

# **Отчет по лабораторной работе №8**

**Дисциплина: архитектура компьютера**

Мазурский Александр Дмитриевич

# 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>
4.1	Реализация циклов в NASM . . . . .	8
4.2	Обработка аргументов командной строки . . . . .	13
4.3	Задание для самостоятельной работы . . . . .	17
<b>5</b>	<b>Выводы</b>	<b>20</b>
<b>6</b>	<b>Список литературы</b>	<b>21</b>

## Список иллюстраций

4.1	Создание каталога . . . . .	8
4.2	Копирование программы из листинга . . . . .	9
4.3	Запуск программы . . . . .	9
4.4	Изменение программы . . . . .	10
4.5	Запуск измененной программы . . . . .	11
4.6	Добавление push и pop в цикл программы . . . . .	12
4.7	Запуск измененной программы . . . . .	13
4.8	Копирование программы из листинга . . . . .	13
4.9	Запуск второй программы . . . . .	14
4.10	Копирование программы из третьего листинга . . . . .	15
4.11	Запуск третьей программы . . . . .	15
4.12	Изменение третьей программы . . . . .	16
4.13	Запуск измененной третьей программы . . . . .	16
4.14	Написание программы для самостоятельной работы . . . . .	17
4.15	Запуск программы для самостоятельной работы . . . . .	19

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. Самостоятельное написание программы по материалам лабораторной работы

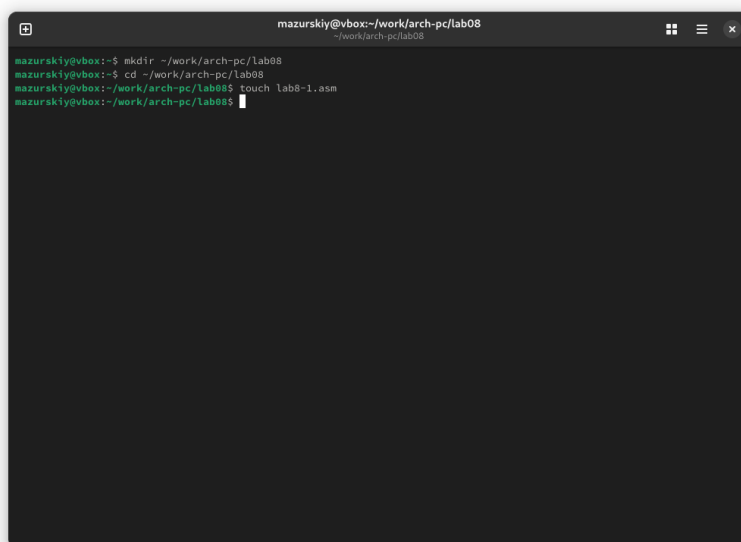
### 3 Теоретическое введение

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.

## 4 Выполнение лабораторной работы

### 4.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8 (рис. 4.1).

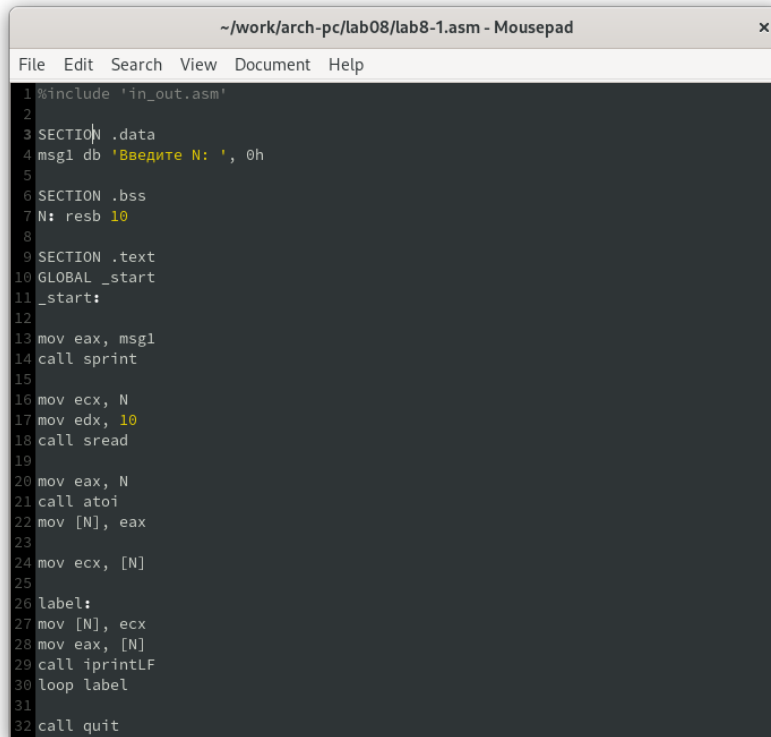


```
mazurskiy@vbox:~/work/arch-pc/lab08
mazurskiy@vbox:~$ mkdir ~/work/arch-pc/lab08
mazurskiy@vbox:~$ cd ~/work/arch-pc/lab08
mazurskiy@vbox:~/work/arch-pc/lab08$ touch lab8-1.asm
mazurskiy@vbox:~/work/arch-pc/lab08$
```

Рис. 4.1: Создание каталога

Копирую в созданный файл программу из листинга. (рис. 4.2).

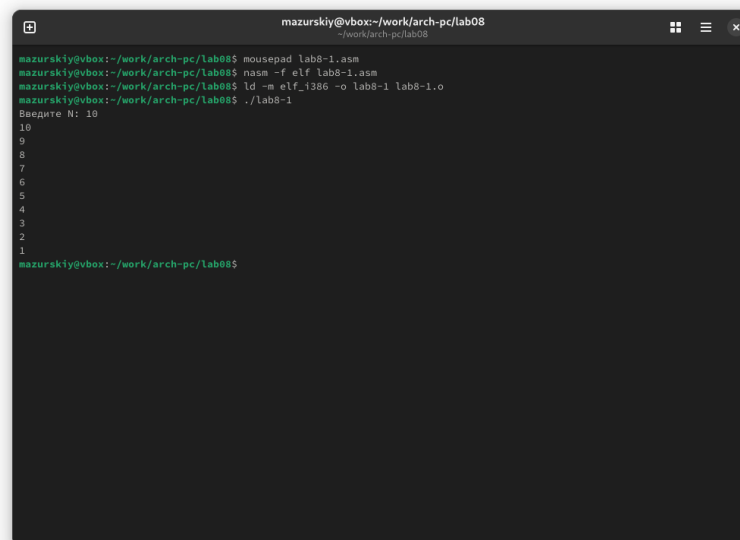




```
~ /work/arch-pc/lab08/lab8-1.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите N: ', 0h
5
6 SECTION .bss
7 N: resb 10
8
9 SECTION .text
10 GLOBAL _start
11 _start:
12
13 mov eax, msg1
14 call sprint
15
16 mov ecx, N
17 mov edx, 10
18 call sread
19
20 mov eax, N
21 call atoi
22 mov [N], eax
23
24 mov ecx, [N]
25
26 label:
27 mov [N], ecx
28 mov eax, [N]
29 call iprintLF
30 loop label
31
32 call quit
```

Рис. 4.2: Копирование программы из листинга

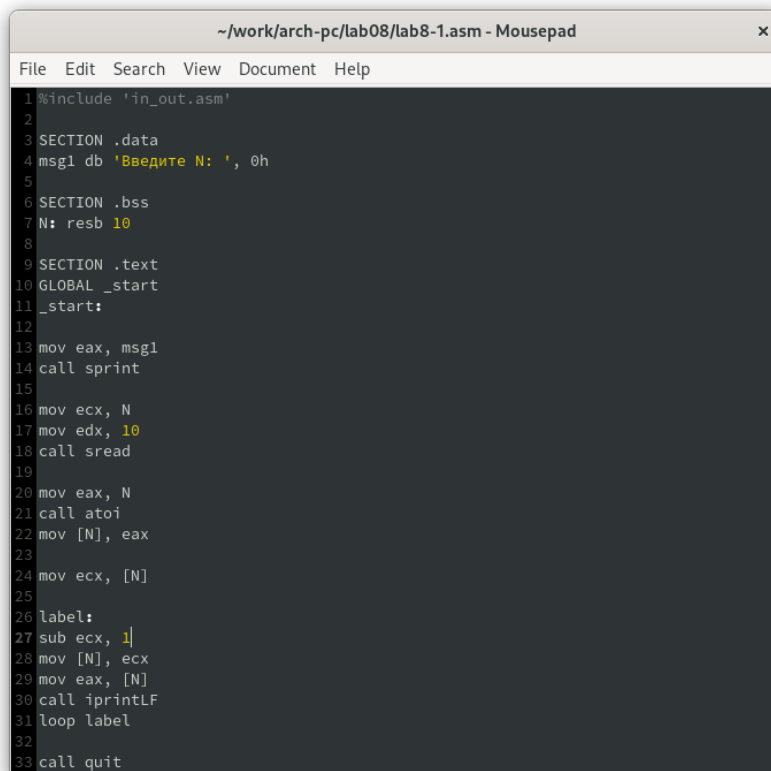
Запускаю программу, она показывает работу циклов в NASM (рис. 4.3).



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

Рис. 4.3: Запуск программы

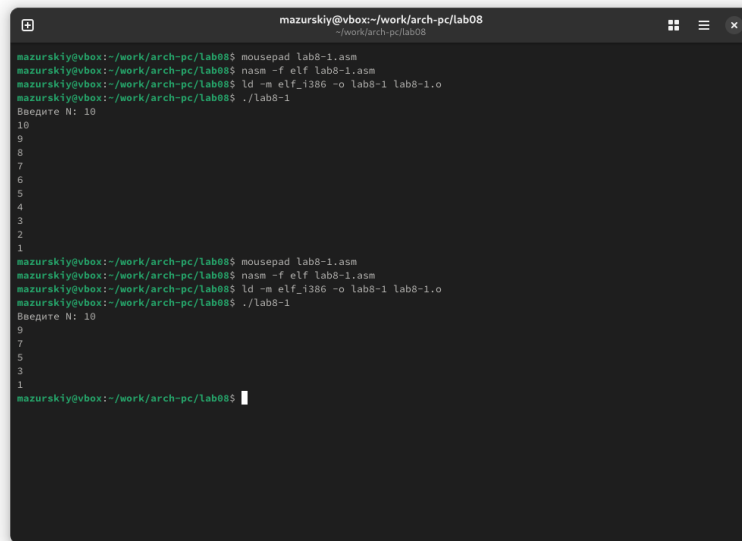
Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра `ecx` (рис. 4.4).



```
~/work/arch-pc/lab08/lab8-1.asm - Mousepad
File Edit Search View Document Help
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 mov eax, msg1
14 call sprint
15
16 mov ecx, N
17 mov edx, 10
18 call sread
19
20 mov eax, N
21 call atoi
22 mov [N], eax
23
24 mov ecx, [N]
25
26 label:
27 sub ecx, 1
28 mov [N], ecx
29 mov eax, [N]
30 call iprintLF
31 loop label
32
33 call quit
```

Рис. 4.4: Изменение программы

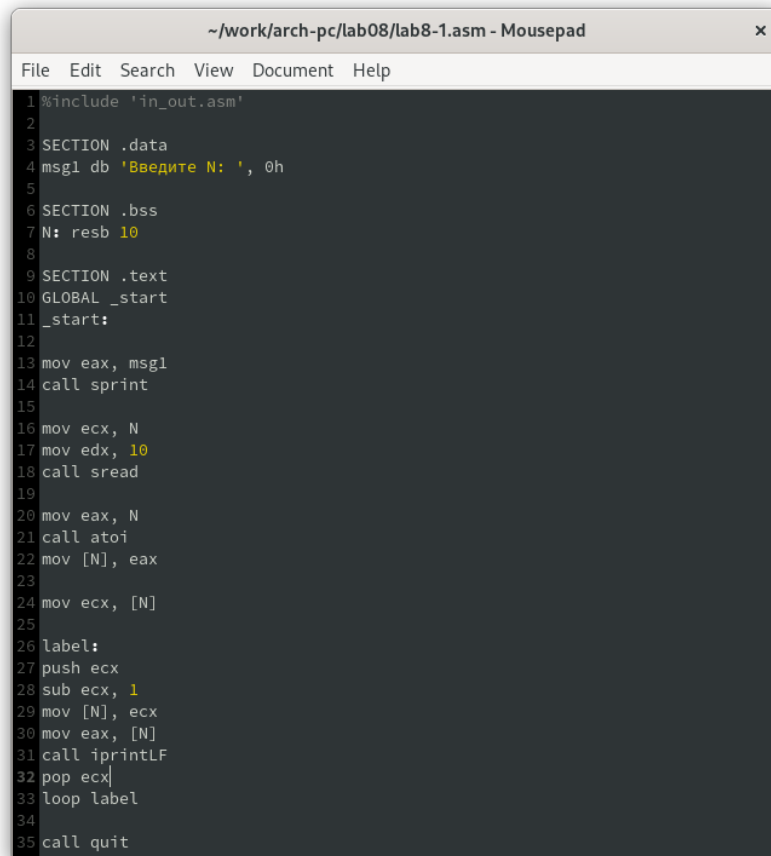
Из-за того, что теперь регистр `ecx` на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. 4.5).



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

Рис. 4.5: Запуск измененной программы

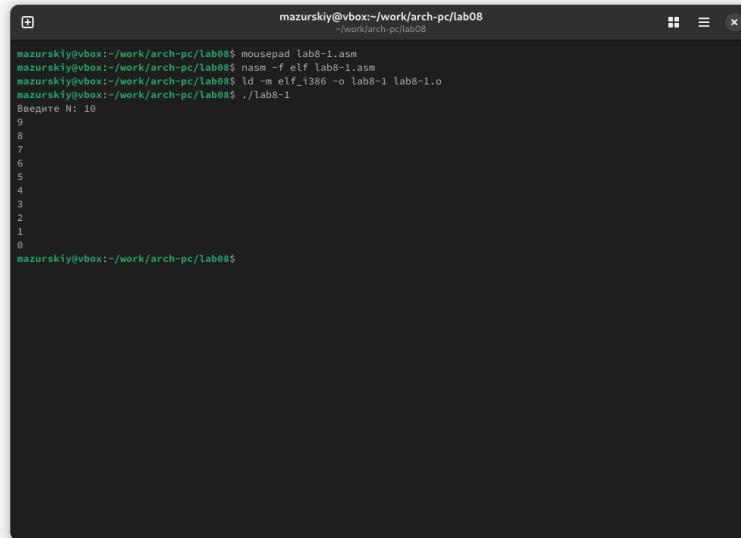
Добавляю команды push и pop в программу (рис. 4.6).

A screenshot of a text editor window titled '~\work\arch-pc\lab08\lab8-1.asm - Mousepad'. The window contains assembly code for a loop. The code starts with a menu bar (File, Edit, Search, View, Document, Help) and a line number column from 1 to 35. The code includes a macro call, section declarations for .data, .bss, and .text, and a loop that reads integers from memory and prints them. The loop body includes a push instruction, a decrement of ecx, a move from memory to eax, a print instruction, a pop instruction, and a loop instruction. The code ends with a quit instruction.

```
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 mov eax, msg1
14 call sprint
15
16 mov ecx, N
17 mov edx, 10
18 call sread
19
20 mov eax, N
21 call atoi
22 mov [N], eax
23
24 mov ecx, [N]
25
26 label:
27 push ecx
28 sub ecx, 1
29 mov [N], ecx
30 mov eax, [N]
31 call iprintLF
32 pop ecx
33 loop label
34
35 call quit
```

Рис. 4.6: Добавление push и pop в цикл программы

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 (рис. 4.7).

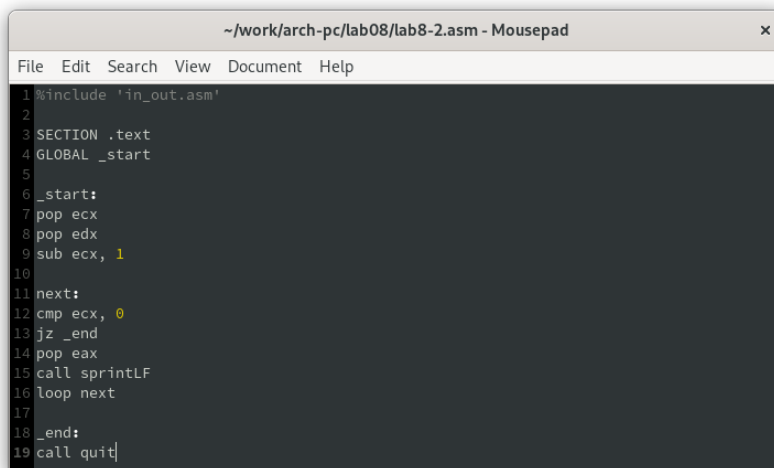


```
mazurskiy@vbox:~/work/arch-pc/lab08$ mousepad lab8-1.asm
mazurskiy@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
mazurskiy@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mazurskiy@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 19
9
8
7
6
5
4
3
2
1
0
mazurskiy@vbox:~/work/arch-pc/lab08$
```

Рис. 4.7: Запуск измененной программы

## 4.2 Обработка аргументов командной строки

Создаю новый файл для программы и копирую в него код из следующего листинга (рис. 4.8).

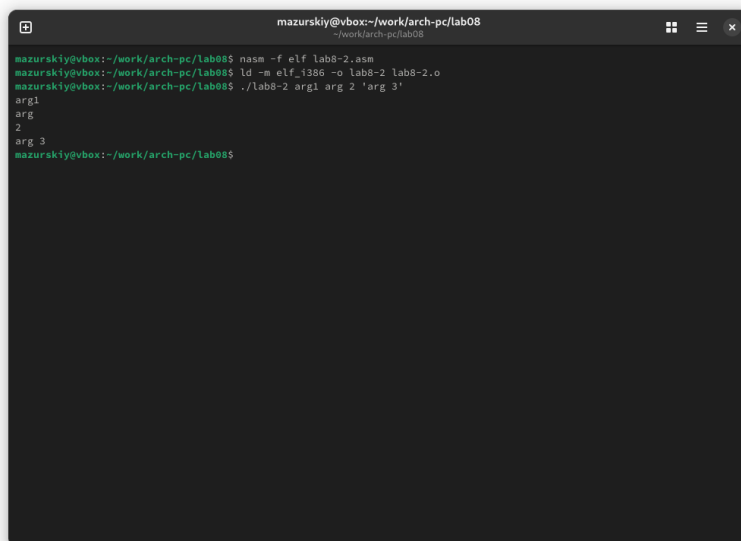


```
~/work/arch-pc/lab08/lab8-2.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5
6 _start:
7 pop ecx
8 pop edx
9 sub ecx, 1
10
11 next:
12 cmp ecx, 0
13 jz _end
14 pop eax
15 call sprintf
16 loop next
17
18 _end:
19 call quit
```

Рис. 4.8: Копирование программы из листинга

Компилирую программу и запускаю, указав аргументы. Программой было об-

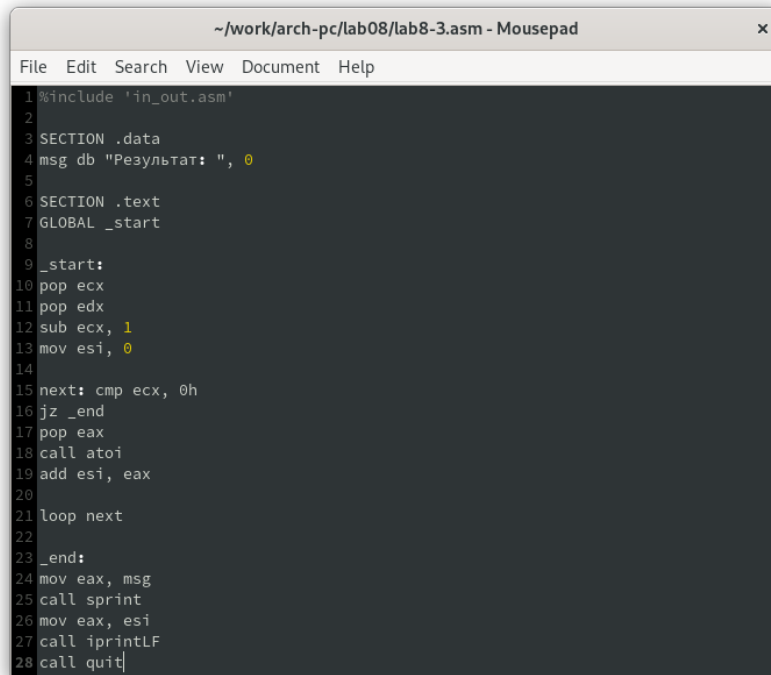
ратоно то же количество аргументов, что и было введено (рис. 4.9).



```
mazurskiy@vbox:~/work/arch-pc/lab08
mazurskiy@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
mazurskiy@vbox:~/work/arch-pc/lab08$ ld -o elf386 -o lab8-2 lab8-2.o
mazurskiy@vbox:~/work/arch-pc/lab08$ ./lab8-2 arg1 arg 2 'arg 3'
arg1
arg 2
arg 3
mazurskiy@vbox:~/work/arch-pc/lab08$
```

Рис. 4.9: Запуск второй программы

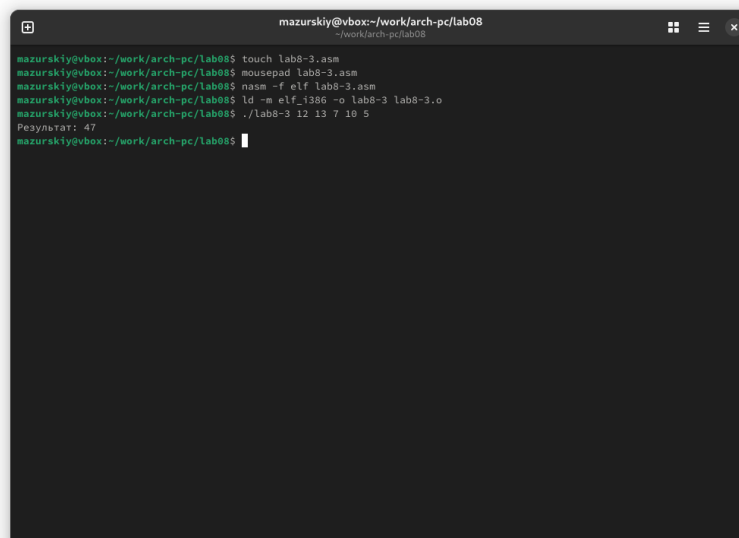
Создаю новый файл для программы и копирую в него код из третьего листинга (рис. 4.10).



```
~/work/arch-pc/lab08/lab8-3.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Результат: ", 0
5
6 SECTION .text
7 GLOBAL _start
8
9 _start:
10 pop ecx
11 pop edx
12 sub ecx, 1
13 mov esi, 0
14
15 next: cmp ecx, 0h
16 jz _end
17 pop eax
18 call atoi
19 add esi, eax
20
21 loop next
22
23 _end:
24 mov eax, msg
25 call sprint
26 mov eax, esi
27 call iprintLF
28 call quit
```

Рис. 4.10: Копирование программы из третьего листинга

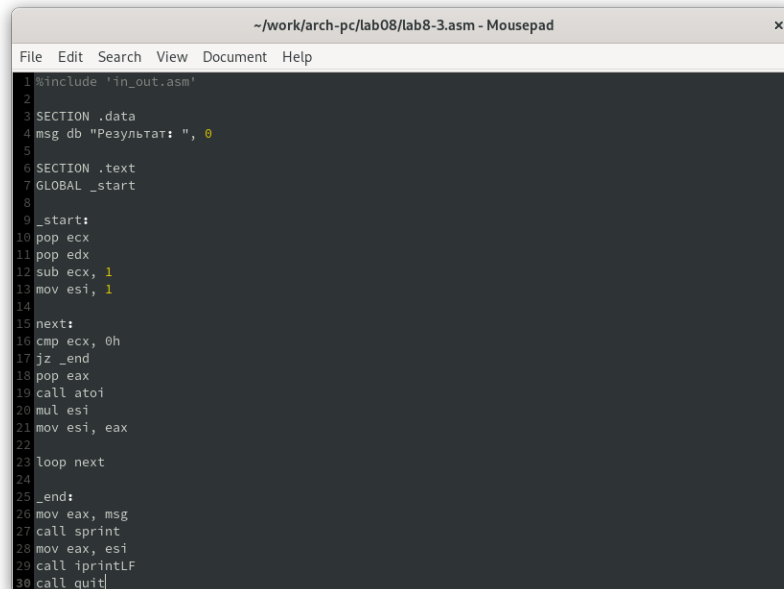
Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. 4.11).



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

Рис. 4.11: Запуск третьей программы

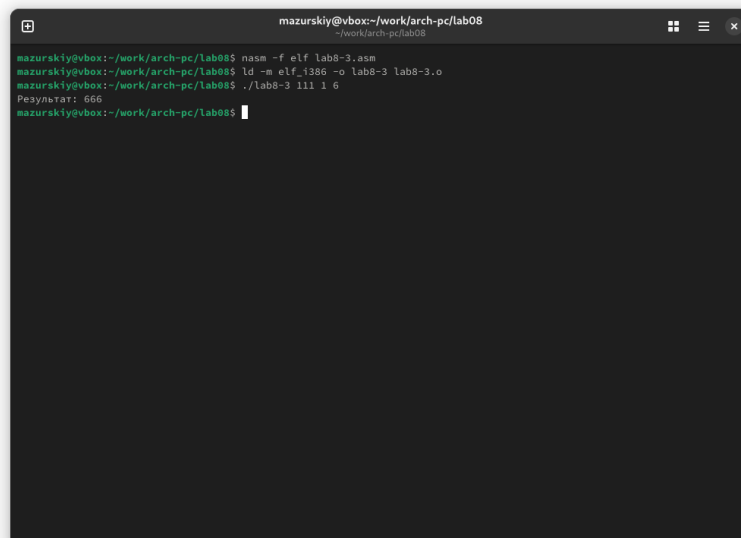
Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. 4.12).



```
~/work/arch-pc/lab08/lab8-3.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Результат: ", 0
5
6 SECTION .text
7 GLOBAL _start
8
9 _start:
10 pop ecx
11 pop edx
12 sub ecx, 1
13 mov esi, 1
14
15 next:
16 cmp ecx, 0h
17 jz _end
18 pop eax
19 call atoi
20 mul esi
21 mov esi, eax
22
23 loop next
24
25 _end:
26 mov eax, msg
27 call sprint
28 mov eax, esi
29 call iprintLF
30 call quit
```

Рис. 4.12: Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. 4.13).



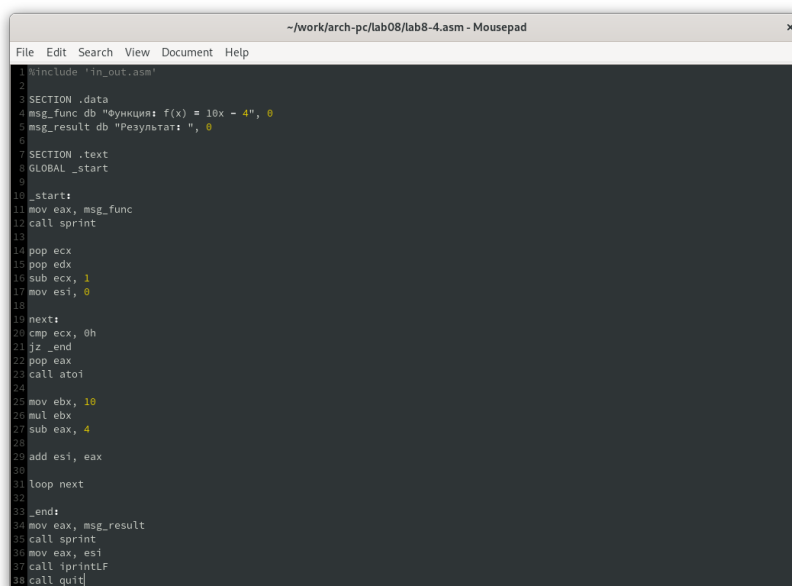
```
mazurskiy@vbox:~/work/arch-pc/lab08
~/work/arch-pc/lab08
mazurskiy@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
mazurskiy@vbox:~/work/arch-pc/lab08$ ld -s elf386 -o lab8-3 lab8-3.o
mazurskiy@vbox:~/work/arch-pc/lab08$ ./lab8-3 111 1 6
Результат: 666
mazurskiy@vbox:~/work/arch-pc/lab08$
```

Рис. 4.13: Запуск измененной третьей программы



## 4.3 Задание для самостоятельной работы

Пишу программу, которая будет находить сумма значений для функции  $f(x) = 10x - 4$ , которая совпадает с моим девытым варинтом (рис. 4.14).



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

Рис. 4.14: Написание программы для самостоятельной работы

Код программы:

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 10x - 4", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
```

```

call sprintf

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

next:
cmp ecx, 0h
jz _end
pop eax
call atoi

mov ebx, 10
mul ebx
sub eax, 4

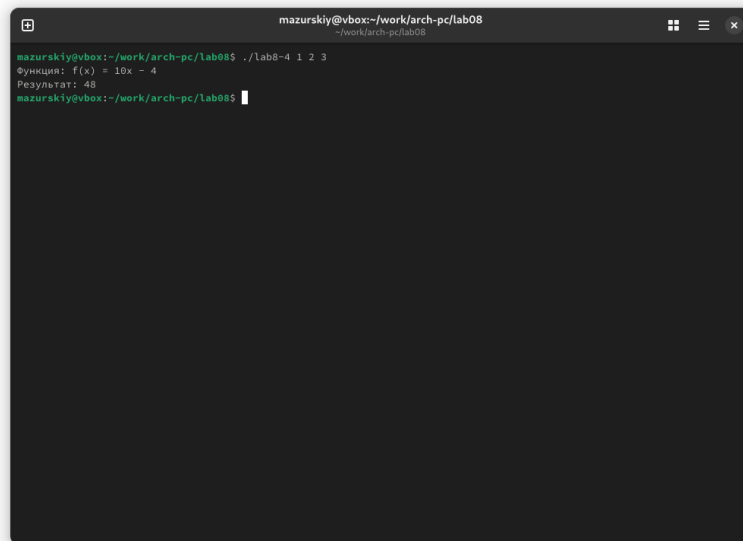
add esi, eax

loop next

_end:
mov eax, msg_result
call sprintf
mov eax, esi
call iprintLF
call quit

```

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. 4.15).

A terminal window with a dark background and light text. The title bar at the top reads 'mazurskiy@vbox: ~/work/arch-pc/lab08'. The terminal content shows a command prompt 'mazurskiy@vbox:~/work/arch-pc/lab08\$' followed by the command './lab8-4 1 2 3'. The output consists of two lines: 'Функция: f(x) = 10x - 4' and 'Результат: 46'. The prompt is followed by a cursor.

```
mazurskiy@vbox:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
Функция: f(x) = 10x - 4
Результат: 46
mazurskiy@vbox:~/work/arch-pc/lab08$
```

Рис. 4.15: Запуск программы для самостоятельной работы

## **5 Выводы**

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием циклов а также научился обрабатывать аргументы командной строки.

## **6 Список литературы**

1. Курс на ТУИС
2. Лабораторная работа №8
3. Программирование на языке ассемблера NASM Столяров А. В.