

Отчет по лабораторной работе №8

Дисциплина: архитектура компьютера

Шабакова Карина Баировна

Содержание

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

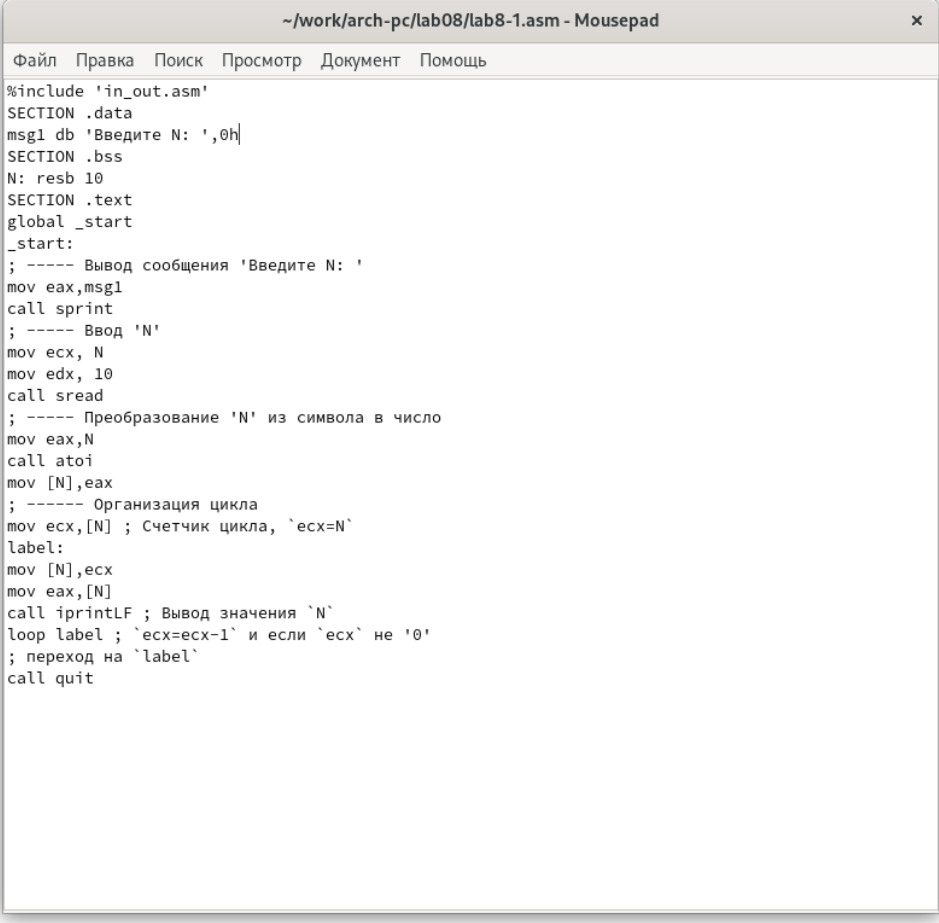
Создаю каталог для программ лабораторной работы №8 (рис. 1).



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

Рис. 1: Создание каталога

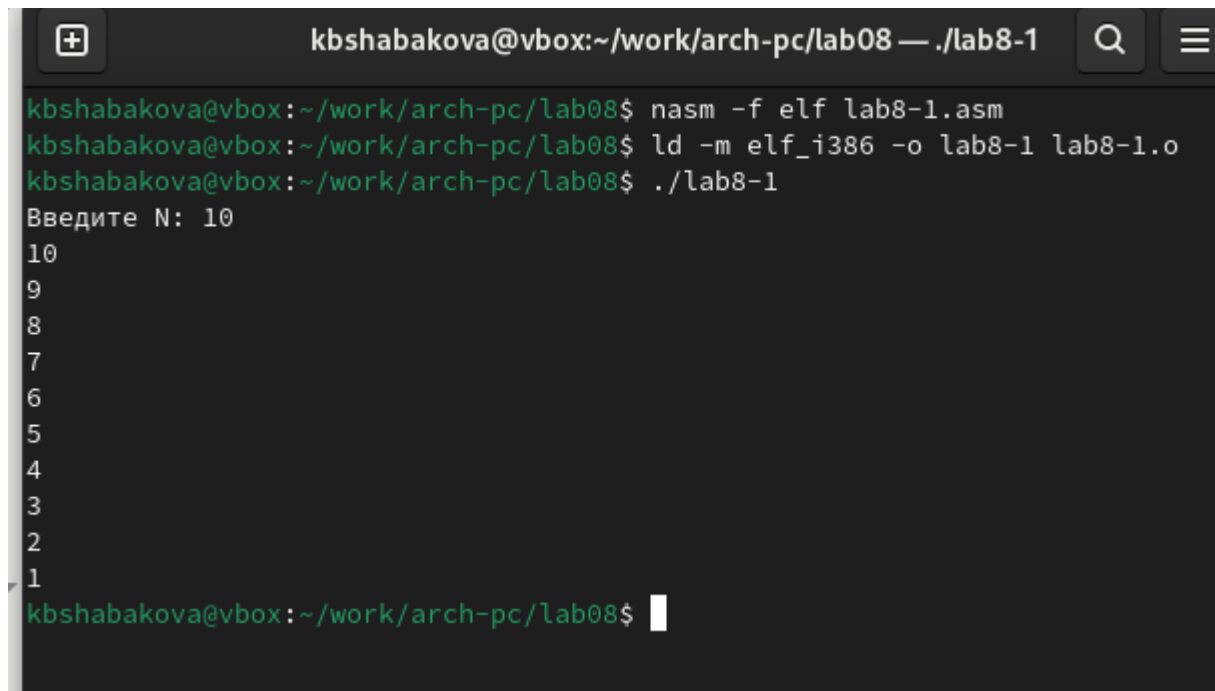
Копирую в созданный файл программу из листинга. (рис. 2).



```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit
```

Рис. 2: Копирование программы из листинга

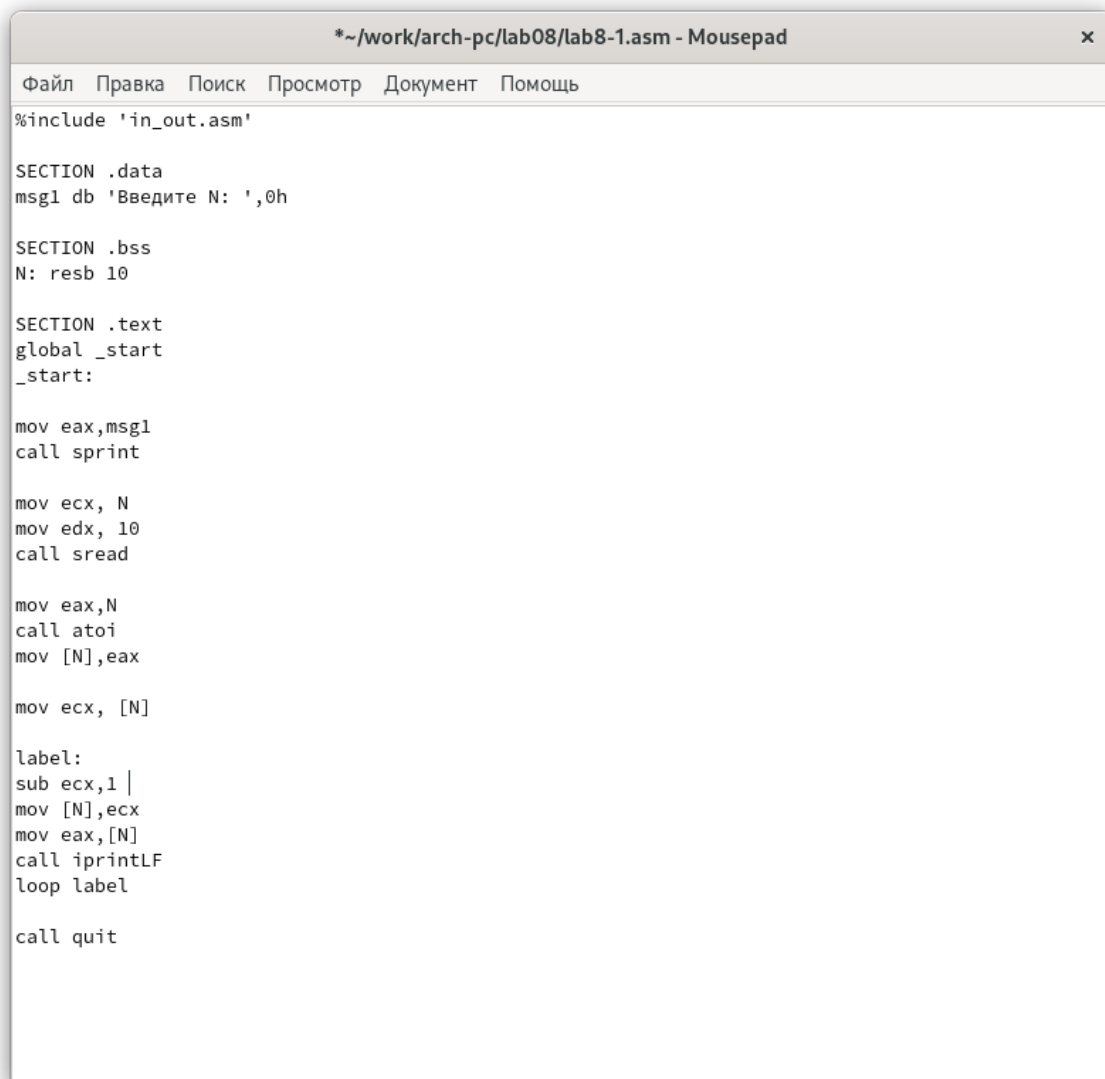
Запускаю программу, она показывает работу циклов в NASM (рис. 3).

A terminal window with a dark background. The title bar shows the user 'kbshabakova@vbox' and the current directory '~/work/arch-pc/lab08' with a subdirectory './lab8-1'. The terminal contains the following text:

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

Рис. 3: Запуск программы

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра есх (рис. 4).



```
*~/work/arch-pc/lab08/lab8-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%include 'in_out.asm'

SECTION .data
msg1 db 'Введите N: ',0h

SECTION .bss
N: resb 10

SECTION .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx, N
mov edx, 10
call sread

mov eax,N
call atoi
mov [N],eax

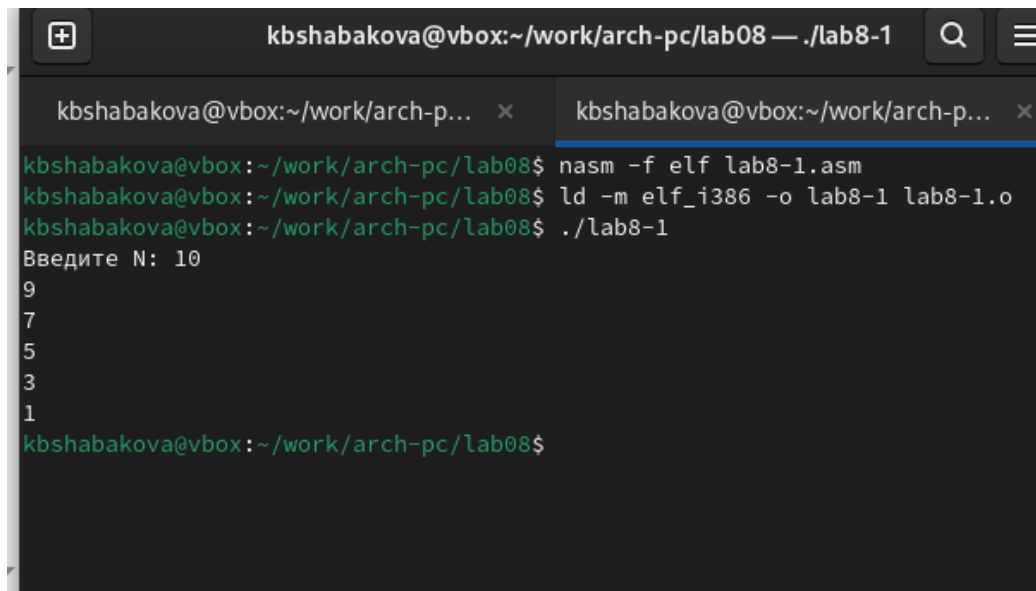
mov ecx, [N]

label:
sub ecx,1 |
mov [N],ecx
mov eax,[N]
call iprintLF
loop label

call quit
```

Рис. 4: Изменение программы

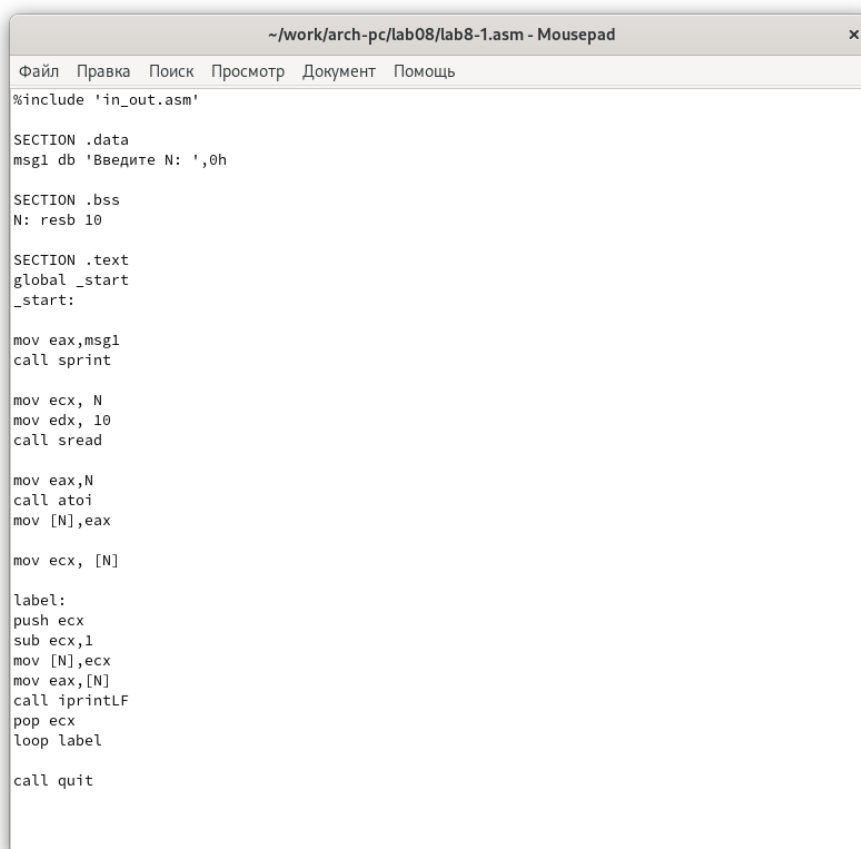
Из-за того, что теперь регистр `ecx` на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. 5).



```
kbshabakova@vbox:~/work/arch-pc/lab08 — ./lab8-1
kbshabakova@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
kbshabakova@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
kbshabakova@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
kbshabakova@vbox:~/work/arch-pc/lab08$
```

Рис. 5: Запуск измененной программы

Добавляю команды `push` и `pop` в программу (рис. 6).



```
~/work/arch-pc/lab08/lab8-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%include 'in_out.asm'

SECTION .data
msg1 db 'Введите N: ',0h

SECTION .bss
N: resb 10

SECTION .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx, N
mov edx, 10
call sread

mov eax,N
call atoi
mov [N],eax

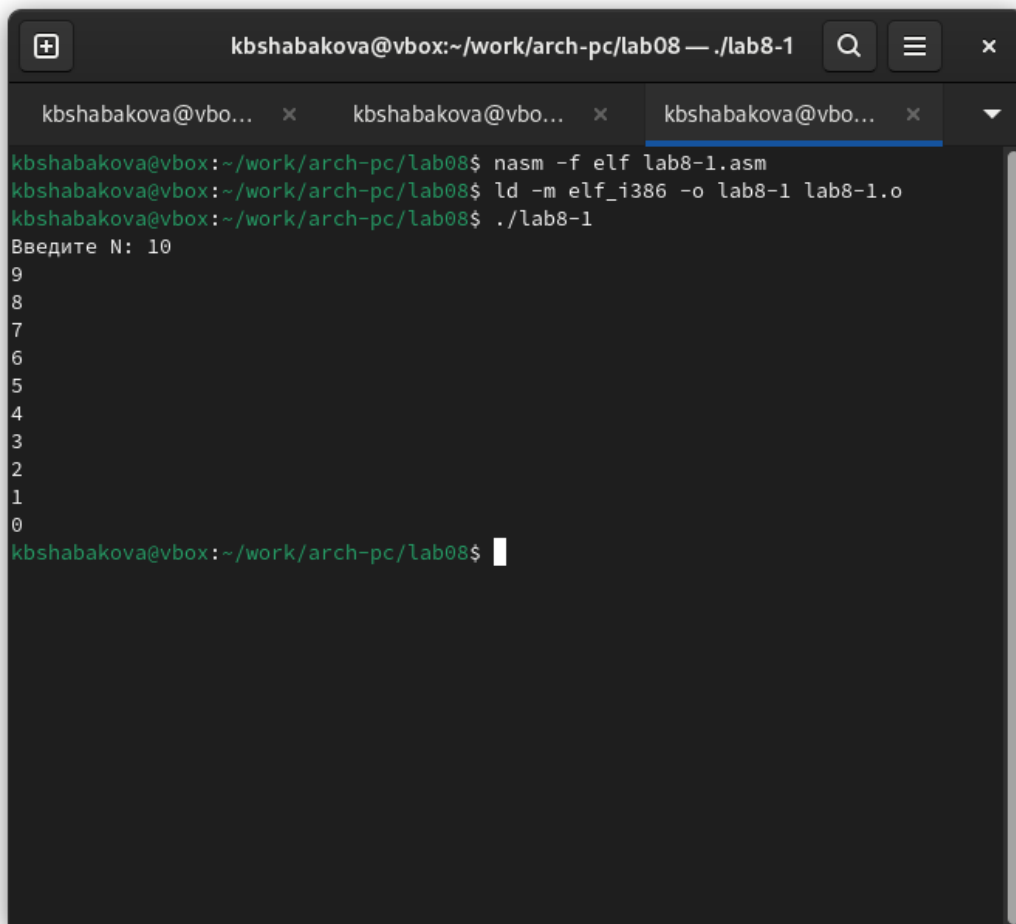
mov ecx, [N]

label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx
loop label

call quit
```

Рис. 6: Добавление `push` и `pop` в цикл программы

Теперь количество итераций совпадает введенному `N`, но произошло смещение выводимых чисел на -1 (рис. 7).

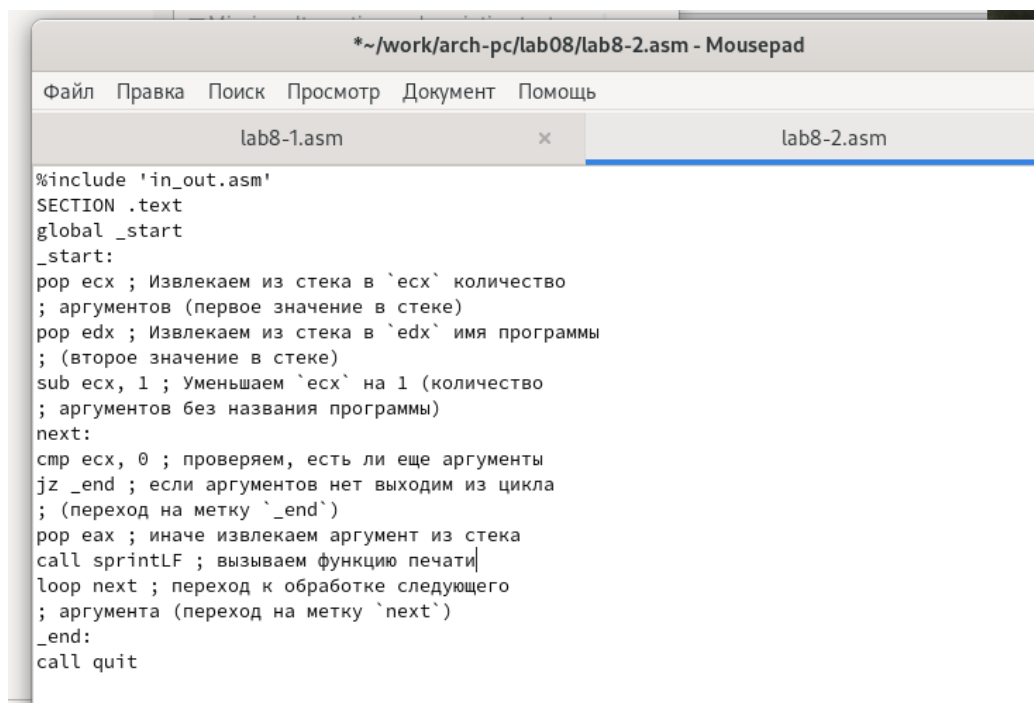


```
kbshabakova@vbox:~/work/arch-pc/lab08 — ./lab8-1
kbshabakova@vbo... x kbshabakova@vbo... x kbshabakova@vbo... x
kbshabakova@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
kbshabakova@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
kbshabakova@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите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

Рис. 7: Запуск измененной программы

4.2 Обработка аргументов командной строки

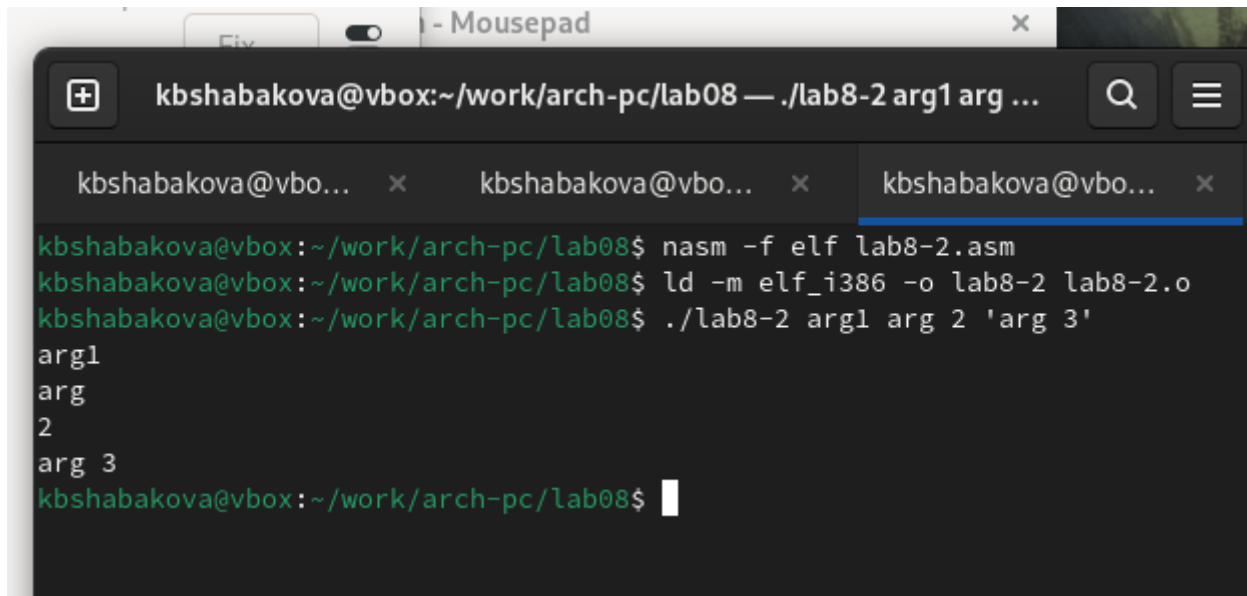
Создаю новый файл для программы и копирую в него код из следующего листинга (рис. 8).



```
*~/work/arch-pc/lab08/lab8-2.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь
lab8-1.asm x lab8-2.asm
%include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx ; Извлекаем из стека в `ecx` количество
    ; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в `edx` имя программы
    ; (второе значение в стеке)
    sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
    ; аргументов без названия программы)
    next:
    cmp ecx, 0 ; проверяем, есть ли еще аргументы
    jz _end ; если аргументов нет выходим из цикла
    ; (переход на метку `_end`)
    pop eax ; иначе извлекаем аргумент из стека
    call sprintf ; вызываем функцию печати
    loop next ; переход к обработке следующего
    ; аргумента (переход на метку `next`)
    _end:
    call quit
```

Рис. 8: Копирование программы из листинга

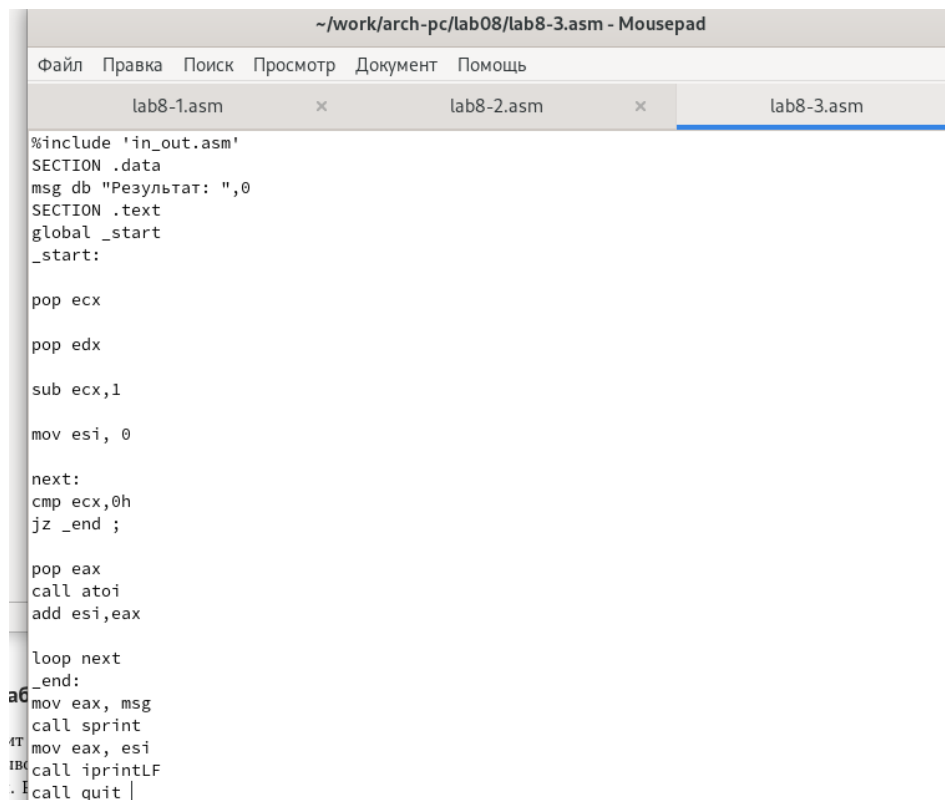
Копирую программу и запускаю, указав аргументы. Программой было обработано то же количество аргументов, что и было введено (рис. 9).



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

Рис. 9: Запуск второй программы

Создаю новый файл для программы и копирую в него код из третьего листинга (рис. 10).



```
~/work/arch-pc/lab08/lab8-3.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
lab8-1.asm  x  lab8-2.asm  x  lab8-3.asm
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:

pop ecx

pop edx

sub ecx,1

mov esi, 0

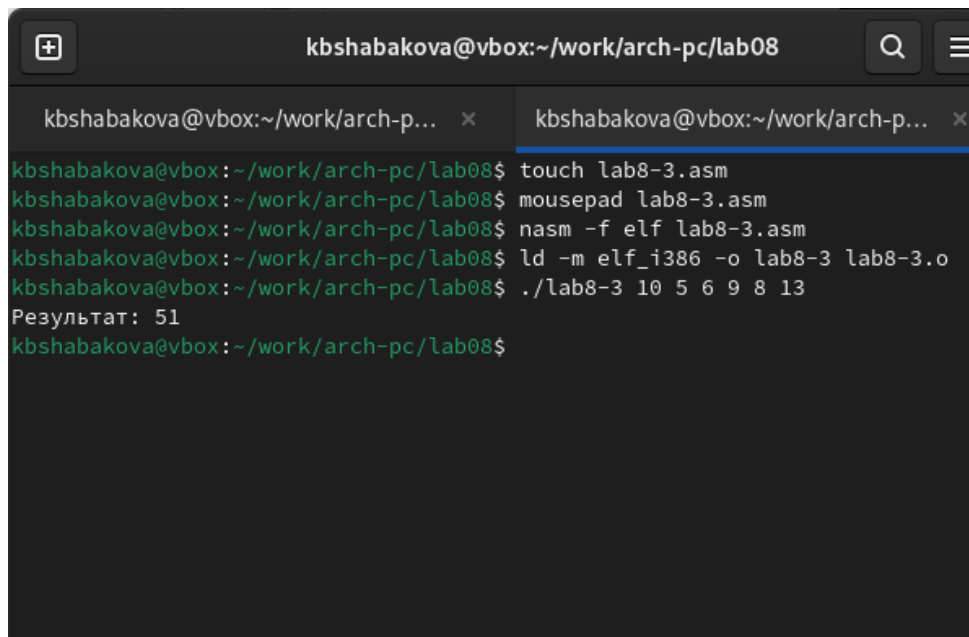
next:
cmp ecx,0h
jz _end ;

pop eax
call atoi
add esi,eax

loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

Рис. 10: Копирование программы из третьего листинга

Копирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. 11).

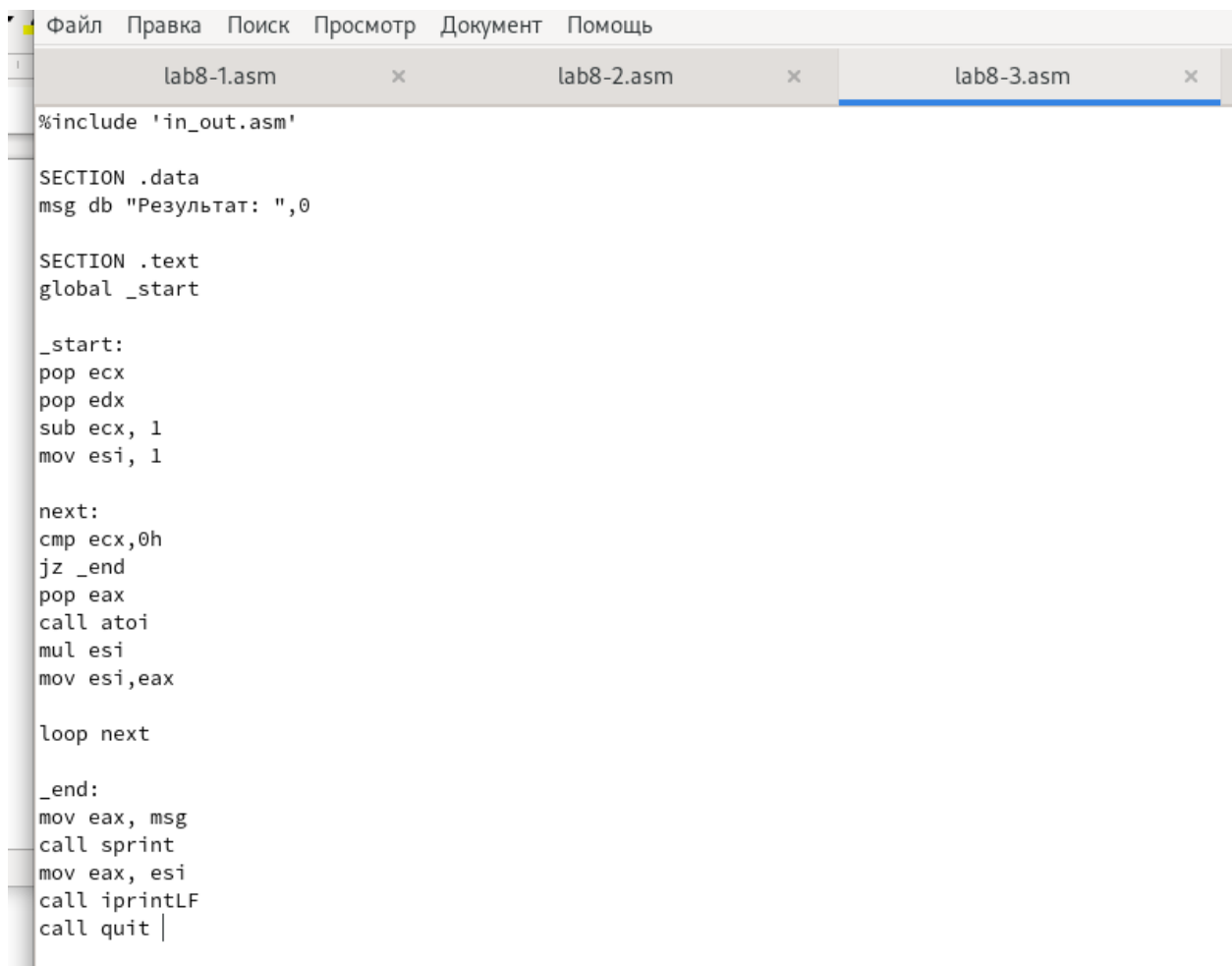


A terminal window titled 'kbshabakova@vbox:~/work/arch-pc/lab08'. It shows the following commands and output:

```
kbshabakova@vbox:~/work/arch-pc/lab08$ touch lab8-3.asm
kbshabakova@vbox:~/work/arch-pc/lab08$ mousepad lab8-3.asm
kbshabakova@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kbshabakova@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
kbshabakova@vbox:~/work/arch-pc/lab08$ ./lab8-3 10 5 6 9 8 13
Результат: 51
kbshabakova@vbox:~/work/arch-pc/lab08$
```

Рис. 11: Запуск третьей программы

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. 12).



A screenshot of a text editor window with the title bar 'Файл Правка Поиск Просмотр Документ Помощь'. The editor has three tabs: 'lab8-1.asm', 'lab8-2.asm', and 'lab8-3.asm'. The 'lab8-3.asm' tab is active and shows the following assembly code:

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ",0

SECTION .text
global _start

_start:
pop ecx
pop edx
sub ecx, 1
mov esi, 1

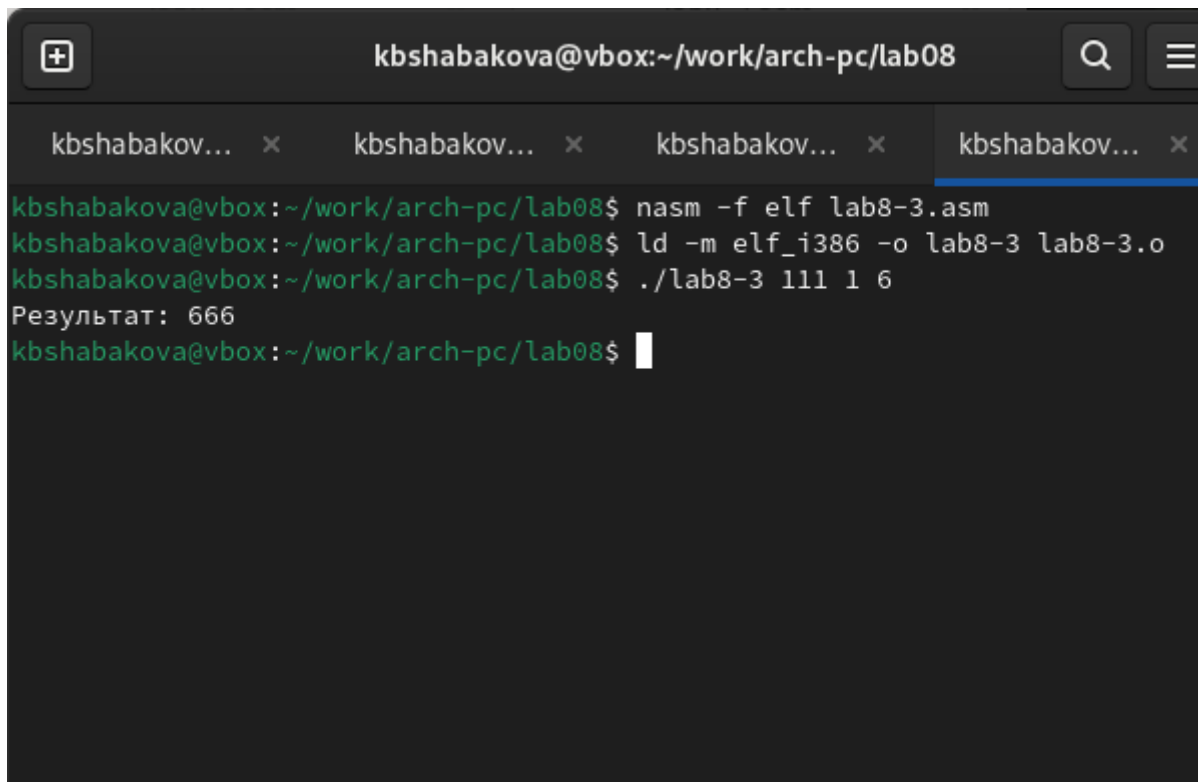
next:
cmp ecx,0h
jz _end
pop eax
call atoi
mul esi
mov esi,eax

loop next

_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit |
```

Рис. 12: Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. 13).

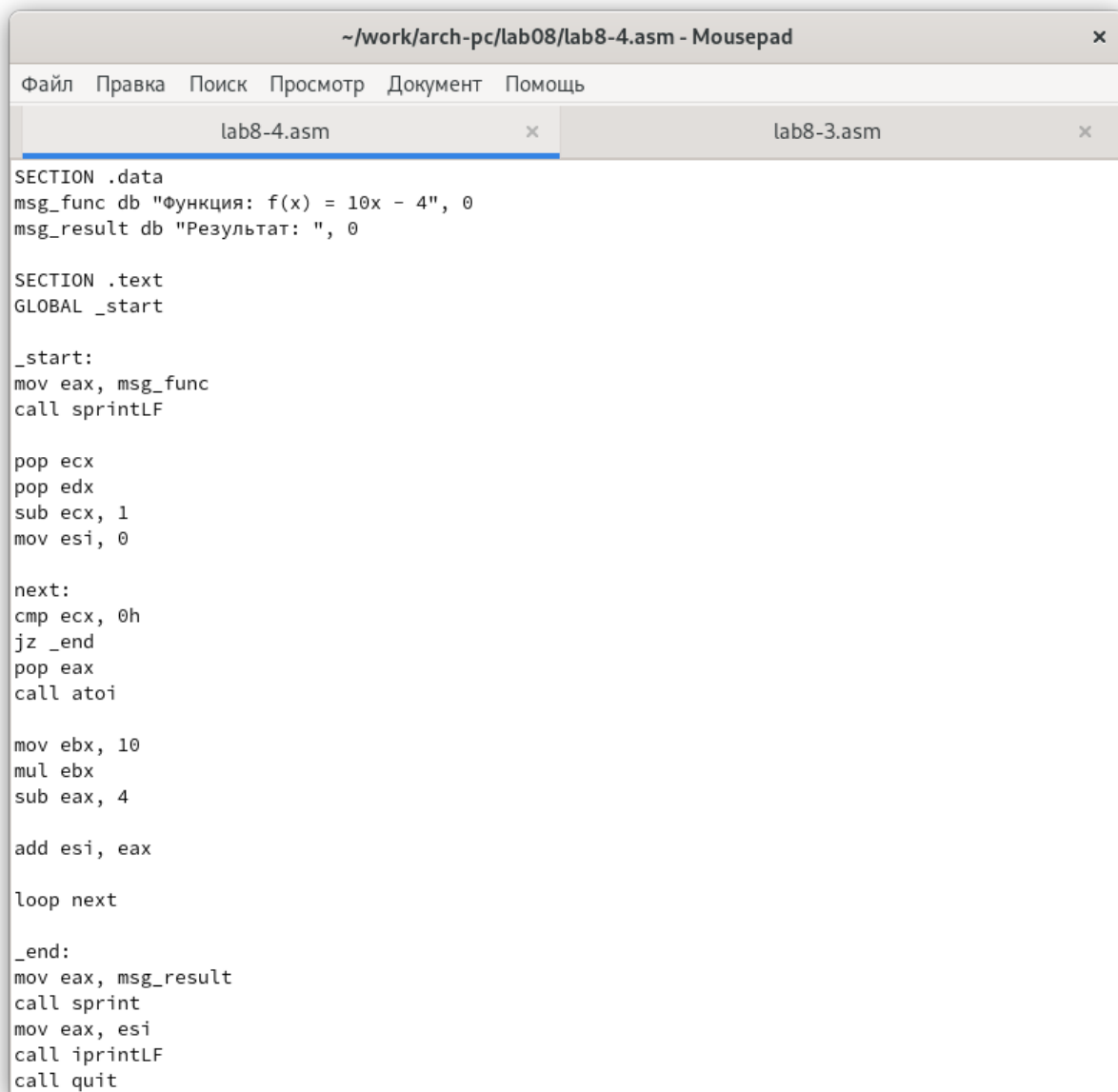
A terminal window with a dark background. The title bar shows the user 'kbshabakova@vbox' and the directory '~/work/arch-pc/lab08'. There are four tabs, all labeled 'kbshabakov...'. The terminal content shows three commands: 'nasm -f elf lab8-3.asm', 'ld -m elf_i386 -o lab8-3 lab8-3.o', and './lab8-3 111 1 6'. The output of the last command is 'Результат: 666'.

```
kbshabakova@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kbshabakova@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
kbshabakova@vbox:~/work/arch-pc/lab08$ ./lab8-3 111 1 6
Результат: 666
kbshabakova@vbox:~/work/arch-pc/lab08$
```

Рис. 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 девятым вариантом (рис. 14).



```
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
```

Рис. 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

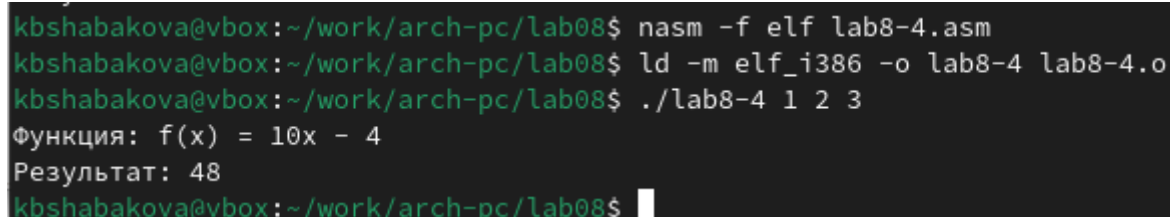
add esi, eax

loop next

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

```

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. 15).



```

kbshabakova@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
kbshabakova@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
kbshabakova@vbox:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
Функция: f(x) = 10x - 4
Результат: 48
kbshabakova@vbox:~/work/arch-pc/lab08$

```

Рис. 15: Запуск программы для самостоятельной работы

5 Выводы

В результате выполнения данной лабораторной работы и самостоятельной работы. Я приобрела навыки написания программ с использованием циклов, а также научилась обрабатывать аргументы командной строки.

6 Список литературы

1. [Курс на ТУИС](#)
2. [Лабораторная работа №8](#)
3. [Программирование на языке ассемблера NASM Столяров А. В.](#)