

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

по лабораторной работе № 2

дисциплина: Архитектура компьютера

Студент: Грязнов Михаил

Группа: НПИбд-01-22

**МОСКВА**

2022 г.

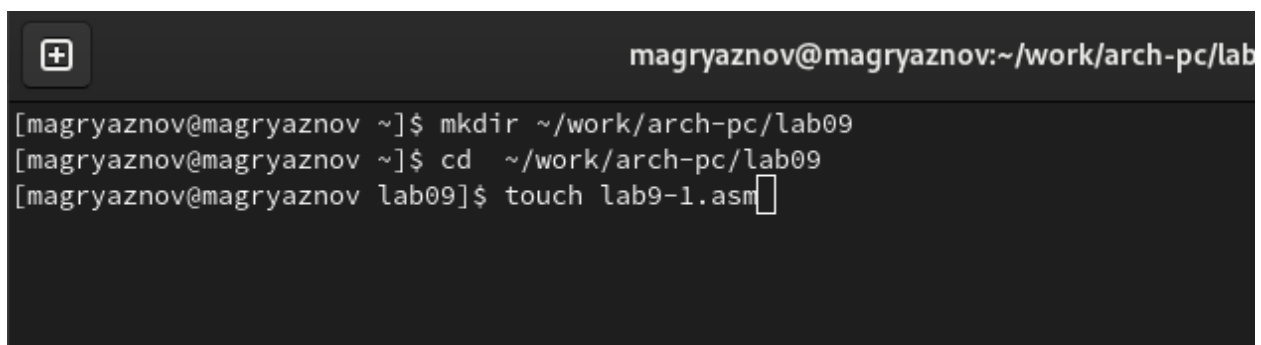
### Цель работы:

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

### Порядок выполнения лабораторной работы:

#### Реализация циклов в NASM.

Создадим каталог для программ лабораторной работы №9, перейдем в него и создадим нужный файл (рис. 1).



```
magryaznov@magryaznov:~/work/arch-pc/lab09
[magryaznov@magryaznov ~]$ mkdir ~/work/arch-pc/lab09
[magryaznov@magryaznov ~]$ cd ~/work/arch-pc/lab09
[magryaznov@magryaznov lab09]$ touch lab9-1.asm
```

рис. 1. Создание файла lab9-1.asm

При реализации циклов в NASM с использованием инструкции `loop` необходимо помнить о том, что эта инструкция использует регистр `ecx` в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра `ecx` (рис. 2).

```
magryaznov@magryaznov:~/work/arch-pc/lab9$ cat lab9-1.asm
[----] 6 L: [ 1+25 26/ 33] *(246 / 307b) 0010 0x
#include 'in_out.asm'

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

section .bss
N: resb 20

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]

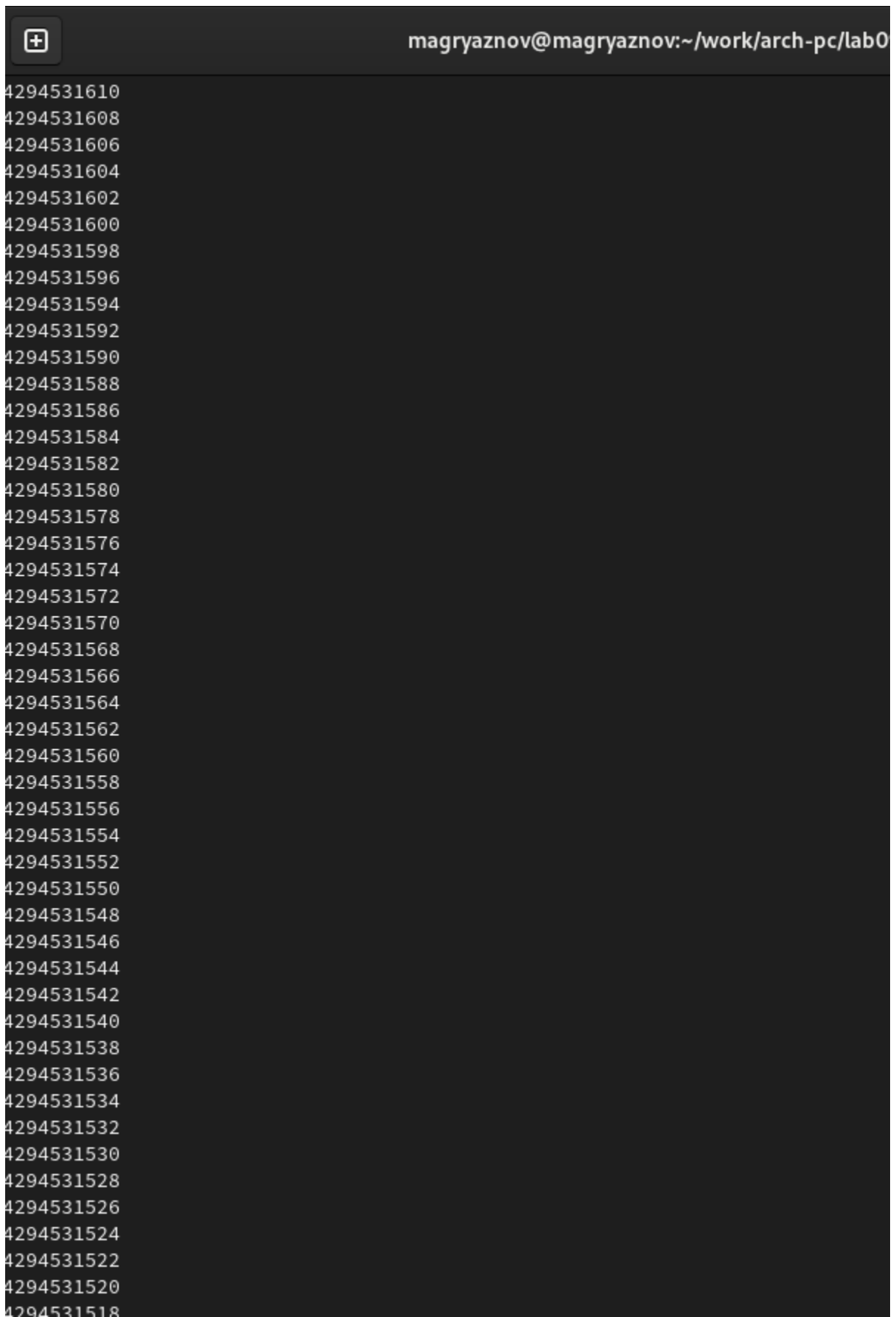
label:
mov [N],ecx
mov eax,[N]
call iprintLF

loop label

call quit
```

рис. 2. Текст программы lab9-1

Создадим исполняемый файл и проверим его работу (рис. 3).



```
magryaznov@magryaznov:~/work/arch-pc/lab0
4294531610
4294531608
4294531606
4294531604
4294531602
4294531600
4294531598
4294531596
4294531594
4294531592
4294531590
4294531588
4294531586
4294531584
4294531582
4294531580
4294531578
4294531576
4294531574
4294531572
4294531570
4294531568
4294531566
4294531564
4294531562
4294531560
4294531558
4294531556
4294531554
4294531552
4294531550
4294531548
4294531546
4294531544
4294531542
4294531540
4294531538
4294531536
4294531534
4294531532
4294531530
4294531528
4294531526
4294531524
4294531522
4294531520
4294531518
```

рис. 3. Результат работы программы lab9-1

Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Изменим текст программы добавив изменение значение регистра `ecx` в цикле по следующему примеру (рис. 4).

```
mov ecx,[N]

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

loop label

call quit
```

рис. 4. Пример изменения части программы lab9-1

Создадим исполняемый файл и проверим его работу (рис. 5).

```
[magryaznov@magryaznov lab09]$ nasm -f elf lab9-1.asm
[magryaznov@magryaznov lab09]$ ld -m elf_i386 lab9-1.o -o lab9-1
[magryaznov@magryaznov lab09]$ ./lab9-1
Введите N: 3
3
2
1
```

рис. 5. Результат работы измененной программы lab9-1

Как видим, все работает. Регистр `ecx` принимает все значения от `N` до 1 включительно, что соответствует числу проходов цикла, введенному с клавиатуры.

Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Внесем изменения в текст программы по примеру, добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop` (рис. 6).

```

mov ecx,[N]

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

call quit

```

рис. 6. Внесение команд *push* и *pop* в текст программы *lab9-1*

Создадим исполняемый файл и проверим его работу (рис. 7).

```

[magryaznov@magryaznov lab09]$ nasm -f elf lab9-1.asm
[magryaznov@magryaznov lab09]$ ld -m elf_i386 lab9-1.o -o lab9-1
[magryaznov@magryaznov lab09]$ ./lab9-1
Введите N: 5
5
4
3
2
1
[magryaznov@magryaznov lab09]$ 

```

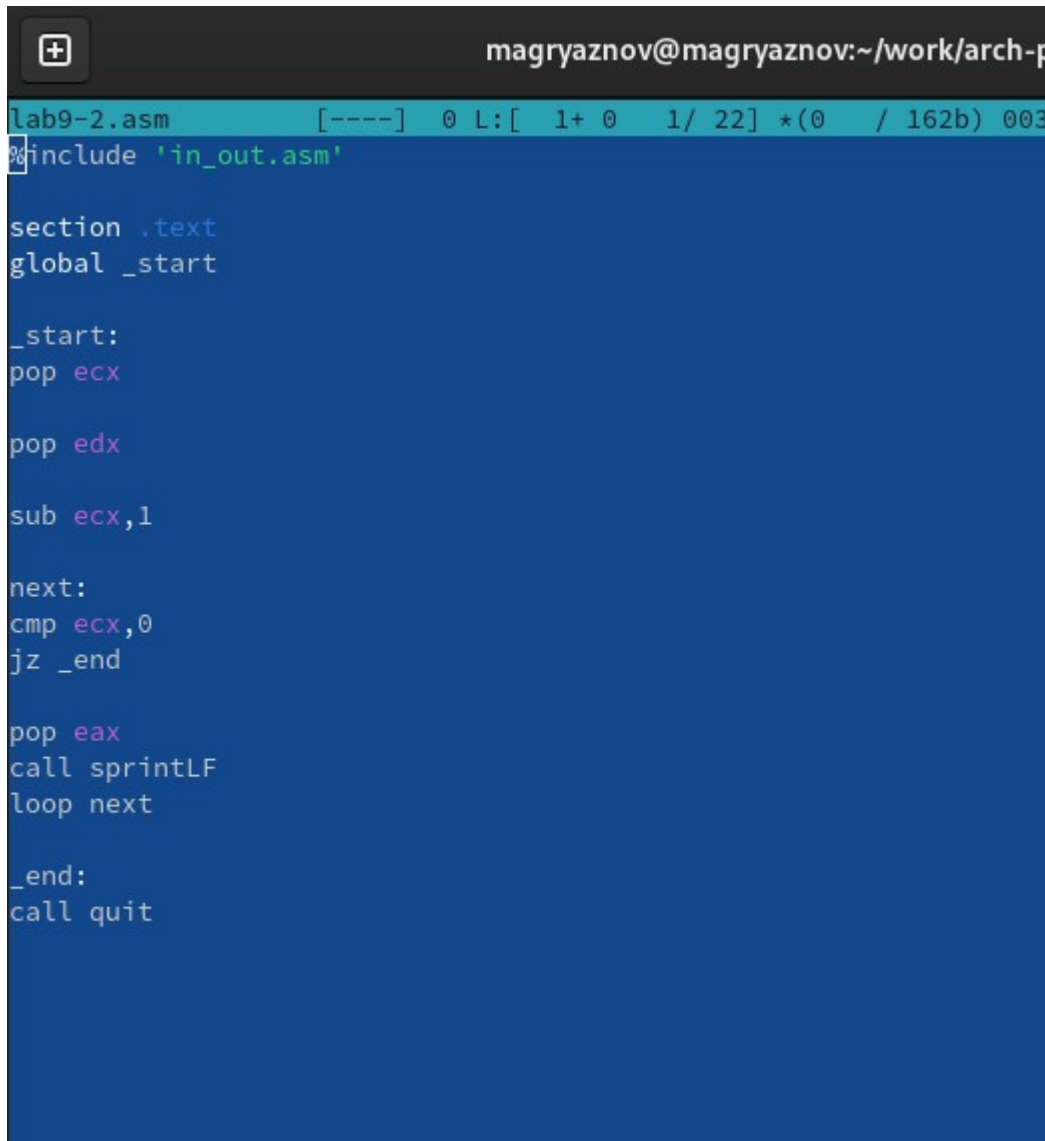
рис. 7. Результат работы измененной программы *lab9-1*

Как видим, программа работает корректно, число проходов цикла соответствует значению N, введенному с клавиатуры.

### Обработка аргументов командной строки.

При разработке программ иногда встает необходимость указывать аргументы, которые будут использоваться в программе, непосредственно из командной строки при запуске программы. При запуске программы в NASM аргументы командной строки загружаются в стек в обратном порядке, кроме того в стек записывается имя программы и общее количество аргументов. Последние два элемента стека для программы, скомпилированной NASM, – это всегда имя программы и количество переданных аргументов. Таким образом, для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку

аргументов нужно проводить в цикле. Т.е. сначала нужно извлечь из стека количество аргументов, а затем циклично для каждого аргумента выполнить логику программы. В качестве примера рассмотрим программу, которая выводит на экран аргументы командной строки (рис. 8). Создадим в каталоге лабораторной работы №9 файл lab9-2 и введем текст из рис. 8.



```
lab9-2.asm [-----] 0 L: [ 1+ 0 1/ 22] *(0 / 162b) 003
%include 'in_out.asm'

section .text
global _start

_start:
pop ecx

pop edx

sub ecx,1

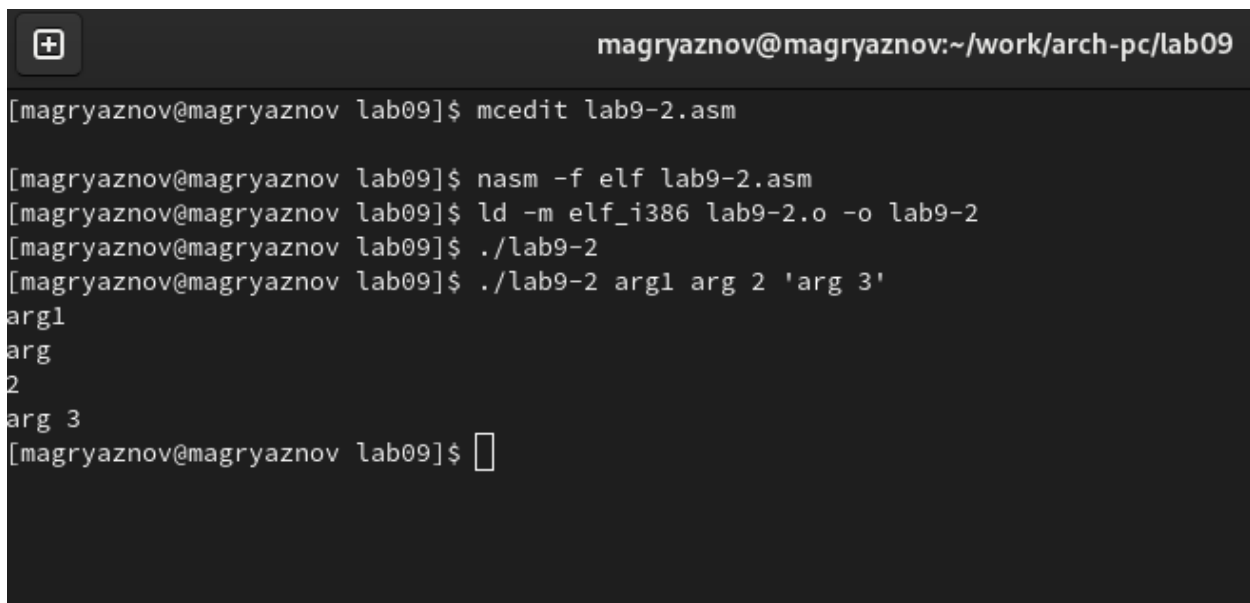
next:
cmp ecx,0
jz _end

pop eax
call sprintf
call printf
loop next

_end:
call quit
```

рис. 8. Текст программы lab9-2

Затем создадим исполняемый файл и запустим программу, указав следующие аргументы (рис. 9).



```
magryaznov@magryaznov:~/work/arch-pc/lab09
[magryaznov@magryaznov lab09]$ mcedit lab9-2.asm
[magryaznov@magryaznov lab09]$ nasm -f elf lab9-2.asm
[magryaznov@magryaznov lab09]$ ld -m elf_i386 lab9-2.o -o lab9-2
[magryaznov@magryaznov lab09]$ ./lab9-2
[magryaznov@magryaznov lab09]$ ./lab9-2 arg1 arg 2 'arg 3'
arg1
arg
2
arg 3
[magryaznov@magryaznov lab09]$
```

*рис. 9. Результат работы программы lab9-2*

Как видим, программа восприняла “аргумент” и “2” как отдельные аргументы, в то время как ‘аргумент 3’ как один. Соответственно программой было обработано 4 аргумента.

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. Создадим файл lab9-3.asm в том же каталоге и введем в него следующий текст программы (рис. 10).



```
magryaznov@magryaznov:~/work/arch-pc/
lab9-3.asm [-M--] 11 L: [ 1+23 24/ 33] *(203 / 283b) 0010
#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. Текст программы lab9-3

Затем создадим исполняемый файл и запустим его, указав аргументы (рис. 11).



```
[magryaznov@magryaznov lab09]$ mcedit lab9-3.asm  
  
[magryaznov@magryaznov lab09]$ nasm -f elf lab9-3.asm  
[magryaznov@magryaznov lab09]$ ld -m elf_i386 lab9-3.o -o lab9-3  
[magryaznov@magryaznov lab09]$ ./lab9-3 1 2 3 4  
Результат: 10  
[magryaznov@magryaznov lab09]$
```

*рис. 11. Результат работы программы lab9-3*

Как видим, все работает корректно.

Изменим строку

*add esi,ecx*

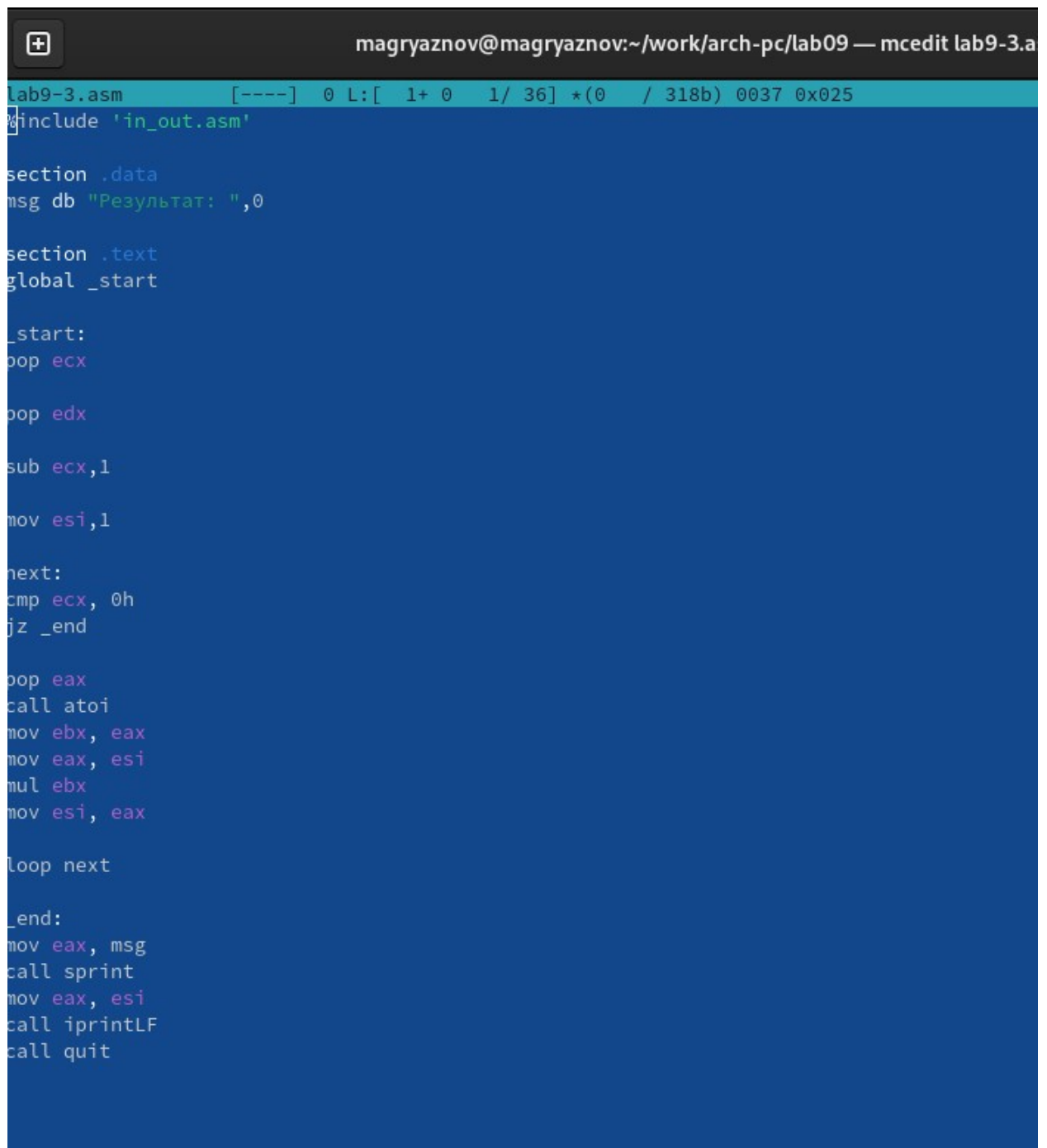
на

*move ebx, eax*

*mov eax, esi*

*mul ecx*

*mov esi, eax*



```
lab9-3.asm [----] 0 L:[ 1+ 0 1/ 36] *(0 / 318b) 0037 0x025
#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
mov ebx, eax
mov eax, esi
mul ebx
mov esi, eax

loop next

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

а также присвоим esi значение 1, чтобы программа выводила произведение аргументов командной строки и запустим ее (рис. 12).

```
[magryaznov@magryaznov lab09]$ mcedit lab9-3.asm

[magryaznov@magryaznov lab09]$ nasm -f elf lab9-3.asm
[magryaznov@magryaznov lab09]$ ld -m elf_i386 lab9-3.o -o lab9-3
[magryaznov@magryaznov lab09]$ ./lab9-3
Результат: 1
[magryaznov@magryaznov lab09]$ ./lab9-3 1 2 3 4
Результат: 24
[magryaznov@magryaznov lab09]$ □
```

*рис. 12. Результат работы программы*

### **Порядок выполнения самостоятельной работы:**

Напишем 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  $f(x)$  выберем в соответствии с вариантом, полученным при выполнении лабораторной работы № 7 (вариант 15). Создадим исполняемый файл и проверим его работу на нескольких наборах  $x = x_1, x_2, \dots, x_n$ .

$$f(x) = 6x + 13$$



```
lab9-sm.asm [-----] 0 L: [ 1+ 0 1/ 39] *(0 /
#include 'in_out.asm'

section .data
msg db "Результат: ",0
msgf db "Функция: f(x)=6x+13",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
mov ebx,6
mul ebx
add eax,13
add esi,eax

loop next

_end:
mov eax, msgf
call sprintfLF
mov eax, msg
call sprintf
mov eax, esi
call iprintLF
call quit
```

рис. 13. Текст программы lab9-sm

```
[magryaznov@magryaznov lab09]$ mcedit lab9-sm.asm

[magryaznov@magryaznov lab09]$ nasm -f elf lab9-sm.asm
[magryaznov@magryaznov lab09]$ ld -m elf_i386 lab9-sm.o -o lab9-sm
[magryaznov@magryaznov lab09]$ ./lab9-sm
Функция:  $f(x)=6x+13$ 
Результат: 0
[magryaznov@magryaznov lab09]$ ./lab9-sm 1
Функция:  $f(x)=6x+13$ 
Результат: 19
[magryaznov@magryaznov lab09]$ ./lab9-sm 1 2
Функция:  $f(x)=6x+13$ 
Результат: 44
[magryaznov@magryaznov lab09]$ ./lab9-sm 1 2 3
Функция:  $f(x)=6x+13$ 
Результат: 75
[magryaznov@magryaznov lab09]$
```

рис. 14. Результат работы программы lab9-sm

Все работает корректно.

### **Вывод:**

Во время выполнения лабораторной работы были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.