

# **Отчёт по лабораторной работе №8**

**Дисциплина: Архитектура компьютера**

Вакутайпа Милдред

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>4</b>	<b>Выполнение самостоятельной работы</b>	<b>18</b>
<b>5</b>	<b>Выводы</b>	<b>22</b>
<b>6</b>	<b>Список литературы</b>	<b>23</b>

# Список иллюстраций

3.1	Рис 1	6
3.2	Рис 2	7
3.3	Рис 3	7
3.4	Рис 4	8
3.5	Рис 5	9
3.6	Рис 6	9
3.7	Рис 7	10
3.8	Рис 8	11
3.9	Рис 9	12
3.10	Рис 10	12
3.11	Рис 11	12
3.12	Рис 12	12
3.13	Рис 13	13
3.14	Рис 14	13
3.15	Рис 15	13
3.16	Рис 16	14
3.17	Рис 17	14
3.18	Рис 18	14
3.19	Рис 19	15
3.20	Рис 20	16
3.21	Рис 21	17
4.1	Рис 22	18
4.2	Рис 23	19
4.3	Рис 24	21

# 1 Цель работы

Цель лабораторной работы – приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

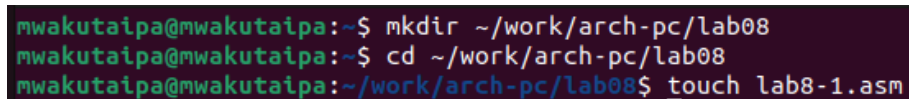
## 2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки

## 3 Выполнение лабораторной работы

### 1. Реализация циклов в NASM

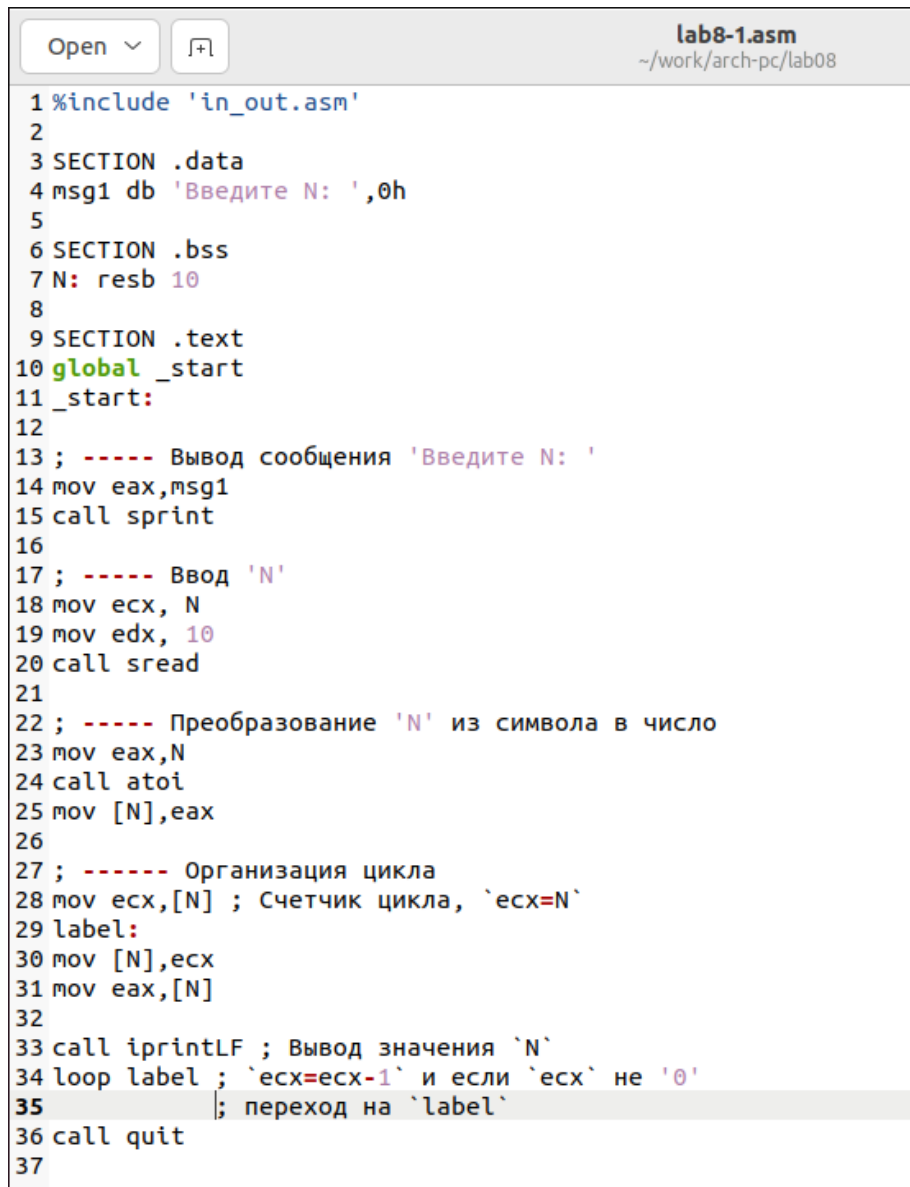
Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm:

A terminal window with a dark background and light-colored text. It shows three lines of commands being executed in a shell. The first line creates a directory, the second changes to that directory, and the third creates a new file.

```
mwakutaipa@mwakutaipa:~$ mkdir ~/work/arch-pc/lab08
mwakutaipa@mwakutaipa:~$ cd ~/work/arch-pc/lab08
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ touch lab8-1.asm
```

Рис. 3.1: Рис 1

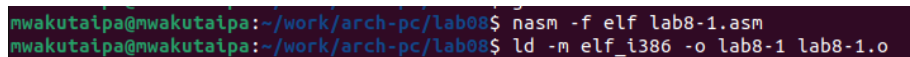
В файле lab8-1.asm вставляю код программы, которая показывает, как инструкция loop использует регистр ecx в качестве счетчика и на каждом шаге уменьшает его значение на единицу:



```
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 ; ----- Вывод сообщения 'Введите N: '
14 mov eax,msg1
15 call sprint
16
17 ; ----- Ввод 'N'
18 mov ecx, N
19 mov edx, 10
20 call sread
21
22 ; ----- Преобразование 'N' из символа в число
23 mov eax,N
24 call atoi
25 mov [N],eax
26
27 ; ----- Организация цикла
28 mov ecx,[N] ; Счетчик цикла, `ecx=N`
29 label:
30 mov [N],ecx
31 mov eax,[N]
32
33 call iprintLF ; Вывод значения `N`
34 loop label ; `ecx=ecx-1` и если `ecx` не `0`
35 ; переход на `label`
36 call quit
37
```

Рис. 3.2: Рис 2

Создаю исполняемый файл:



```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
```

Рис. 3.3: Рис 3

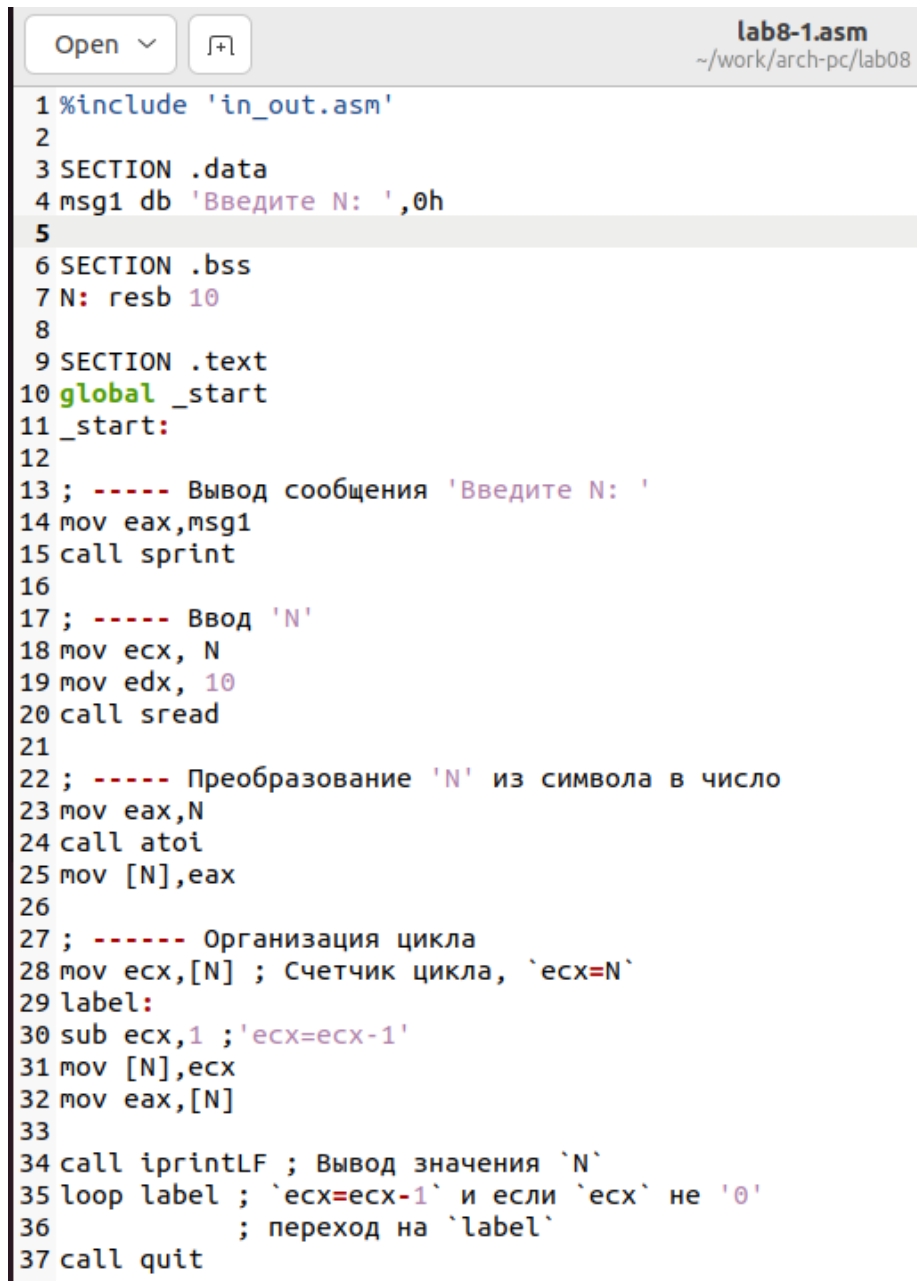
При запуске, программа выводит значение регистра ecx:

```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 7
7
6
5
4
3
2
1
```

Рис. 3.4: Рис 4

Изменяю текст программы, чтобы она показала, что использование регистра `esx` в теле цикла `loop` может привести к некорректной работе программы:

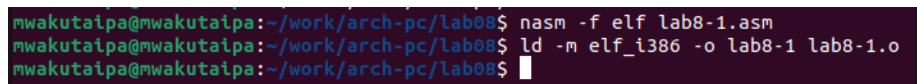




```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите N: ',0h
5
6 SECTION .bss
7 N: resb 10
8
9 SECTION .text
10 global _start
11 _start:
12
13 ; ----- Вывод сообщения 'Введите N: '
14 mov eax,msg1
15 call sprint
16
17 ; ----- Ввод 'N'
18 mov ecx, N
19 mov edx, 10
20 call sread
21
22 ; ----- Преобразование 'N' из символа в число
23 mov eax,N
24 call atoi
25 mov [N],eax
26
27 ; ----- Организация цикла
28 mov ecx,[N] ; Счетчик цикла, `ecx=N`
29 label:
30 sub ecx,1 ; `ecx=ecx-1`
31 mov [N],ecx
32 mov eax,[N]
33
34 call iprintLF ; Вывод значения `N`
35 loop label ; `ecx=ecx-1` и если `ecx` не `0`
36 ; переход на `label`
37 call quit
38
```

Рис. 3.5: Рис 5

Создаю исполняемый файл:



```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$
```

Рис. 3.6: Рис 6

При запуске, программа выводит бесконечное значение, которое не соответствует значению N введенному с клавиатуры:

```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 1
```

Рис. 3.7: Рис 7

4294940418  
4294940416  
4294940414  
4294940412  
4294940410  
4294940408  
4294940406  
4294940404  
4294940402  
4294940400  
4294940398  
4294940396  
4294940394  
4294940392  
4294940390  
4294940388  
4294940386  
4294940384  
4294940382  
4294940380  
42949

Рис. 3.8: Рис 8

Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Изменяю текст программы добавив команды `push` и `pop`:

```
29 label:
30 push ecx ; добавление значения ecx в стек
31 sub ecx,1
32 mov [N],ecx
33 mov eax,[N]
34 call iprintLF
35 pop ecx ; извлечение значения ecx из стека
36 loop label
37
38 call quit
```

Рис. 3.9: Рис 9

Создаю исполняемый файл:

```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$
```

Рис. 3.10: Рис 10

При запуске, программа выводит значение, которое соответствует значению `N` введенному с клавиатуры:

```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
```

Рис. 3.11: Рис 11

```
Введите N: 4
3
2
1
0
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$
```

Рис. 3.12: Рис 12

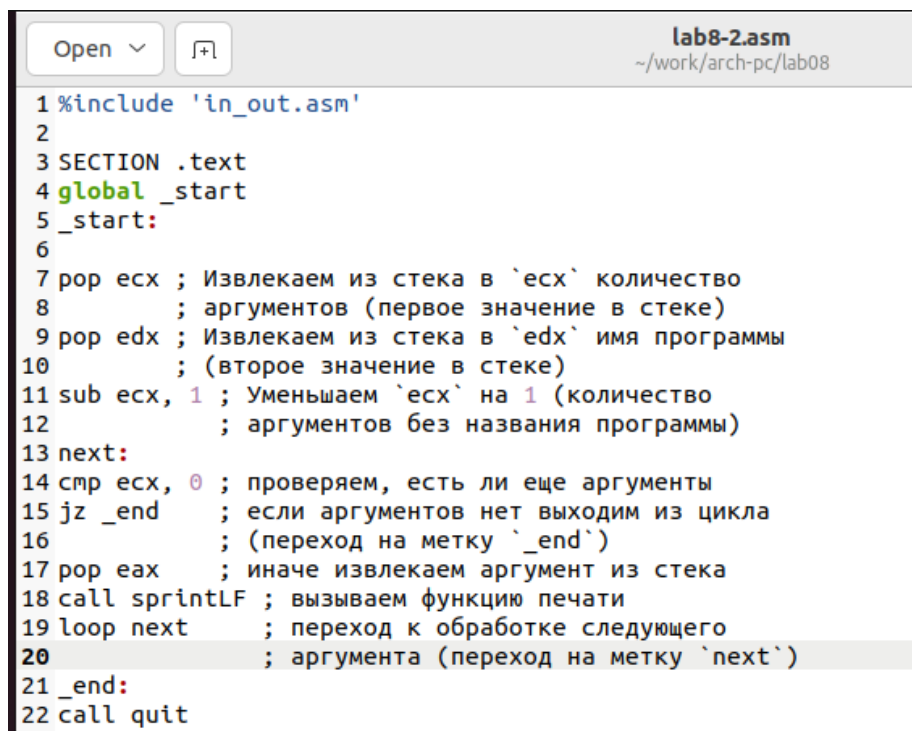
## 2. Обработка аргументов командной строки

Создаю файл lab8-2.asm:

```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ touch lab8-2.asm
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$
```

Рис. 3.13: Рис 13

Ввожу в него текст программы, которая выводит на экран аргументы командной строки:



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

1 %include 'in_out.asm'
2
3 SECTION .text
4 global _start
5 _start:
6
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 next:
14 cmp ecx, 0 ; проверяем, есть ли еще аргументы
15 jz _end    ; если аргументов нет выходим из цикла
16           ; (переход на метку `_end`)
17 pop eax    ; иначе извлекаем аргумент из стека
18 call printf ; вызываем функцию печати
19 loop next  ; переход к обработке следующего
20           ; аргумента (переход на метку `next`)
21 _end:
22 call quit
23
```

Рис. 3.14: Рис 14

Создаю исполняемый файл и запускаю его:

```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ./lab8-2 5 6 '1'
5
6
1
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ./lab8-2 5 b '1'
5
b
1
```

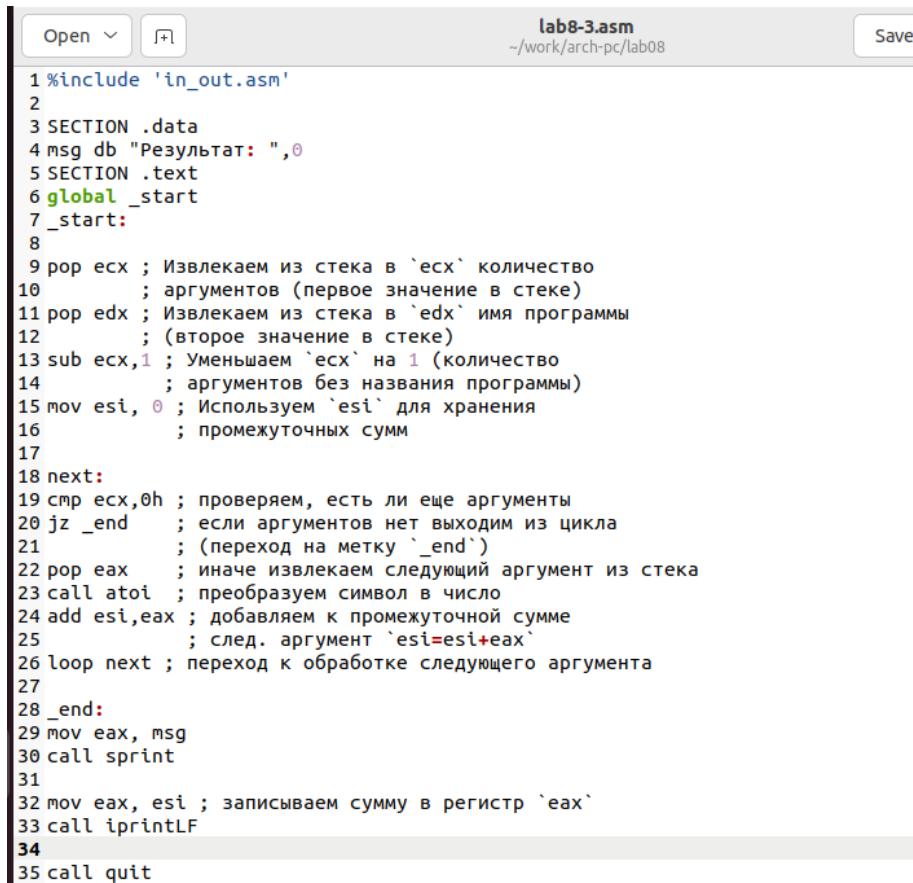
Рис. 3.15: Рис 15

Создаю файл lab8-3.asm:

```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ touch lab8-3.asm
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$
```

Рис. 3.16: Рис 16

Ввожу в него текст программы, которая выводит на экран сумму аргументов:



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

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

Рис. 3.17: Рис 17

Создаю исполняемый файл и запускаю его:

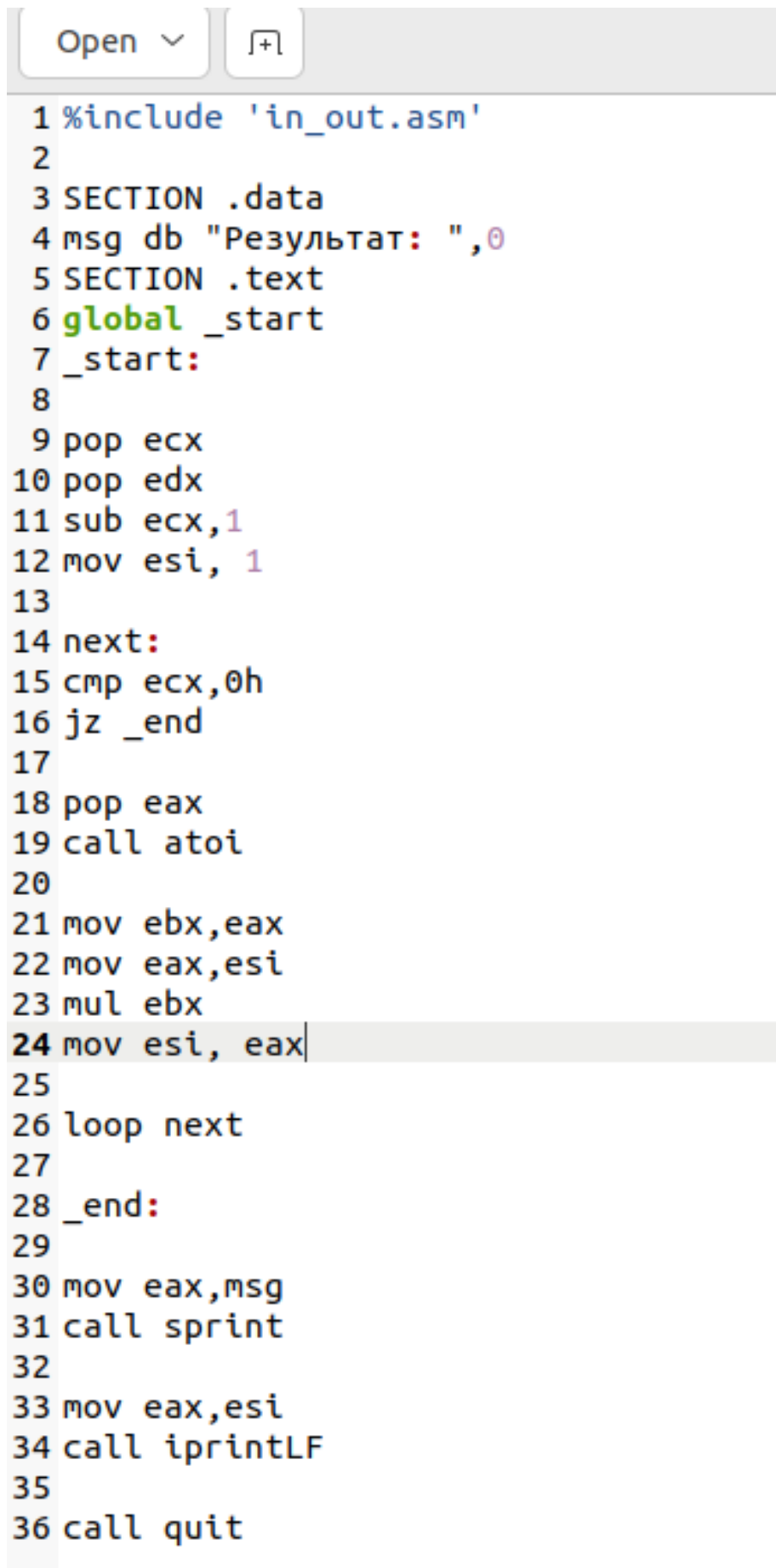
```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ gcc -c lab8-3.asm
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
```

Рис. 3.18: Рис 18

```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ./lab8-3 7 3 8
Результат: 18
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$
```

Рис. 3.19: Рис 19

Изменяю текст программы, чтобы она выводила произведение аргументов:



```
Open  ▾  [ ]
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Результат: ",0
5 SECTION .text
6 global _start
7 _start:
8
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
18 pop eax
19 call atoi
20
21 mov ebx,eax
22 mov eax,esi
23 mul ebx
24 mov esi, eax
25
26 loop next
27
28 _end:
29
30 mov eax,msg
31 call sprint
32
33 mov eax,esi
34 call iprintLF
35
36 call quit
```

Рис. 3.20: Рис 20



Создаю исполняемый файл и запускаю его:

```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ./lab8-3 4 3
Результат: 12
```

Рис. 3.21: Рис 21

## 4 Выполнение самостоятельной работы

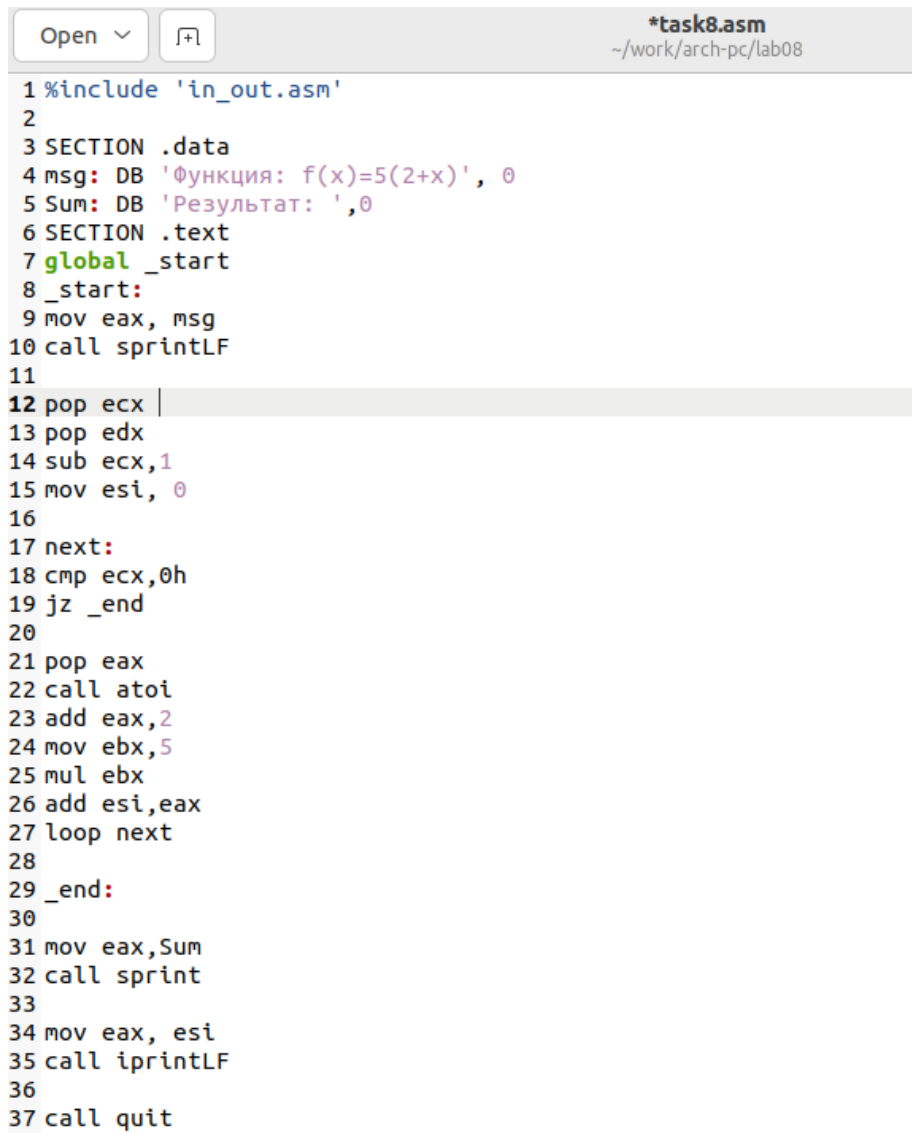
Создаю файл task8.asm:



```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ touch task8.asm
```

Рис. 4.1: Рис 22

В него пишу программу, которая находит сумму значений функции  $f(x) = 5(2 + x)$  для некоторых значений  $x$  (вариант 10):



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg: DB 'Функция: f(x)=5(2+x)', 0
5 Sum: DB 'Результат: ', 0
6 SECTION .text
7 global _start
8 _start:
9 mov eax, msg
10 call sprintf
11
12 pop ecx
13 pop edx
14 sub ecx, 1
15 mov esi, 0
16
17 next:
18 cmp ecx, 0h
19 jz _end
20
21 pop eax
22 call atoi
23 add eax, 2
24 mov ebx, 5
25 mul ebx
26 add esi, eax
27 loop next
28
29 _end:
30
31 mov eax, Sum
32 call sprintf
33
34 mov eax, esi
35 call iprintf
36
37 call quit
```

Рис. 4.2: Рис 23

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg: DB 'Функция: f(x)=5(2+x)', 0
```

```
Sum: DB 'Результат: ', 0
```

```
SECTION .text
```

```
global _start
```

```
_start:  
mov eax, msg  
call sprintLF
```

```
pop ecx  
pop edx  
sub ecx,1  
mov esi, 0
```

```
next:  
cmp ecx,0h  
jz _end
```

```
pop eax  
call atoi  
add eax,2  
mov ebx,5  
mul ebx  
add esi,eax  
loop next
```

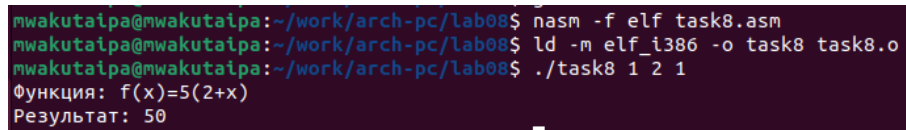
```
_end:
```

```
mov eax,Sum  
call sprint
```

```
mov eax, esi  
call iprintLF
```

```
call quit
```

Создаю исполняемый файл и запускаю его. Программа выводит сумму  $f(1)+f(2)+f(1)$ :



```
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ nasm -f elf task8.asm
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ld -m elf_i386 -o task8 task8.o
mwakutaipa@mwakutaipa:~/work/arch-pc/lab08$ ./task8 1 2 1
Функция: f(x)=5(2+x)
Результат: 50
```

Рис. 4.3: Рис 24

## 5 Выводы

При выполнении данной работы я освоила использование циклов и обработку аргументов командной строки.

## 6 Список литературы

Архитектура ЭВМ