

# **Отчёт по лабораторной работе №8**

**Программирование цикла. Обработка аргументов командной строки.**

Власов Артем Сергеевич

# 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>
3.1	Реализация циклов в NASM . . . . .	6
3.2	Обработка аргументов командной строки. . . . .	9
3.3	Задание для самостоятельной работы . . . . .	12
<b>4</b>	<b>Выводы</b>	<b>14</b>

## Список иллюстраций

3.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code> . . . . .	6
3.2	Заполняем файл . . . . .	7
3.3	Запускаем файл и проверяем его работу . . . . .	7
3.4	Изменяем файл . . . . .	8
3.5	Запускаем файл и смотрим на его работу . . . . .	8
3.6	Редактируем файл . . . . .	9
3.7	Проверяем, сошелся ли наш вывод с данным в условии выводом .	9
3.8	Создаем файл командой <code>touch</code> . . . . .	9
3.9	Заполняем файл . . . . .	10
3.10	Смотрим на работу программ . . . . .	10
3.11	Создаем файл командой <code>touch</code> . . . . .	10
3.12	Заполняем файл . . . . .	11
3.13	Смотрим на работу программы . . . . .	11
3.14	Изменяем файл . . . . .	11
3.15	Проверяем работу файла(работает правильно) . . . . .	11
3.16	Создаем файл командой <code>touch</code> . . . . .	12
3.17	Пишем программу . . . . .	13
3.18	Смотрим на работу программы при $x_1=1$ $x_2=2$ $x_3=3$ $x_4=4$ (всё верно)	13
3.19	Смотрим на работу программы при $x_1=5$ $x_2=4$ $x_3=9$ $x_4=6$ (всё верно)	13

# **1 Цель работы**

Изучить работу циклов и обработкой аргументов командной строки.

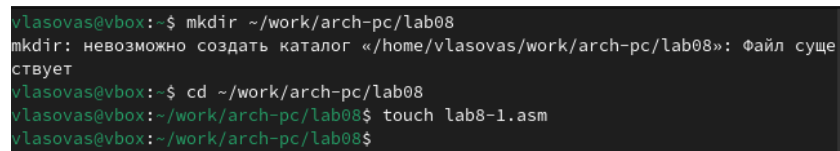
## 2 Задание

Написать программы с использованием циклов и обработкой аргументов командной строки.

## 3 Выполнение лабораторной работы

### 3.1 Реализация циклов в NASM

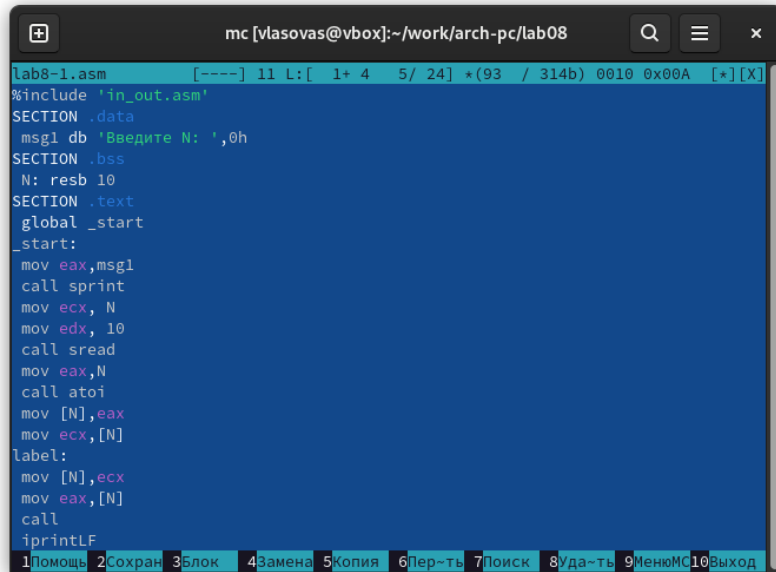
Создаем каталог для программ ЛБ8, и в нем создаем файл (рис. fig. 3.1).



```
vlasovas@ybox: ~$ mkdir ~/work/arch-pc/lab08
mkdir: невозможно создать каталог «/home/vlasovas/work/arch-pc/lab08»: файл существует
vlasovas@ybox: ~$ cd ~/work/arch-pc/lab08
vlasovas@ybox:~/work/arch-pc/lab08$ touch lab8-1.asm
vlasovas@ybox:~/work/arch-pc/lab08$
```

Рис. 3.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

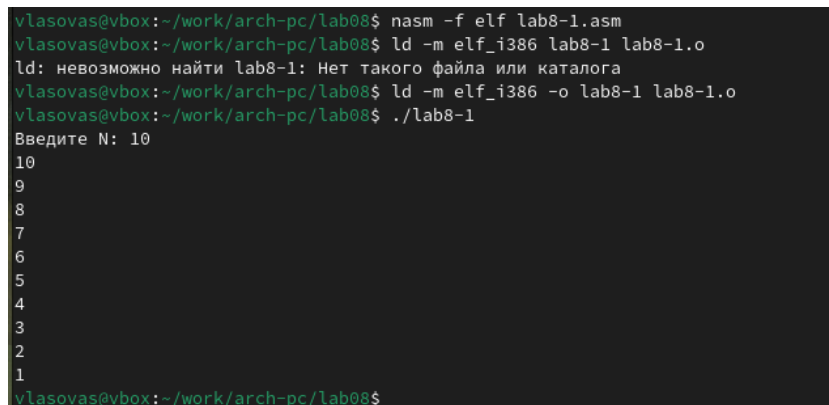
Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.1 (рис. fig. 3.2).



```
lab8-1.asm [----] 11 L: [ 1+ 4 5/ 24] *(93 / 314b) 0010 0x00A [*][X]
#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:
mov [N],ecx
mov eax,[N]
call
iprintLF
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 3.2: Заполняем файл

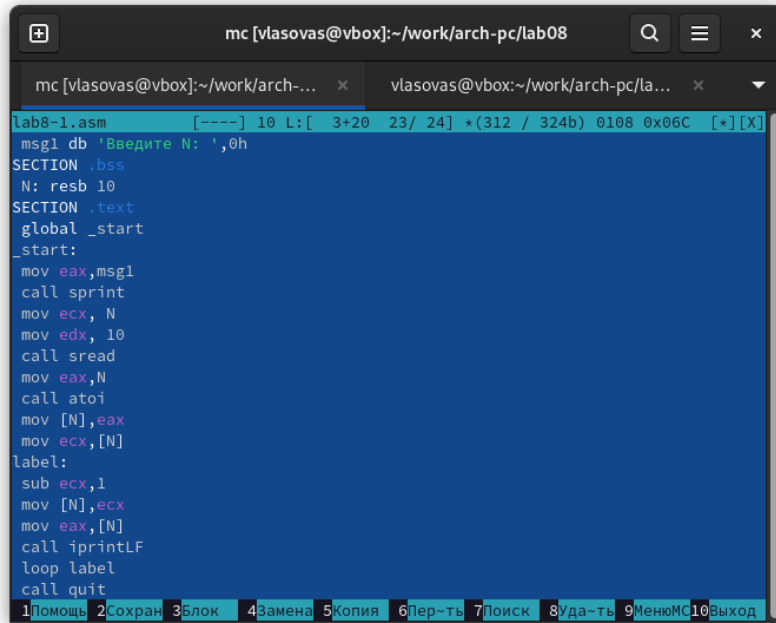
Создаем исполняемый файл и запускаем его (рис. fig. 3.3).



```
vlasovas@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
vlasovas@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-1 lab8-1.o
ld: невозможно найти lab8-1: Нет такого файла или каталога
vlasovas@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
vlasovas@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
vlasovas@vbox:~/work/arch-pc/lab08$
```

Рис. 3.3: Запускаем файл и проверяем его работу

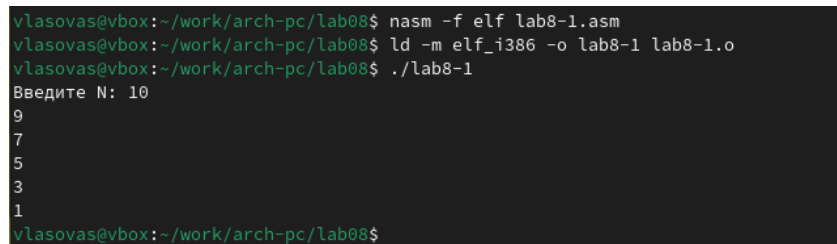
Снова открываем файл для редактирования и изменяем его, добавив изменение значения регистра в цикле (рис. fig. 3.4).



```
lab8-1.asm [----] 10 L: [ 3+20 23/ 24] *(312 / 324b) 0108 0x06C [*] [X]
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. 3.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.5).



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

Рис. 3.5: Запускаем файл и смотрим на его работу

Регистр `ecx` принимает значения 9,7,5,3,1(на вход подается число 10, в цикле `label` данный регистр уменьшается на 2 командой `sub` и `loop`).

Число проходов цикла не соответствует числу `N`, так как уменьшается на 2.

Снова открываем файл для редактирования и изменяем его, чтобы все корректно работало (рис. fig. 3.6).



```
label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx
loop label
call quit
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС10Выход
```

Рис. 3.6: Редактируем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.7).

```
vlasovas@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
vlasovas@vbox:~/work/arch-pc/lab08$ ^[[200~ld -m elf_i386 -o lab8-1 lab8-1.o
bash: ld: команда не найдена...
vlasovas@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
vlasovas@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
vlasovas@vbox:~/work/arch-pc/lab08$
```

Рис. 3.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

В данном случае число проходов цикла равна числу N.

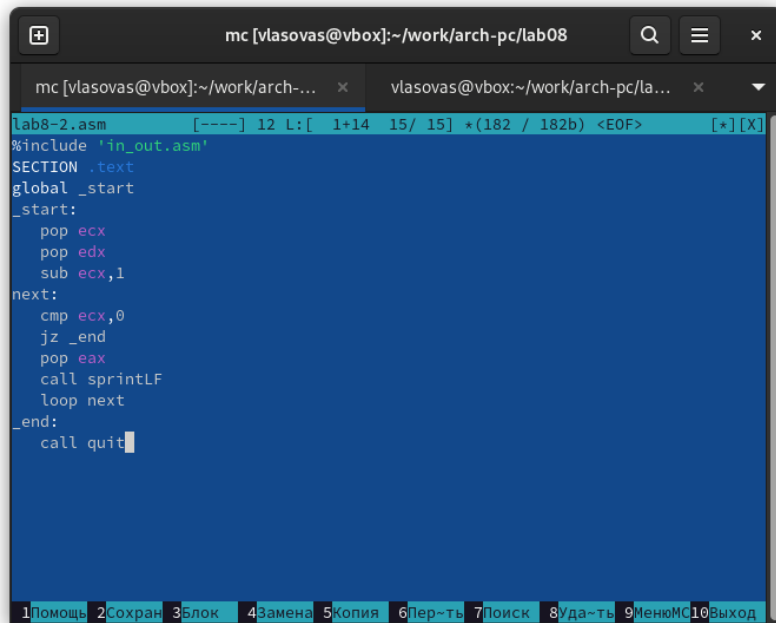
## 3.2 Обработка аргументов командной строки.

Создаем новый файл (рис. fig. 3.8).

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

Рис. 3.8: Создаем файл командой touch

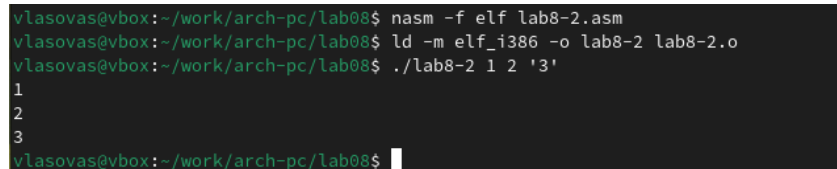
Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.2 (рис. fig. 3.9).

A screenshot of a text editor window titled 'mc [vlasovas@vbox]:~/work/arch-pc/lab08'. The editor shows the contents of 'lab8-2.asm'. The code includes a comment line, an include directive for 'in\_out.asm', a section declaration for '.text', and assembly instructions for setting up registers and a loop. The instructions are: 'global \_start', '\_start:', 'pop ecx', 'pop edx', 'sub ecx,1', 'next:', 'cmp ecx,0', 'jz \_end', 'pop eax', 'call sprintf', 'loop next', '\_end:', and 'call quit'. The bottom of the window features a menu bar with options: 1Помощь, 2Сохран, 3Блок, 4Замена, 5Копия, 6Пер-ть, 7Поиск, 8Уда-ть, 9Меню, 10Выход.

```
lab8-2.asm [----] 12 L: [ 1+14 15/ 15] *(182 / 182b) <EOF> [*] [X]
#include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx,1
next:
    cmp ecx,0
    jz _end
    pop eax
    call sprintf
    loop next
_end:
    call quit
```

Рис. 3.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, указав аргументы (рис. fig. 3.10).

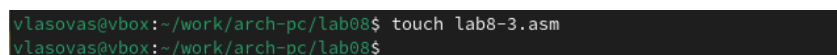
A screenshot of a terminal window showing the compilation and execution of the assembly file. The commands and their outputs are: 'nasm -f elf lab8-2.asm', 'ld -m elf\_i386 -o lab8-2 lab8-2.o', and './lab8-2 1 2 '3''. The output of the execution shows the numbers 1, 2, and 3 on separate lines.

```
vlasovas@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
vlasovas@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
vlasovas@vbox:~/work/arch-pc/lab08$ ./lab8-2 1 2 '3'
1
2
3
vlasovas@vbox:~/work/arch-pc/lab08$
```

Рис. 3.10: Смотрим на работу программ

Программой было обработано 3 аргумента.

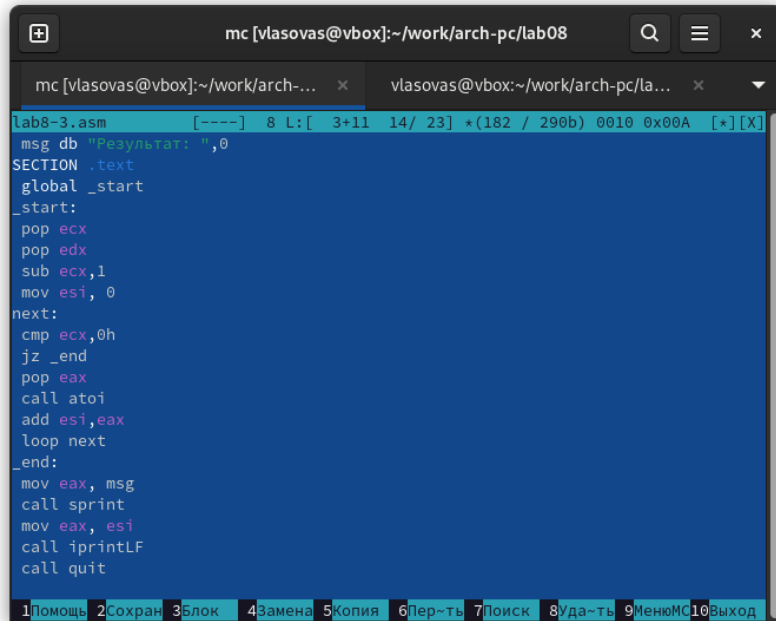
Создаем новый файл lab8-3.asm (рис. fig. 3.11).

A screenshot of a terminal window showing the command 'touch lab8-3.asm' being executed to create a new file.

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

Рис. 3.11: Создаем файл командой touch

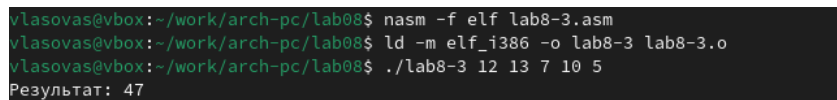
Открываем файл и заполняем его в соответствии с листингом 8.3 (рис. fig. 3.12).



```
lab8-3.asm [----] 8 L: [ 3+11 14/ 23] *(182 / 290b) 0010 0x00A [*] [X]
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. 3.12: Заполняем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. fig. 3.13).



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

Рис. 3.13: Смотрим на работу программы

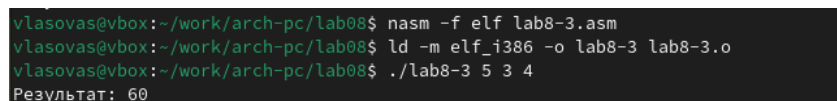
Снова открываем файл для редактирования и изменяем его, чтобы вычислялось произведение вводимых значений (рис. fig. 3.14).



```
call atoi
mul esi,
mov esi, eax
loop next
```

Рис. 3.14: Изменяем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. fig. 3.15).



```
vlasovas@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
vlasovas@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
vlasovas@vbox:~/work/arch-pc/lab08$ ./lab8-3 5 3 4
Результат: 60
```

Рис. 3.15: Проверяем работу файла(работает правильно)

### 3.3 Задание для самостоятельной работы

#### ВАРИАНТ-2

1. Напишите программу, которая находит сумму значений функции  $f(x)$  для  $x = x_1, x_2, \dots, x_n$ , т.е. программа должна выводить значение  $f(x_1) + f(x_2) + \dots + f(x_n)$ . Значения  $x_i$  передаются как аргументы. Вид функции  $f(x)$  выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах  $x = x_1, x_2, \dots, x_n$ .

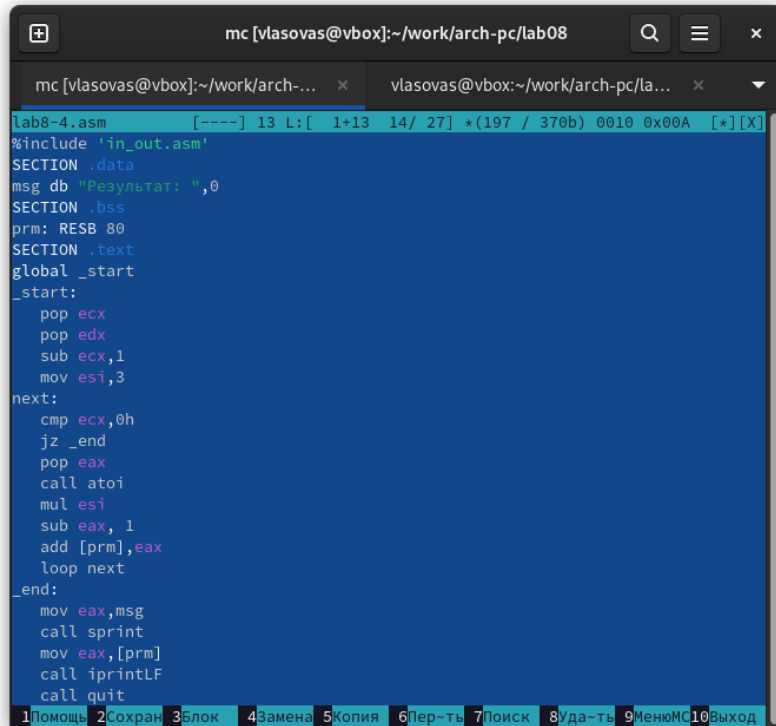
Создаем новый файл (рис. fig. 3.16).



```
lasovas@vbox:~/work/arch-pc/lab08$ touch lab8-4.asm
```

Рис. 3.16: Создаем файл командой touch

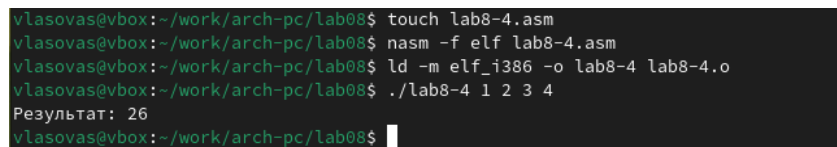
Открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения  $3x-1$  (рис. fig. 3.17).



```
lab8-4.asm [----] 13 L: [ 1+13 14/ 27] *(197 / 370b) 0010 0x00A [*] [X]
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .bss
prm: RESB 80
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx,1
    mov esi,3
next:
    cmp ecx,0h
    jz _end
    pop eax
    call atoi
    mul esi
    sub eax, 1
    add [prm],eax
    loop next
_end:
    mov eax,msg
    call sprint
    mov eax,[prm]
    call iprintLF
    call quit
```

Рис. 3.17: Пишем программу

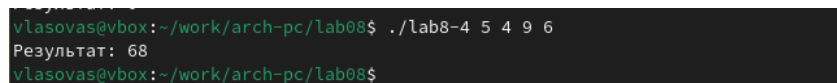
Транслируем файл и смотрим на работу программы (рис. fig. 3.18).



```
vlasovas@vbox:~/work/arch-pc/lab08$ touch lab8-4.asm
vlasovas@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
vlasovas@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
vlasovas@vbox:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Результат: 26
vlasovas@vbox:~/work/arch-pc/lab08$
```

Рис. 3.18: Смотрим на работу программы при  $x_1=1$   $x_2=2$   $x_3=3$   $x_4=4$ (всё верно)

Транслируем файл и смотрим на работу программы (рис. fig. 3.19).



```
vlasovas@vbox:~/work/arch-pc/lab08$ ./lab8-4 5 4 9 6
Результат: 68
vlasovas@vbox:~/work/arch-pc/lab08$
```

Рис. 3.19: Смотрим на работу программы при  $x_1=5$   $x_2=4$   $x_3=9$   $x_4=6$ (всё верно)

## 4 Выводы

Мы научились решать программы с использованием циклов и обработкой аргументов командной строки.