

# **Отчёт по лабораторной работе №8**

**Дисциплина: архитектура компьютера**

**Панявкина Ирина Васильевна**

**НКАбд-04-24**

# Содержание

1 Цель работы.....	3
2 Задание.....	3
3 Теоретическое введение.....	3
4 Выполнение лабораторной работы.....	4
4.1 Реализация циклов в NASM.....	4
4.2 Обработка аргументов командной строки.....	11
4.3    Выполнение заданий для самостоятельной работы.....	15
5 Выводы.....	17
Список литературы.....	18

# 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.

Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции:

- добавление элемента в вершину стека (push);
- извлечение элемента из вершины стека (pop).

## 4 Выполнение лабораторной работы

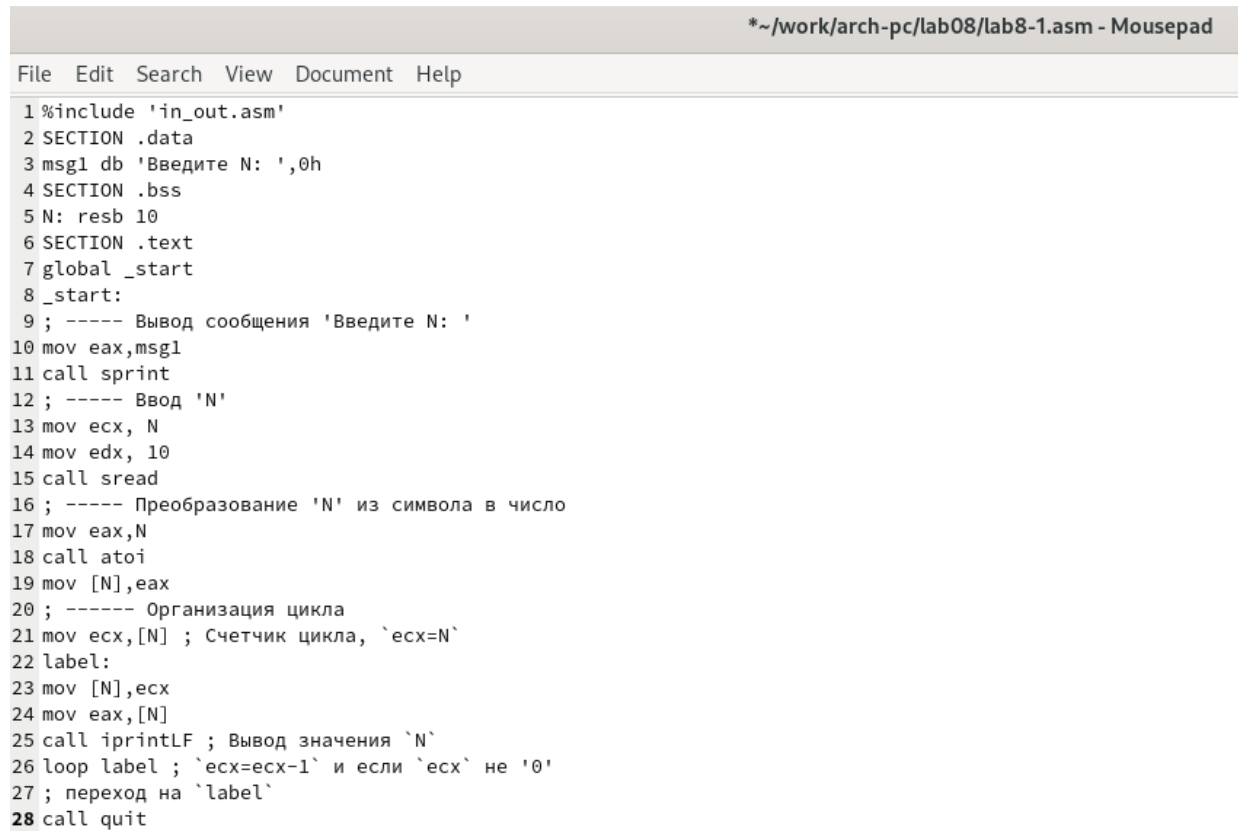
### 4.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8, а также файл lab8-1.asm и копирую в текущий каталог файл in\_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. 4.1).

```
irina_panyavkina@vbox:~$ mkdir ~/work/arch-pc/lab08
irina_panyavkina@vbox:~$ cd ~/work/arch-pc/lab08
irina_panyavkina@vbox:~/work/arch-pc/lab08$ touch lab8-1.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ls
lab8-1.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ cp ~/Downloads/in_out.asm in_out.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ls
in_out.asm  lab8-1.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ □
```

*Рис. 4.1: Создание каталога и файла для программы, создание копии внешнего файла*

Открываю созданный файл и вставляю в него скопированную программу из листинга (рис. 4.2).

A screenshot of a text editor window titled '\*~/work/arch-pc/lab08/lab8-1.asm - Mousepad'. The window has a menu bar with 'File', 'Edit', 'Search', 'View', 'Document', and 'Help'. The main text area contains assembly code with line numbers from 1 to 28. The code includes directives like '%include', 'SECTION', and 'global', as well as instructions like 'mov', 'call', 'sprint', 'sread', 'atoi', 'iprintLF', and 'quit'. Comments in Russian are interspersed throughout the code.

```
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
```

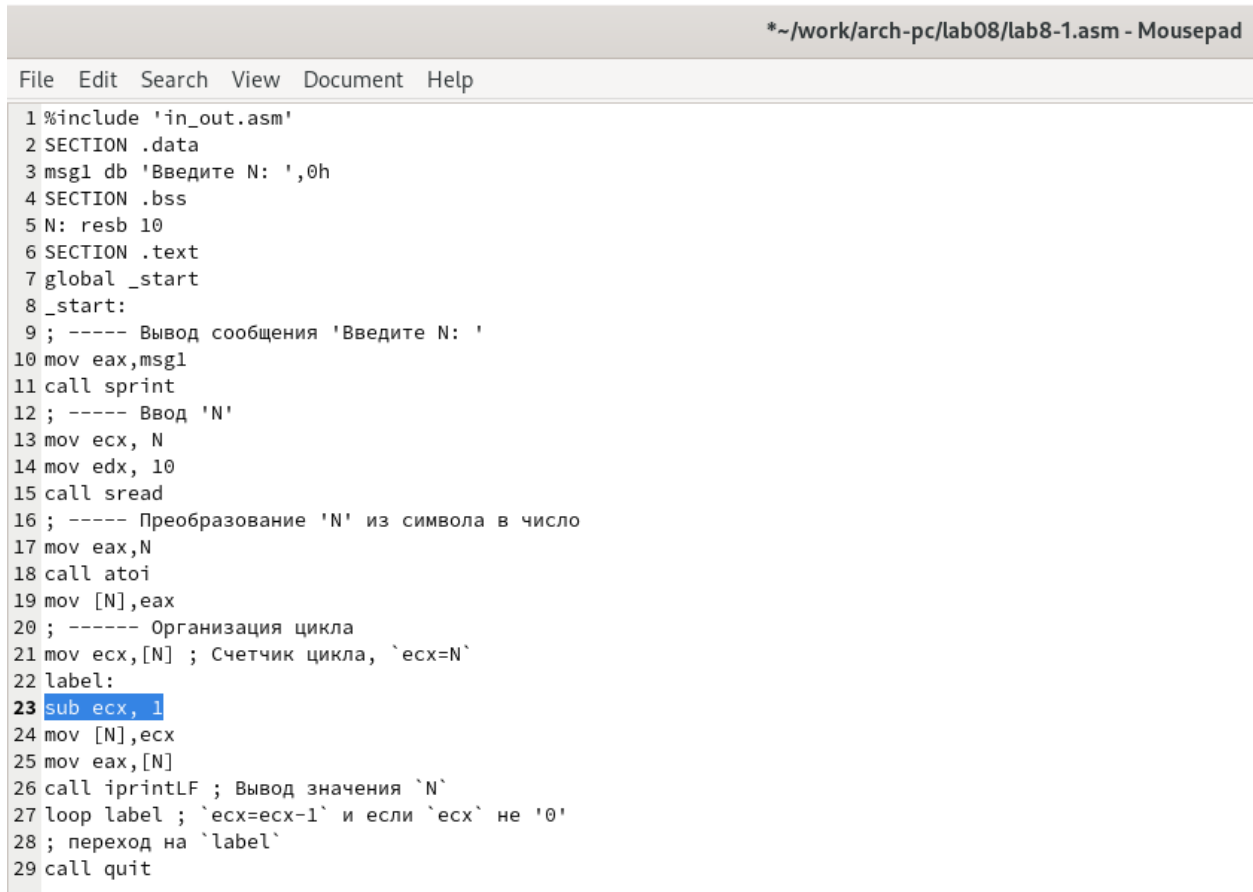
*Рис. 4.2: Редактирование файла и сохранение программы*

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

```
irina_panyavkina@vbox:~/work/arch-pc/lab08$ mousepad lab8-1.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 12
12
11
10
9
8
7
6
5
4
3
2
1
```

Рис. 4.3: Запуск исполняемого файла

Изменяю программу таким образом: в теле цикла я изменяю значение регистра `ecx` (рис. 4.4).



```
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
24 mov [N],ecx
25 mov eax,[N]
26 call iprintLF ; Вывод значения `N`
27 loop label ; `ecx=ecx-1` и если `ecx` не '0'
28 ; переход на `label`
29 call quit
```

Рис. 4.4: Редактирование файла

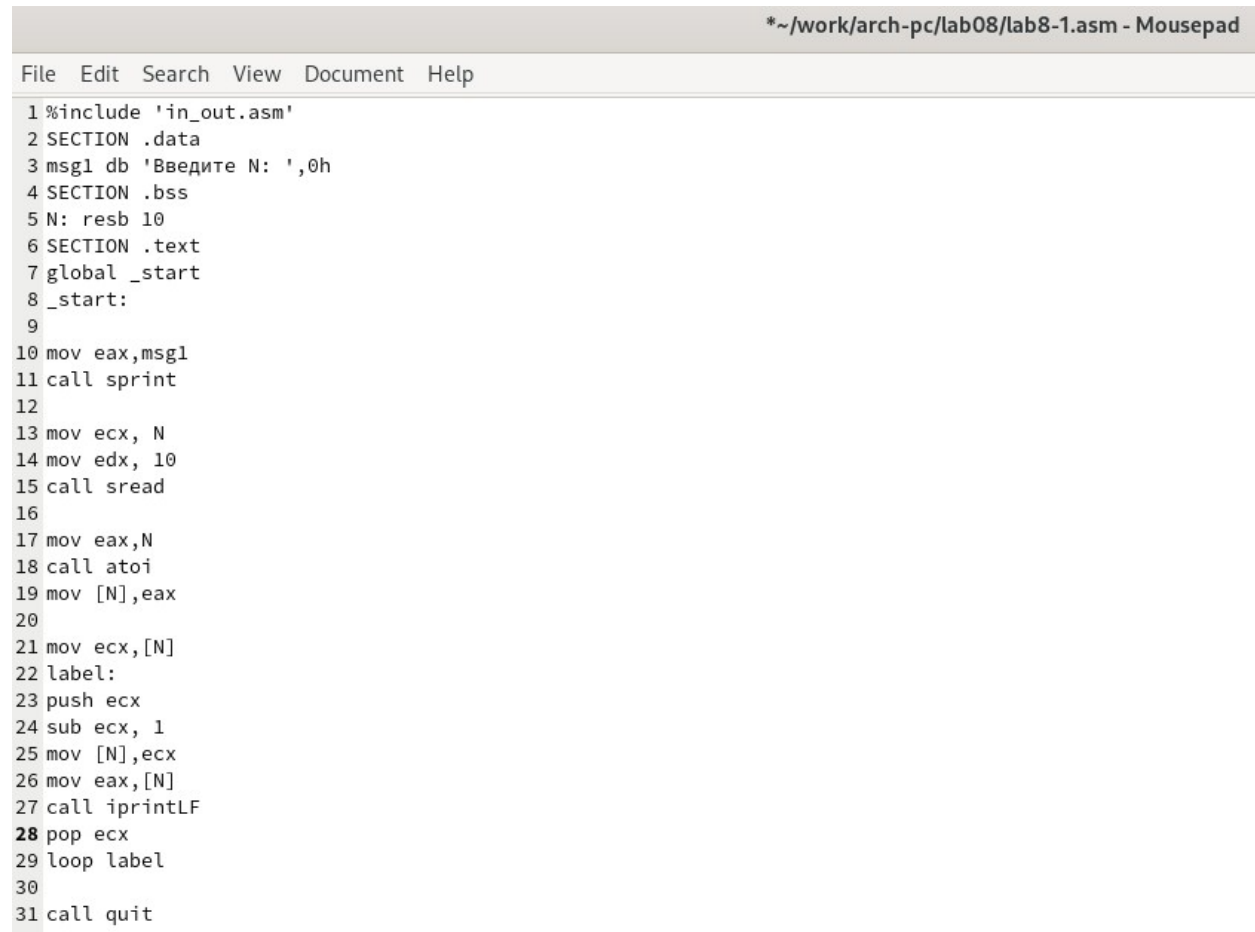
Теперь из-за того, что регистр `ecx` на каждой итерации уменьшается на 2 значения, количество итераций так же уменьшается в два раза (рис. 4.5).

```
irina_panyavkina@vbox:~/work/arch-pc/lab08$ mousepad lab8-1.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 12
11
9
7
5
3
1
irina_panyavkina@vbox:~/work/arch-pc/lab08$
```

*Рис. 4.5: Запуск исполняемого файла (изменённой программы)*



Редактирую программу, добавляя в неё команды push и pop (рис. 4.6).



```
*~/work/arch-pc/lab08/lab8-1.asm - Mousepad
File Edit Search View Document Help
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
10 mov eax,msg1
11 call sprint
12
13 mov ecx, N
14 mov edx, 10
15 call sread
16
17 mov eax,N
18 call atoi
19 mov [N],eax
20
21 mov ecx,[N]
22 label:
23 push ecx
24 sub ecx, 1
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF
28 pop ecx
29 loop label
30
31 call quit
```

*Рис. 4.6: Редактирование файла и сохранение программы*

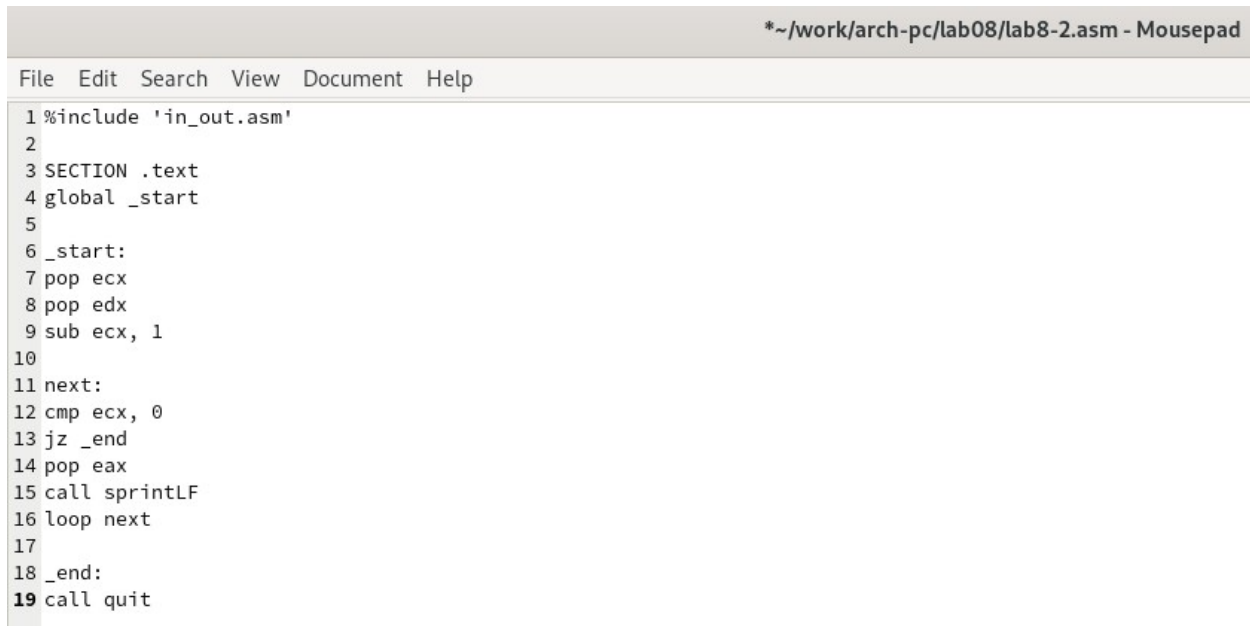
Видно, что теперь количество итераций совпадает введенному N, однако произошло смещение выводимых чисел на -1 (рис. 4.7).

```
irina_panyavkina@vbox:~/work/arch-pc/lab08$ mousepad lab8-1.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 12
11
10
9
8
7
6
5
4
3
2
1
0
```

Рис. 4.7: Запуск исполняемого файла (изменённой программы)

## 4.2 Обработка аргументов командной строки

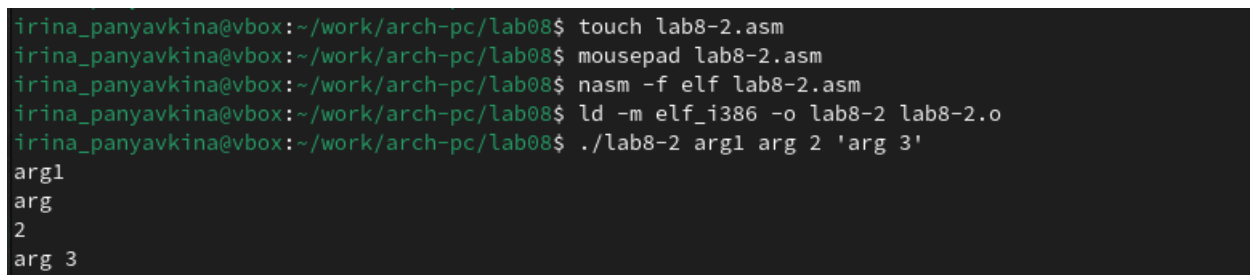
Создаю новый рабочий файл lab8-2.asm и вставляю в него скопированную программу из следующего листинга (рис. 4.8)



```
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 sprintLF
16 loop next
17
18 _end:
19 call quit
```

Рис. 4.8: Сохранение новой программы из листинга

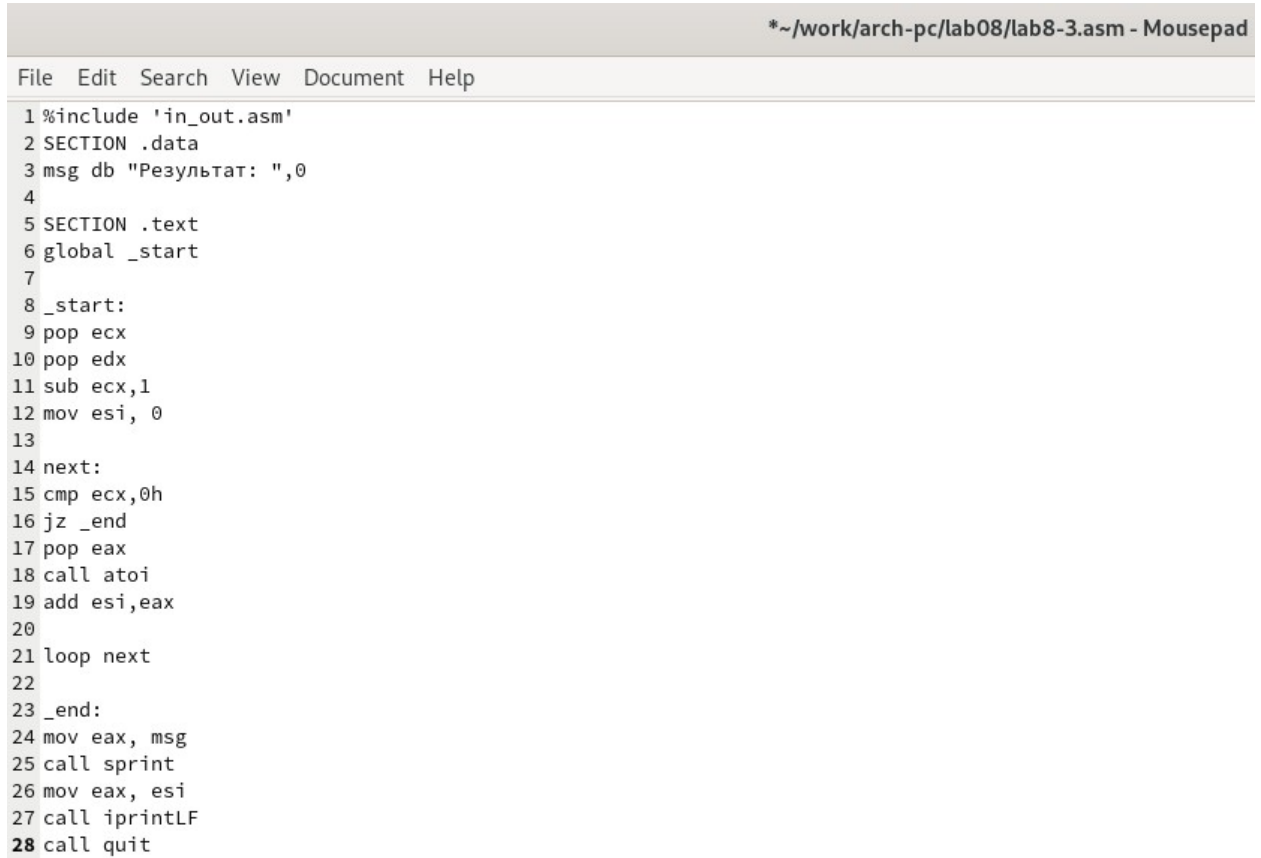
Компилирую программу и запускаю, указав аргументы. Программа обработала то же количество аргументов, что и вводилось (рис. 4.9).



```
irina_panyavkina@vbox:~/work/arch-pc/lab08$ touch lab8-2.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ mousepad lab8-2.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ./lab8-2 arg1 arg 2 'arg 3'
arg1
arg
2
arg 3
```

Рис. 4.9: Компиляция и запуск исполняемого файла с аргументами

Создаю новый рабочий файл lab8-3.asm и вставляю в него скопированную программу из третьего листинга (рис. 4.10).



The screenshot shows a text editor window titled `*~/work/arch-pc/lab08/lab8-3.asm - Mousepad`. The menu bar includes `File`, `Edit`, `Search`, `View`, `Document`, and `Help`. The code is as follows:

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4
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 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: Сохранение новой программы из листинга*

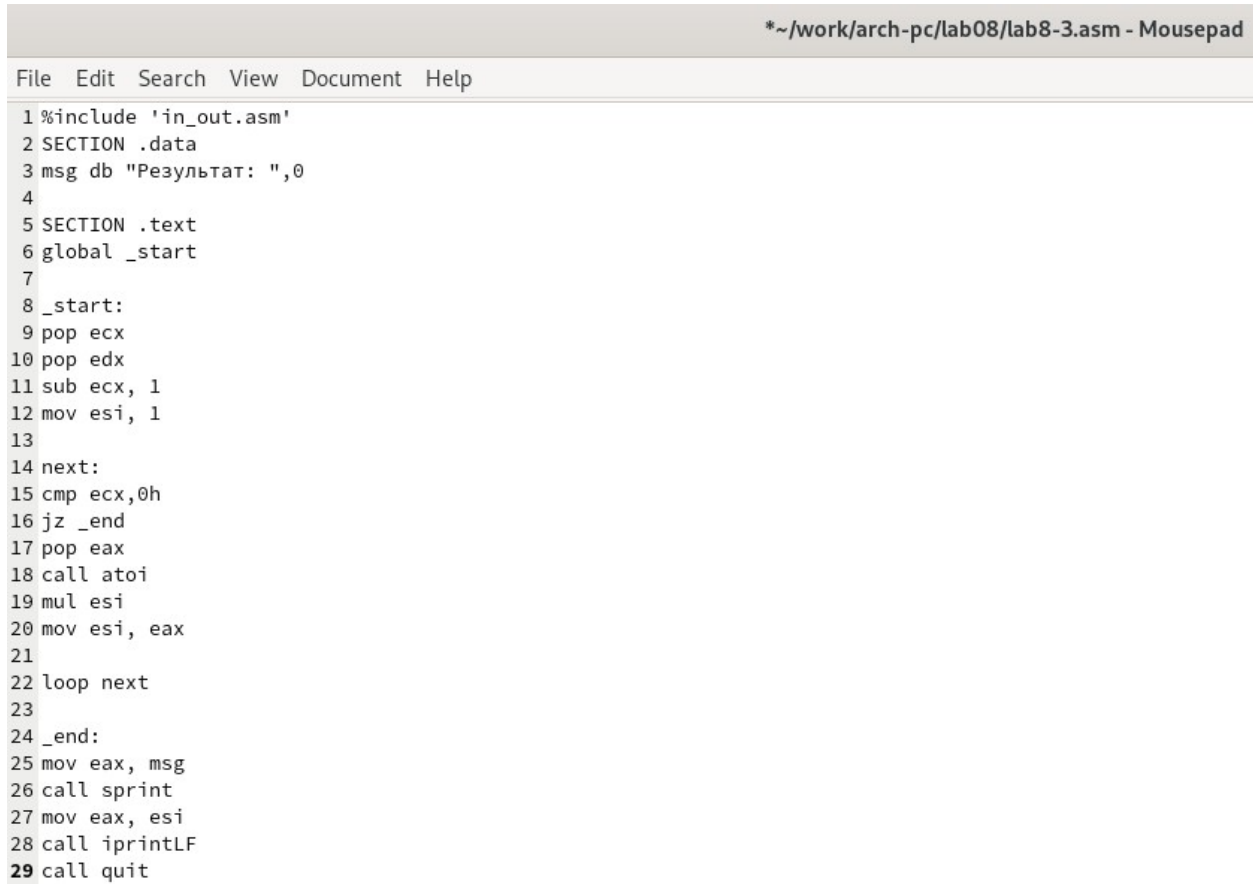
Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа должна их складывать (рис. 4.11).

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

*Рис. 4.11: Компиляция и запуск исполняемого файла с аргументами*

Программа работает корректно, выдаёт верный результат.

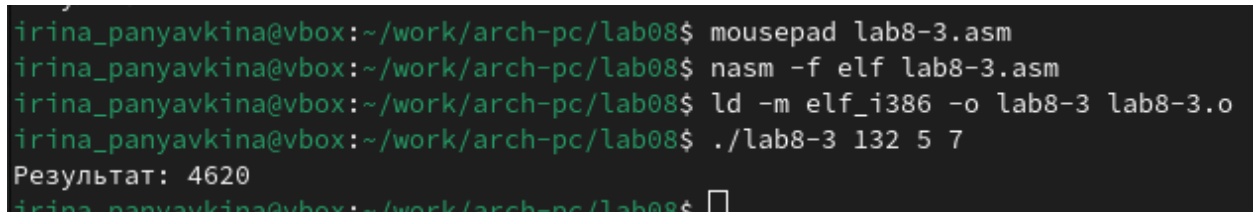
Затем изменяю программу так, чтобы указанные аргументы умножались, а не складывались (рис. 4.12).



```
*~/work/arch-pc/lab08/lab8-3.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4
5 SECTION .text
6 global _start
7
8 _start:
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 mul esi
20 mov esi, eax
21
22 loop next
23
24 _end:
25 mov eax, msg
26 call sprint
27 mov eax, esi
28 call iprintLF
29 call quit
```

Рис. 4.12: Редактирование файла и сохранение программы

Теперь программа умножает введенные числа, результат верный (рис. 4.13).



```
irina_panyavkina@vbox:~/work/arch-pc/lab08$ mousepad lab8-3.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ./lab8-3 132 5 7
Результат: 4620
irina_panyavkina@vbox:~/work/arch-pc/lab08$
```

Рис. 4.13: Запуск исполняемого файла (изменённой программы)

## 4.3 Выполнение заданий для самостоятельной работы

При выполнении 6 лабораторной работы, с помощью программы я выяснила, что мой вариант — 15. Мне нужно написать программу, используя следующую функцию:  $f(x)=6x+13$  (рис. 4.14).

```

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

Рис. 4.14: Программа самостоятельной работы

Код первой программы:

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

```
SECTION .data
```

```
msg_func db "Функция:  $f(x) = 6x + 13$ ", 0
```

```
msg_result db "Результат: ", 0
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax, msg_func
```

```
call sprintLF
```

```
pop ecx
```

```
pop edx
```

```
sub ecx, 1
```

```
mov esi, 0
```

```
next:
```

```
cmp ecx, 0
```

```
jz _end
```

```
pop eax
```

```
call atoi
```

```
mov ebx, 6
```

```
mul ebx
```



```
add eax, 13
```

```
add esi, eax
```

```
loop next
```

```
_end:
```

```
mov eax, msg_result
```

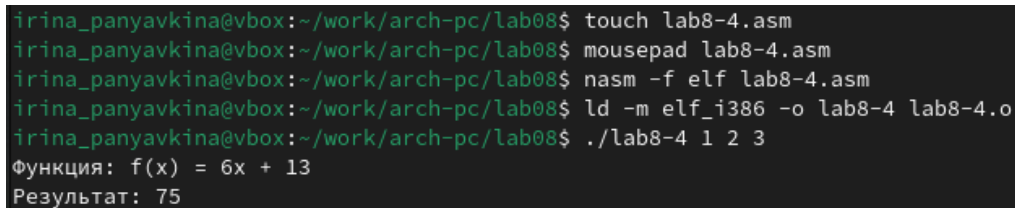
```
call sprint
```

```
mov eax, esi
```

```
call iprintLF
```

```
call quit
```

Проверяю корректность написания программы и её работу, указав в качестве аргумента несколько чисел (рис. 4.15).



```
irina_panyavkina@vbox:~/work/arch-pc/lab08$ touch lab8-4.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ mousepad lab8-4.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
irina_panyavkina@vbox:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
Функция: f(x) = 6x + 13
Результат: 75
```

*Рис. 4.15: Запуск исполняемого файла*

Программа работает успешно, результат получается правильным — соответствует условию задания.

## 5 Выводы

Выполнив данную лабораторную работу, я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.

## Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.Org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.

14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).