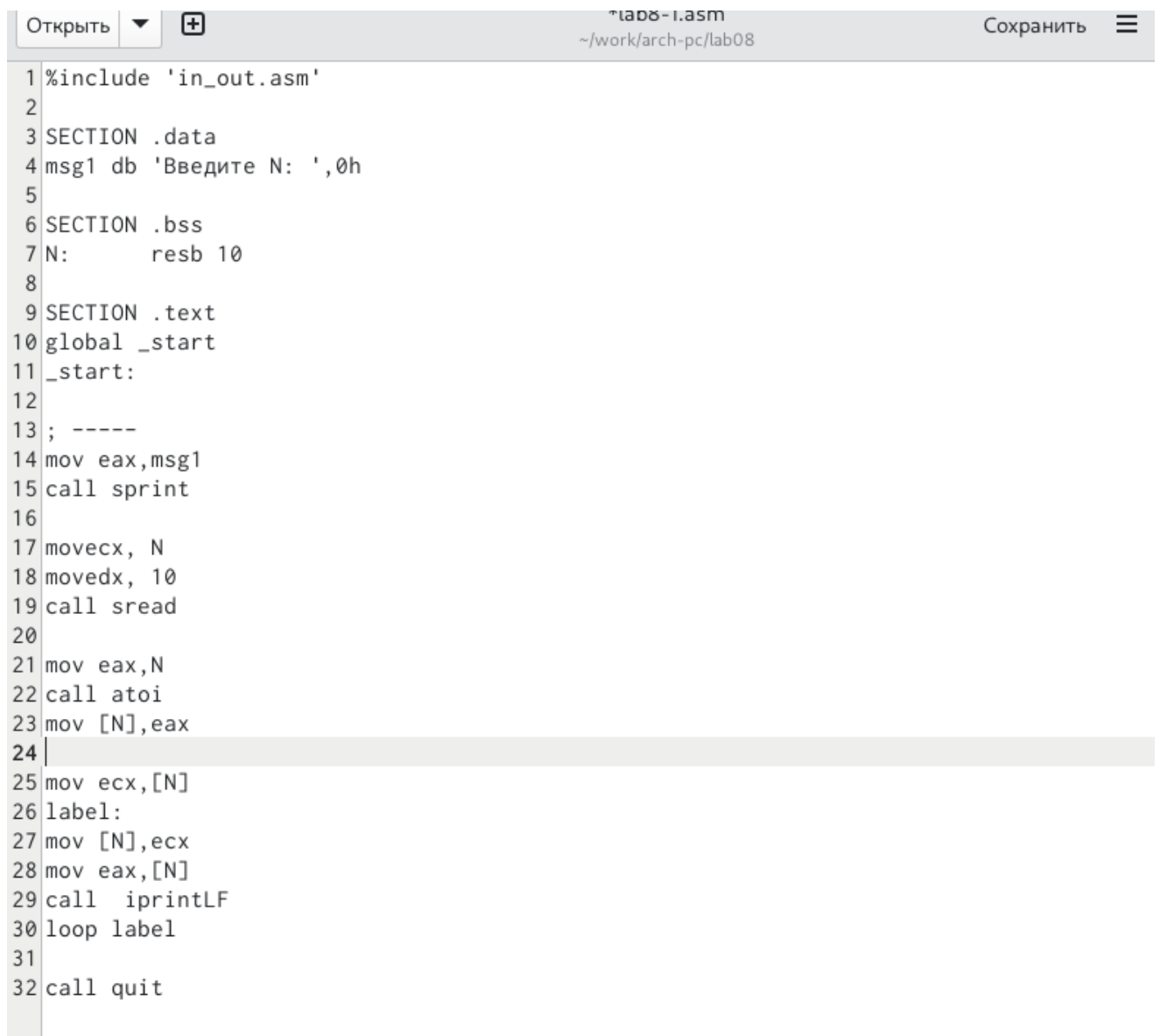
- Здесь мы начали с создания каталога для программы лабораторной работы No 8, а затем переместились в 8 каталог лаборатории “~/work/arch- pc/lab08”, после чего мы создали файл “lab8-1.asm”. (рис. [2.1])



```
lab08 : bash — Konsole
Новая вкладка  Разделить окно  Копировать  Вставить  Найти  ≡
tftalebu@dk3n38 ~ $ mkdir ~/work/arch-pc/lab08
tftalebu@dk3n38 ~ $ cd ~/work/arch-pc/lab08
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ touch lab8-1.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ls
lab8-1.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ls
in_out.asm  lab8-1.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $
```

Рис. 2.1: Ресунок

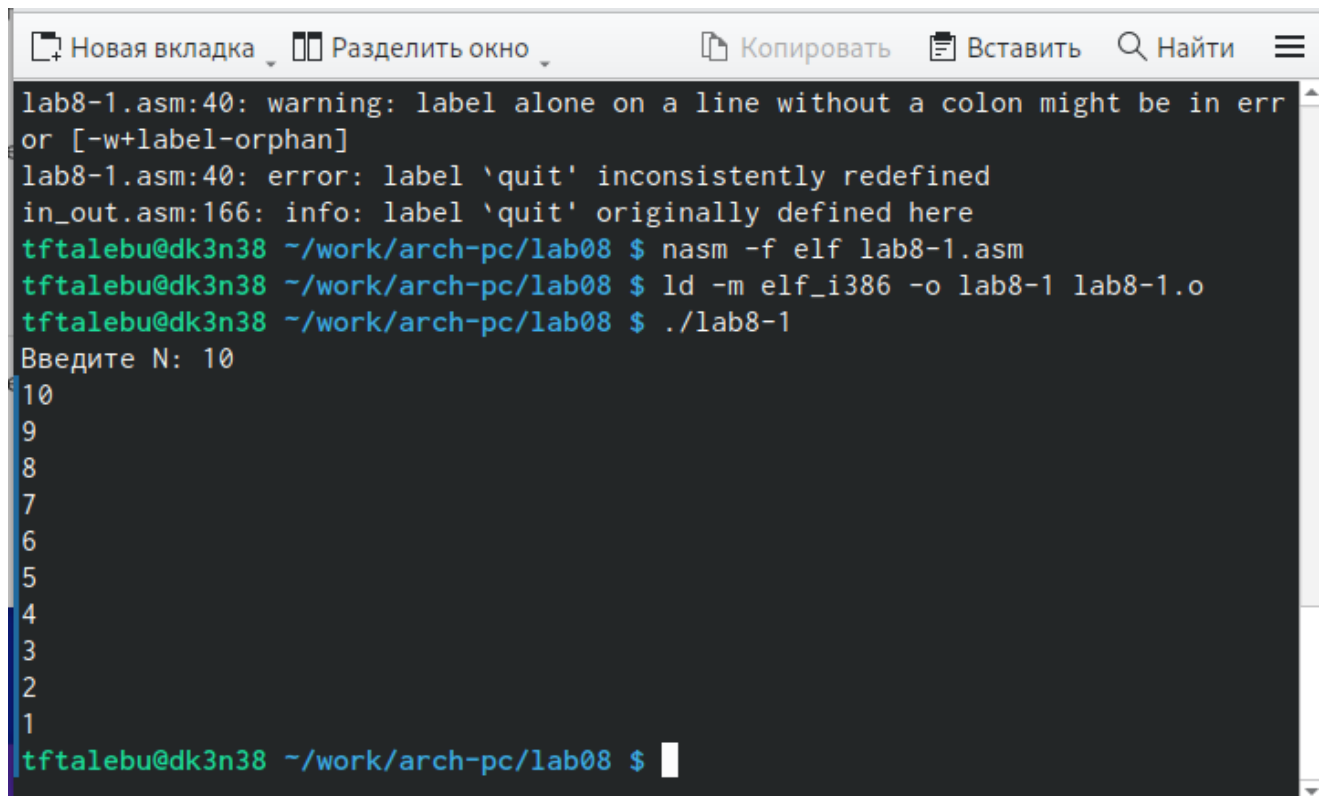
- Затем мы заполнили код нашей программы в файле lab8-1.asm. (рис. [2.2])



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

Рис. 2.2: Ресунок

- После этого мы скомпилировали файл, создали исполняемый файл и проверили его работу.(рис. [2.3])

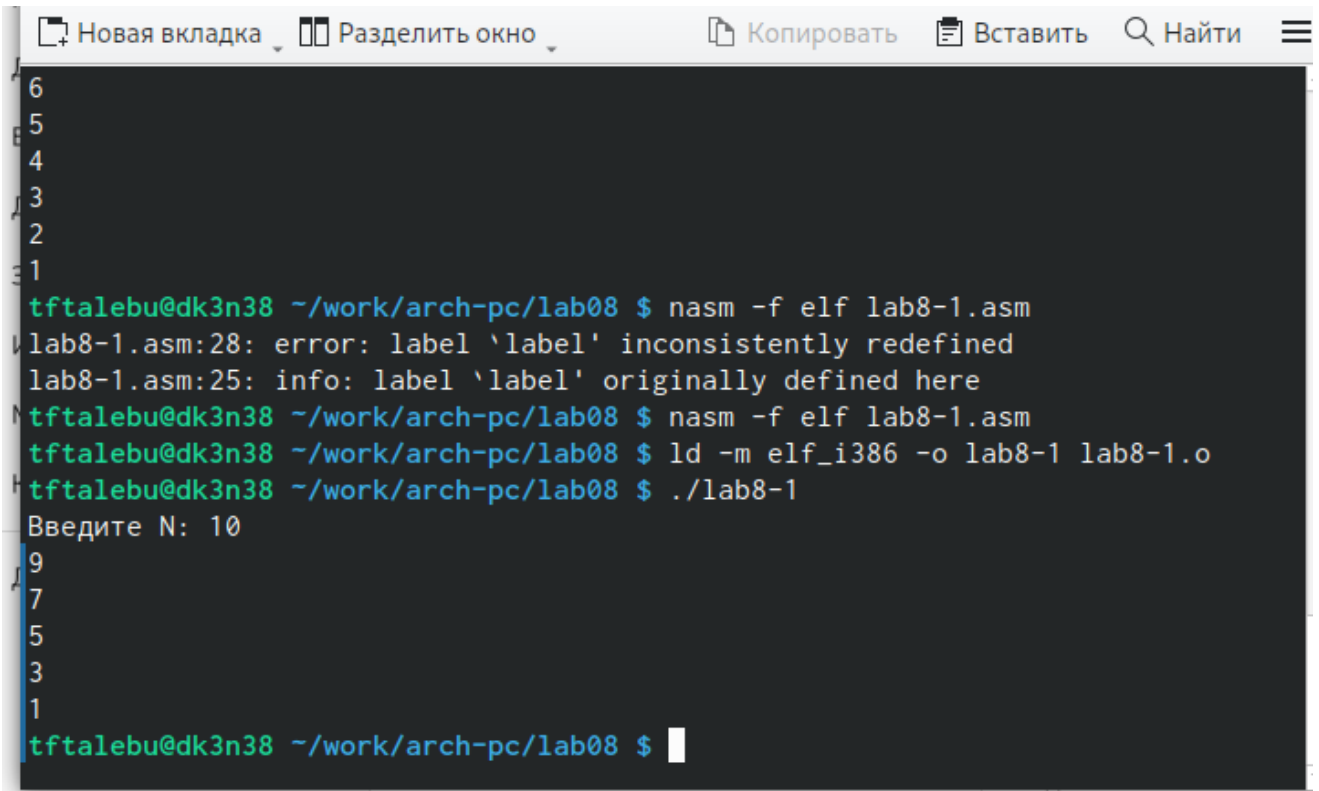


The screenshot shows a terminal window with a dark background and light-colored text. At the top, there is a menu bar with icons and text in Russian: 'Новая вкладка', 'Разделить окно', 'Копировать', 'Вставить', 'Найти', and a hamburger menu icon. The terminal output shows the following sequence of events:

```
lab8-1.asm:40: warning: label alone on a line without a colon might be in error [-w+label-orphan]
lab8-1.asm:40: error: label 'quit' inconsistently redefined
in_out.asm:166: info: label 'quit' originally defined here
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
tftalebu@dk3n38 ~/work/arch-pc/lab08 $
```

Рис. 2.3: Ресунок

- Мы внесли изменения в наш код, а затем создали исполняемый файл.(рис. [2.4])



```
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
lab8-1.asm:28: error: label 'label' inconsistently redefined
lab8-1.asm:25: info: label 'label' originally defined here
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
7
5
3
1
tftalebu@dk3n38 ~/work/arch-pc/lab08 $
```

Рис. 2.4: Ресунок

- Регистр еsx принимает пять значений, которые являются: 9,7,5,3,1, мы можем заметить, что количество циклов не соответствует числу, введенному пользователем
- На этот раз мы использовали стек, и в конечном итоге количество циклов соответствует числу, которое было введено в начале.(рис. [2.5])

```
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
tftalebu@dk3n38 ~/work/arch-pc/lab08 $
```

Рис. 2.5: Рисунок

2.2 Обработка аргументов командной строки :

- На этом шаге мы создали файл lab8-2.asm, затем заполнили в нем наш код.(рис. [2.6])

```

1 %include 'in_out.asm'
2
3 SECTION .text
4 global _start
5
6 _start:
7 pop ecx
8 pop edx
9 sub ecx, 1
10 next:
11 cmp ecx, 0
12 jz _end
13
14 pop eax
15 call sprintLF
16 loop next
17
18 _end:
19 call quit

```

Рис. 2.6: Ресунок

- После этого мы скомпилировали файл и создали исполняемый файл.(рис. [2.7])

```

nasm: fatal: unable to open input file 'lab8-2.asm': no such file or directory
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ touch lab8-2.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ gedit lab8-2.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ./lab8-2
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
tftalebu@dk3n38 ~/work/arch-pc/lab08 $

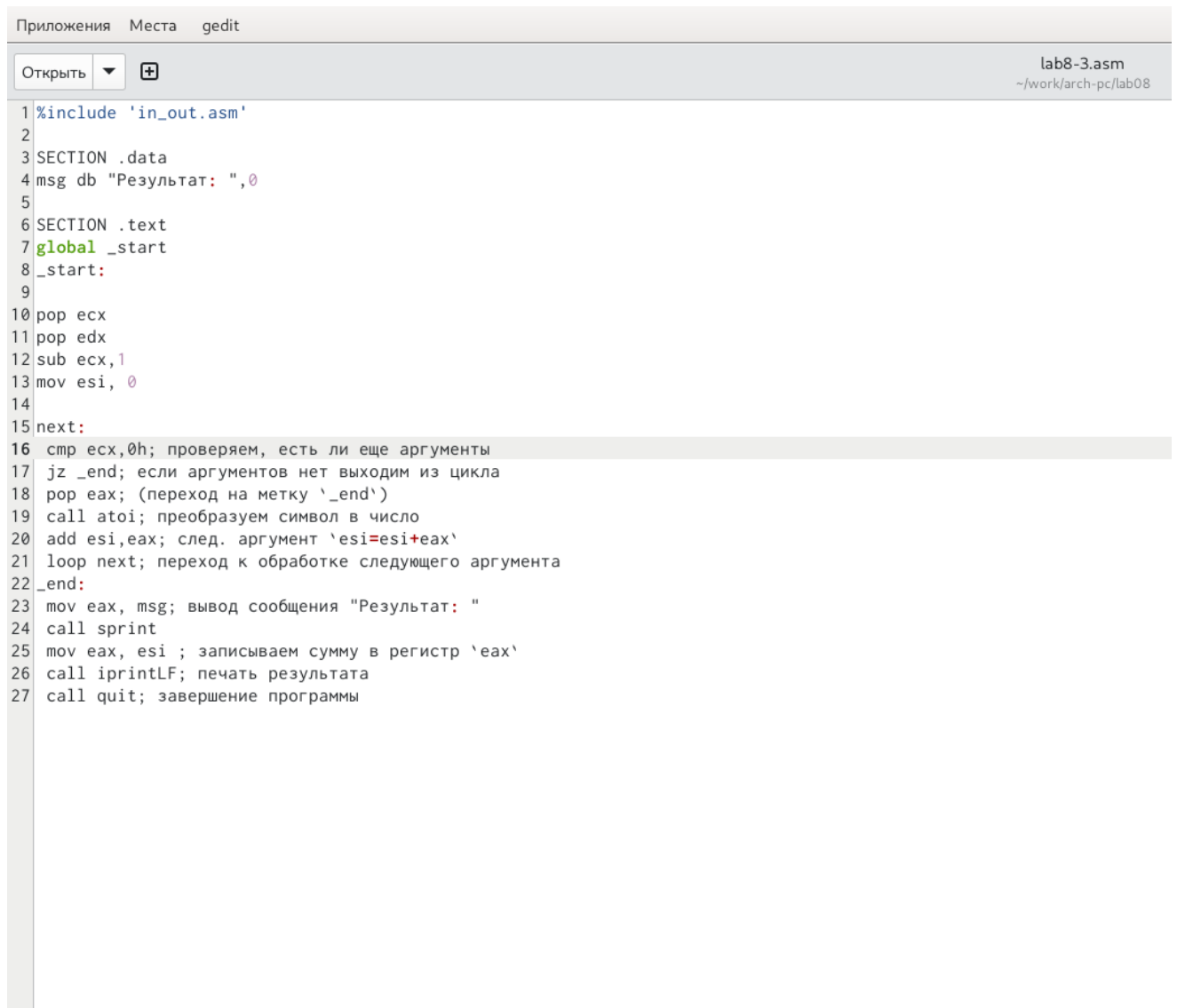
```

Рис. 2.7: Ресунок

- И, как вы можете видеть, на этот раз при запуске программы мы добавили в команду три аргумента, и в этом случае были обработаны три аргумента.

2.3 Программа вычисления суммы аргументов командной строки :

- Первым делом мы создали файл lab8-3.asm, затем заполнили кодом программы.(рис. [2.8])



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Результат: ",0
5
6 SECTION .text
7 global _start
8 _start:
9
10 pop ecx
11 pop edx
12 sub ecx,1
13 mov esi, 0
14
15 next:
16 cmp ecx,0h; проверяем, есть ли еще аргументы
17 jz _end; если аргументов нет выходим из цикла
18 pop eax; (переход на метку '_end')
19 call atoi; преобразуем символ в число
20 add esi,eax; след. аргумент 'esi=esi+eax'
21 loop next; переход к обработке следующего аргумента
22 _end:
23 mov eax, msg; вывод сообщения "Результат: "
24 call sprint
25 mov eax, esi ; записываем сумму в регистр 'eax'
26 call iprintLF; печать результата
27 call quit; завершение программы
```

Рис. 2.8: Ресунок

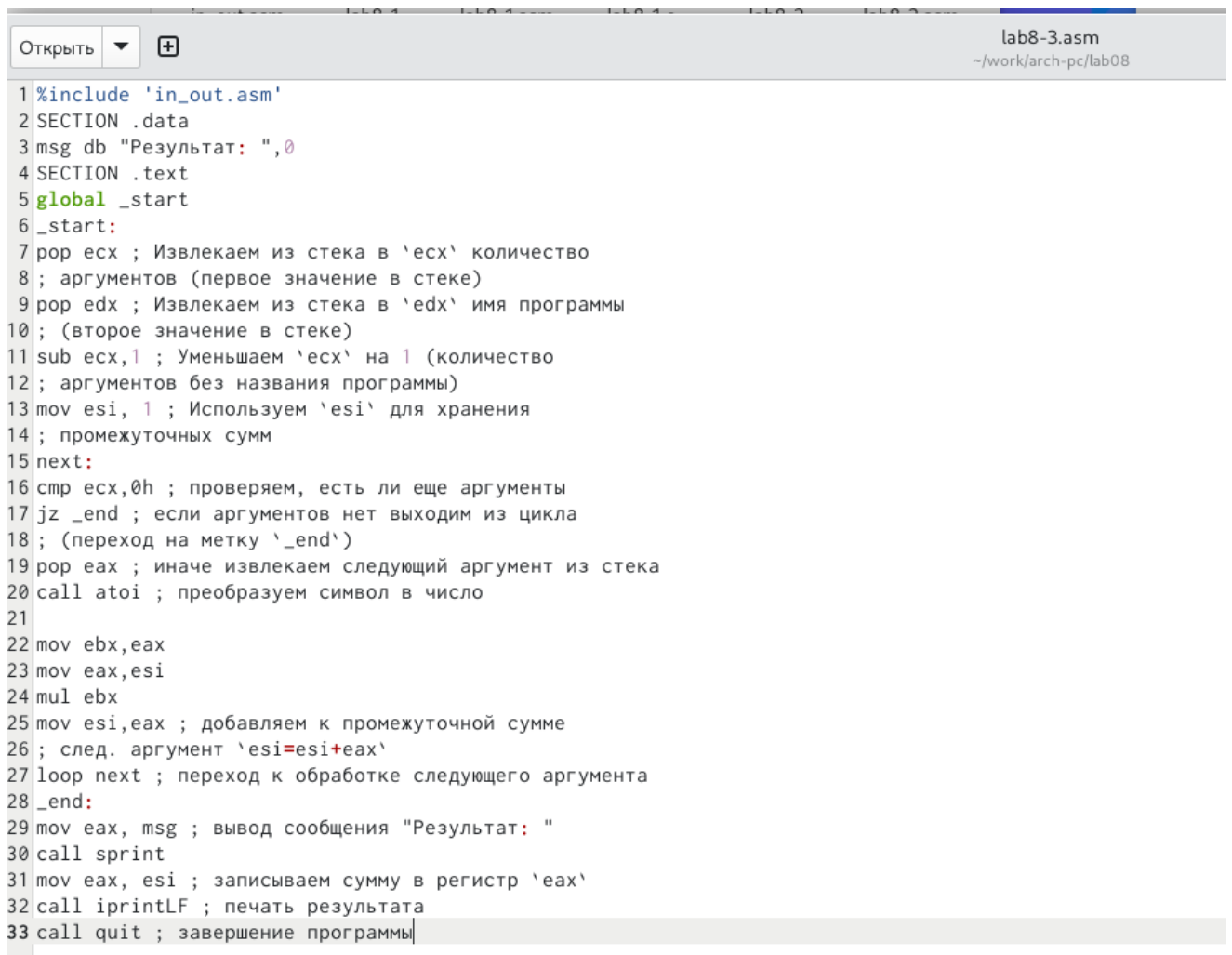
- После этого мы скомпилировали файл, затем создали исполняемый файл,

ввели нужное количество аргументов и запустили prgoram.(рис. [2.9])

```
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
tftalebu@dk3n38 ~/work/arch-pc/lab08 $
```

Рис. 2.9: Ресунок

- Затем мы изменили код, чтобы вычислить произведение аргументов командной строки.(рис. [2.10])



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

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

Рис. 2.10: Ресунок

- После этого е скомпилировал код и запустил исполняемый файл.(рис. [2.11])

```
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3 4 5
Результат: 120
tftalebu@dk3n38 ~/work/arch-pc/lab08 $
```

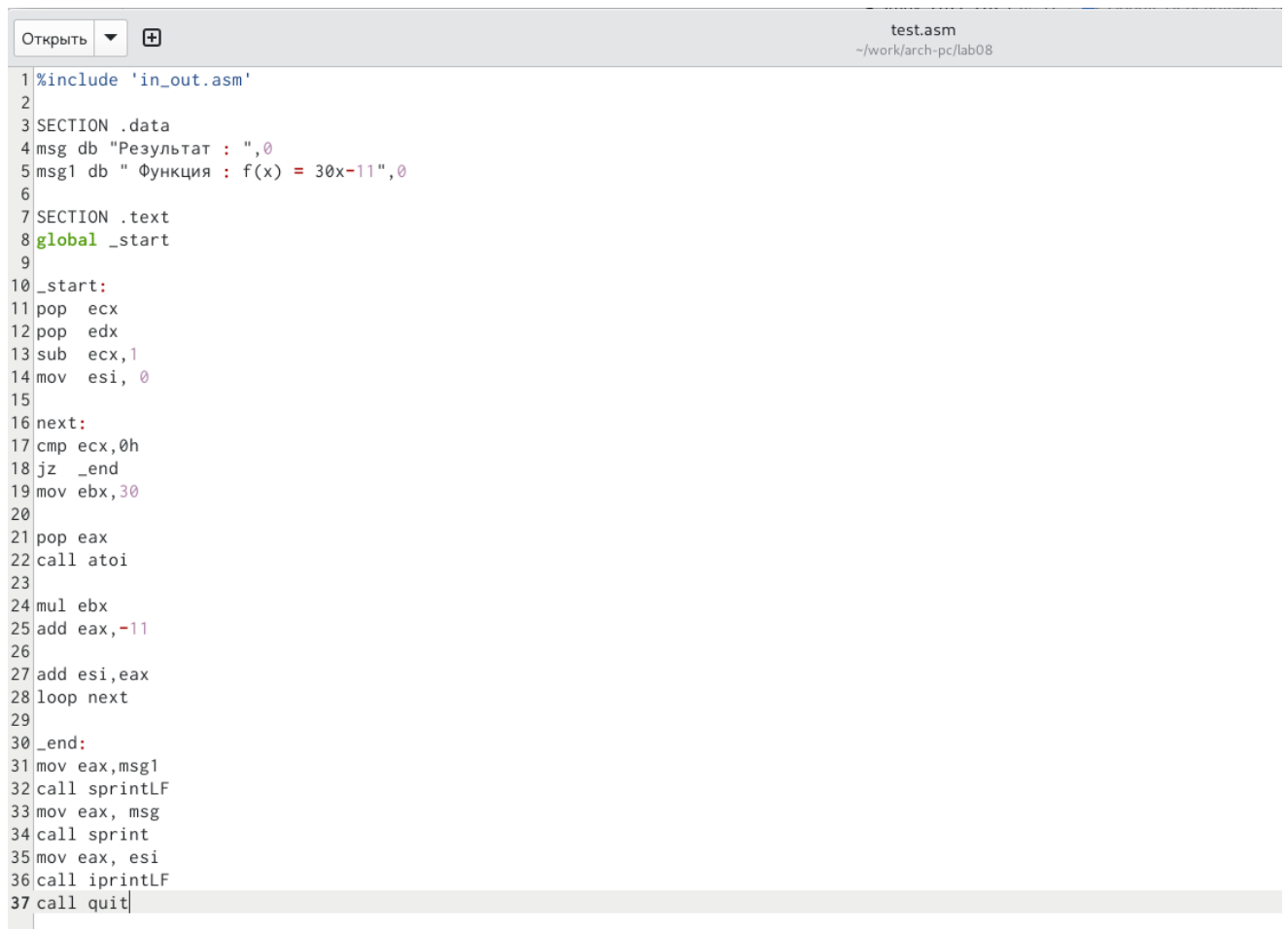
Рис. 2.11: Ресунок

2.4 Выводы по результатам выполнения заданий :

- В этой части работы мы узнали, как манипулировать циклами, как правильно использовать стек для написания программ

3 Задание для самостоятельной работы :

- В этой части мы должны были написать программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x$
- сначала мы создали наш файл `test.asm`, где будет находиться наш код, затем мы написали программу. (рис. [3.1])



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Результат : ",0
5 msg1 db " Функция : f(x) = 30x-11",0
6
7 SECTION .text
8 global _start
9
10 _start:
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 mov ebx,30
20
21 pop eax
22 call atoi
23
24 mul ebx
25 add eax,-11
26
27 add esi,eax
28 loop next
29
30 _end:
31 mov eax,msg1
32 call sprintfLF
33 mov eax, msg
34 call sprintf
35 mov eax, esi
36 call iprintLF
37 call quit
```

Рис. 3.1: Рисунок

- Затем мы протестировали нашу программу.(рис. [2.1])


```
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ touch test.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ gedit test.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ nasm -f elf test.asm
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ls -m elf_i386 -o test test.o
ls: невозможно получить доступ к 'elf_i386': Нет такого файла или каталога
ls: невозможно получить доступ к 'test': Нет такого файла или каталога
-rw-r--r-- 1 tftalebu 1360 ноя 27 14:14 test.o
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o test test.o
tftalebu@dk3n38 ~/work/arch-pc/lab08 $ ./test 1 2 3 4
Функция :  $f(x) = 30x - 11$ 
Результат : 256
tftalebu@dk3n38 ~/work/arch-pc/lab08 $
```

Рис. 3.2: Ресунок

3.1 Выводы по результатам выполнения заданий :

В этой части мы узнали, как вычислить сложную математическую операцию, которая имеет функции, используя циклы и стек.

4 Выводы, согласованные с целью работы :

- В 8 лабораторной работе мы узнали, как использовать циклы и стек в NASM.

Список литературы