

# **Отчет по лабораторной работе No.8**

**Дисциплины: Архитектура компьютера**

Нджову Нелиа

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>16</b>
	<b>Список литературы</b>	<b>17</b>

# Список иллюстраций

3.1	Рис 1	7
3.2	Рис 2	8
3.3	Рис 3	8
3.4	Рис 4	9
3.5	Рис 5	9
3.6	Рис 6	10
3.7	Рис 7	10
3.8	Рис 8	10
3.9	Рис 9	11
3.10	Рис 10	11
3.11	Рис 11	12
3.12	Рис 12	12
3.13	Рис 13	12
3.14	Рис 14	13
3.15	Рис 15	13
3.16	Рис 16	13
3.17	Рис 17	14
3.18	Рис 18	14

## Список таблиц

# 1 Цель работы

Целью лабораторной работы является приобретение навыков написания программ с использованием циклов и обработки аргументов командной строки.

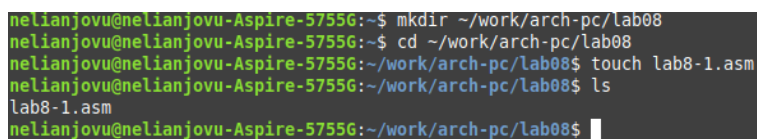
## 2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Задание для самостоятельной работы

## 3 Выполнение лабораторной работы

### 1. Реализация циклов в NASM

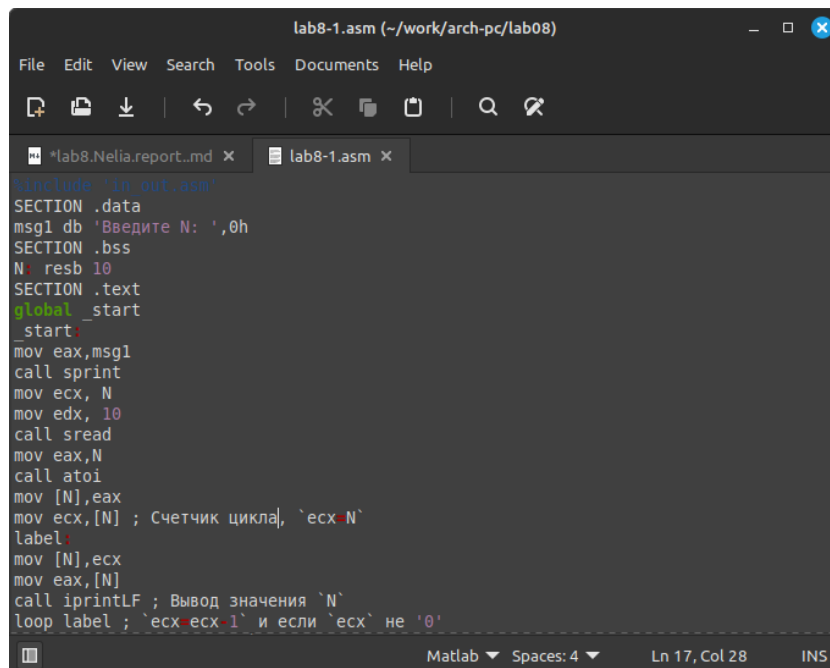
Я создам каталог для программ лабораторных работ 8, зайду в него и создам файл lab8-1.asm(рис 1)



```
nelianjovu@nelianjovu-Aspire-5755G:~$ mkdir ~/work/arch-pc/lab08
nelianjovu@nelianjovu-Aspire-5755G:~$ cd ~/work/arch-pc/lab08
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$ touch lab8-1.asm
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$ ls
lab8-1.asm
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$
```

Рис. 3.1: Рис 1

При реализации циклов в NASM с помощью инструкции цикла важно помнить, что эта инструкция использует регистр esx в качестве счетчика и уменьшает его значение на единицу на каждом шаге. Теперь я открою созданный мной файл, затем скопирую и изучу текст данной программы(рис 2)



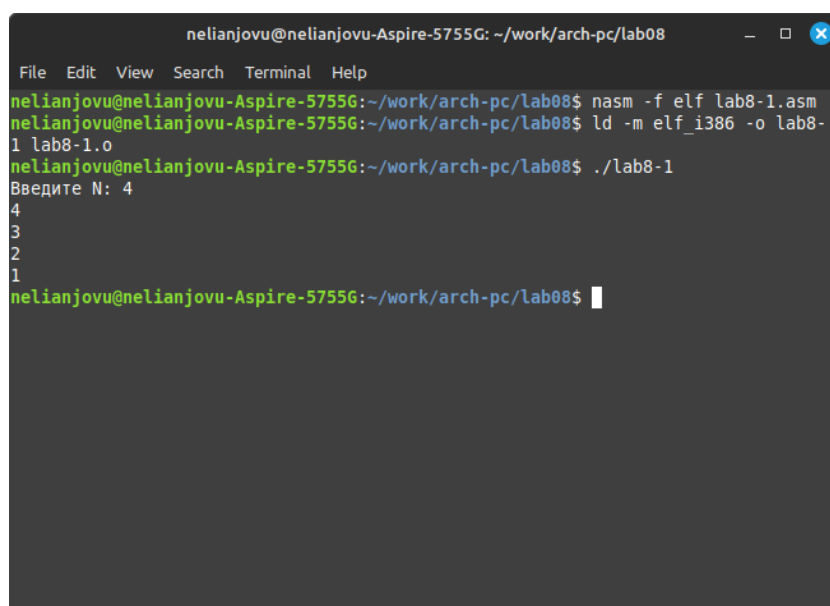
The screenshot shows a text editor window titled 'lab8-1.asm (~/.work/arch-pc/lab08)'. The code is as follows:

```
%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] ; 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'
```

The status bar at the bottom indicates 'Matlab', 'Spaces: 4', 'Ln 17, Col 28', and 'INS'.

Рис. 3.2: Рис 2

Теперь я создам исполняемый файл и запущу его(рис 3)



The screenshot shows a terminal window with the following commands and output:

```
nelianjovu@nelianjovu-Aspire-5755G: ~/.work/arch-pc/lab08
nelianjovu@nelianjovu-Aspire-5755G:~/.work/arch-pc/lab08$ nasm -f elf lab8-1.asm
nelianjovu@nelianjovu-Aspire-5755G:~/.work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
nelianjovu@nelianjovu-Aspire-5755G:~/.work/arch-pc/lab08$ ./lab8-1
Введите N: 4
4
3
2
1
nelianjovu@nelianjovu-Aspire-5755G:~/.work/arch-pc/lab08$
```

Рис. 3.3: Рис 3

Я изменю текст программы, меняя в цикле значение регистра ecx(рис 4)



```

lab8-1.asm (~work/arch-pc/lab08)
File Edit View Search Tools Documents Help
*lab8.Nelia.report..md x lab8-1.asm 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] ; Счетчик цикла, `ecx`-N
label:
sub ecx,1 ; `ecx`-ecx-1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`

```

Рис. 3.4: Рис 4

Я создам исполняемый файл и проверю его работу(рис 5)

```

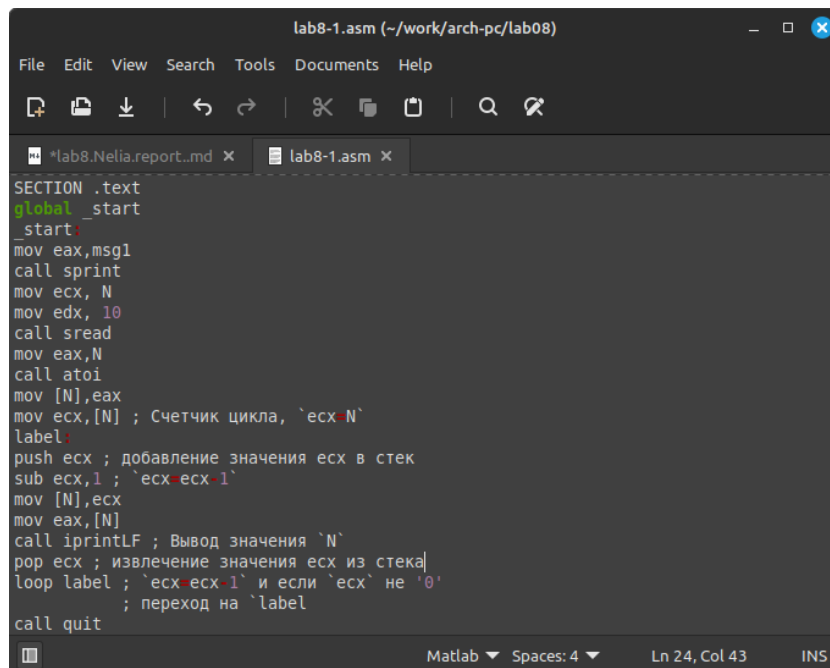
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$

```

Рис. 3.5: Рис 5

*Когда я запускаю программу, она отображает значения 3 и 1, количество циклов не соответствует значению n*

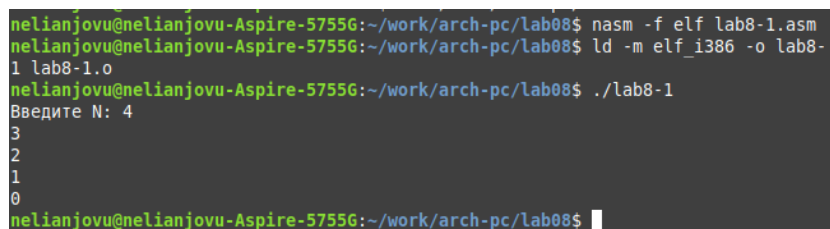
Чтобы использовать регистр `ecx` в цикле и обеспечить правильную работу программы, мне нужно использовать стек. Поэтому я внесу изменения в текст программы, добавив команды `push` и `pop` (добавление в стек и извлечение из стека), чтобы сохранить значение счетчика цикла(рис 6)



```
lab8-1.asm (~/work/arch-pc/lab08)
File Edit View Search Tools Documents Help
*lab8.Nelia.report.md x lab8-1.asm x
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] ; Счетчик цикла, `ecx`-N
label:
push ecx ; добавление значения ecx в стек
sub ecx,1 ; `ecx`-ecx-1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
pop ecx ; извлечение значения ecx из стека
loop label ; `ecx`-ecx-1 и если `ecx` не '0'
; переход на `label`
call quit
Matlab Spaces: 4 Ln 24, Col 43 INS
```

Рис. 3.6: Рис 6

Я создам исполняемый файл и проверю его работу(рис 7)



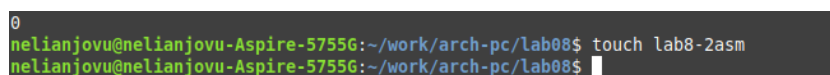
```
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
2
1
0
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$
```

Рис. 3.7: Рис 7

*В этом случае количество проходов цикла соответствует значению N, введенному с клавиатуры*

## 2. Обработка аргументов командной строки

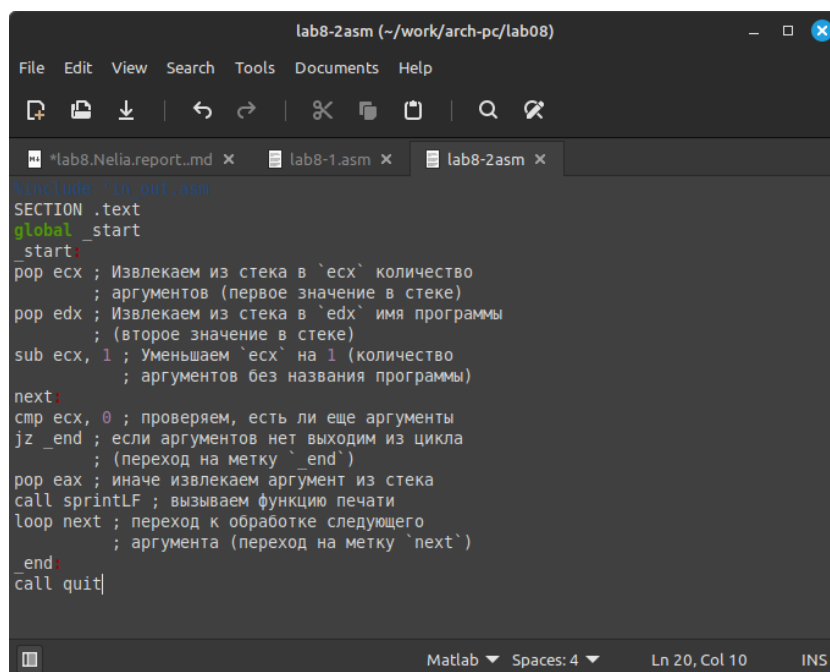
Я создам файл lab8-2.asm с помощью команды touch(рис 8)



```
0
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$ touch lab8-2asm
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$
```

Рис. 3.8: Рис 8

Когда вы запускаете программу, аргументы располагаются в стеке, поэтому, чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Аргументы должны обрабатываться в цикле. Сначала вам нужно извлечь количество аргументов из стека, а затем просмотреть логику программы для каждого аргумента. Чтобы показать это, я скопирую данную программу в файл, который я только что создал(рис 9)

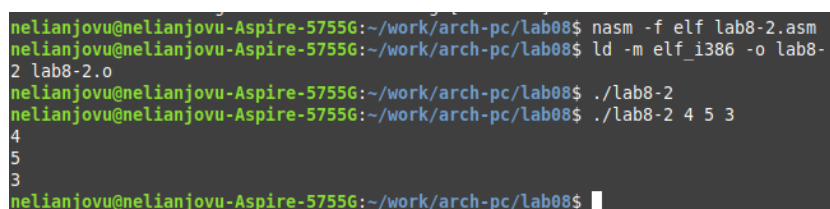


```

lab8-2asm (~/.work/arch-pc/lab08)
File Edit View Search Tools Documents Help
*lab8.Nelia.report.md x lab8-1.asm x lab8-2asm x
#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 printf ; вызываем функцию печати
    loop next ; переход к обработке следующего
             ; аргумента (переход на метку `next`)
_end:
    call quit
    
```

Рис. 3.9: Рис 9

Я создам исполняемый файл и проверю его работу(рис 10)



```

nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$ ./lab8-2
4
5
3
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$
    
```

Рис. 3.10: Рис 10

*Я ввела три аргумента, и программа обработала количество введенных мной*

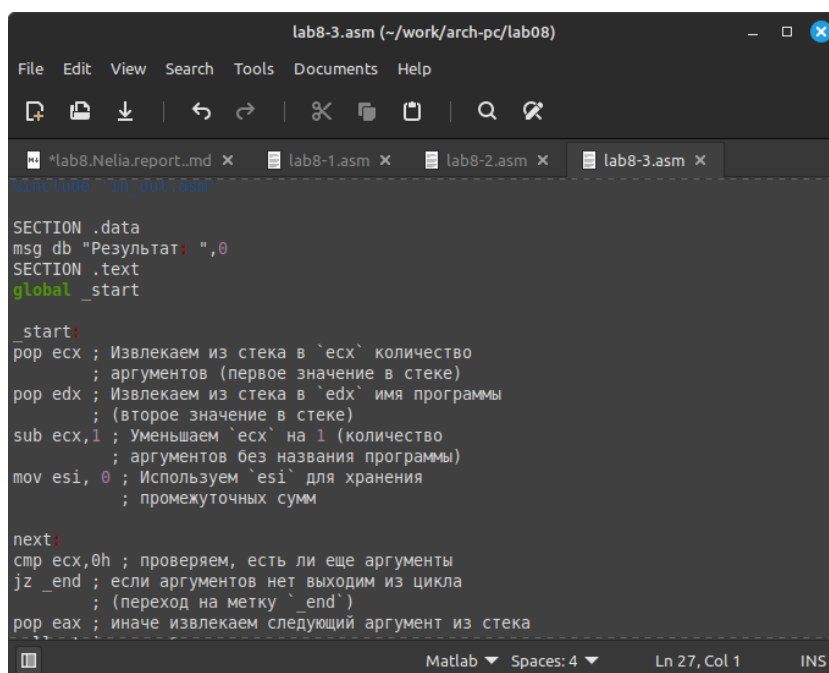
аргументов.

Я создам файл lab8-2.asm с помощью команды touch(рис 11)

```
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$ touch lab8-3.asm
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$
```

Рис. 3.11: Рис 11

Я открою его и скопирую в него заданную программу, программа отображает сумму чисел, которые передаются программе в качестве аргументов(рис 12)



```
lab8-3.asm (~/.work/arch-pc/lab08)
File Edit View Search Tools Documents Help
*lab8.Nelia.report..md x lab8-1.asm x lab8-2.asm x lab8-3.asm x
%include "lab_out.asm"

SECTION .data
msg db "Результат: ",0
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)
    mov esi, 0 ; Используем `esi` для хранения
             ; промежуточных сумм

next:
    cmp ecx,0h ; проверяем, есть ли еще аргументы
    jz _end ; если аргументов нет выходим из цикла
             ; (переход на метку `_end`)
    pop eax ; иначе извлекаем следующий аргумент из стека
```

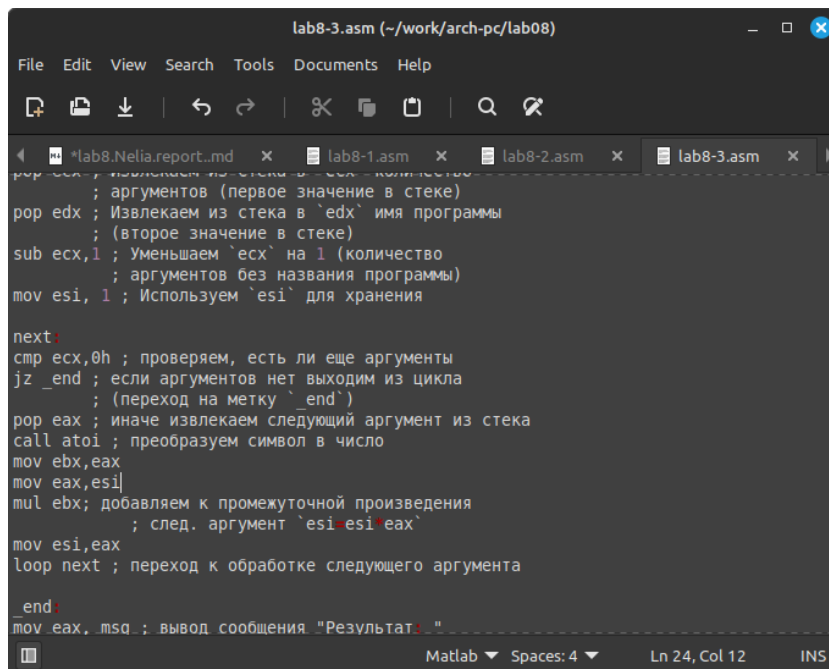
Рис. 3.12: Рис 12

Я создам исполняемый файл и проверю его работу(рис 13)

```
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$ ./lab8-3 4 5 3
Результат: 12
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$
```

Рис. 3.13: Рис 13

Я изменю программу так, чтобы она вычисляла произведение аргументов командной строки(рис 14)



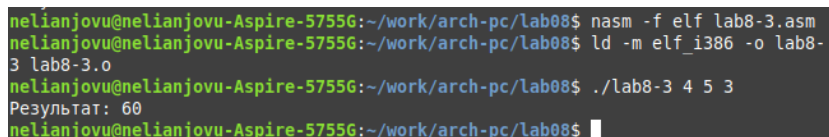
```
lab8-3.asm (~/.work/arch-pc/lab08)
File Edit View Search Tools Documents Help
[Icons]
lab8.Nelia.report.md x lab8-1.asm x lab8-2.asm x lab8-3.asm x
; аргументов (первое значение в стеке)
pop ecx ; Извлекаем из стека в `ecx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 1 ; Используем `esi` для хранения

next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
mov ebx,eax
mov eax,esi
mul ebx; добавляем к промежуточной произведения
; след. аргумент `esi*esi*eax`
mov esi,eax
loop next ; переход к обработке следующего аргумента

_end
mov eax, _msg ; вывод сообщения "Результат: "
; -----
```

Рис. 3.14: Рис 14

Я создам исполняемый файл и проверю его работу(рис 15)

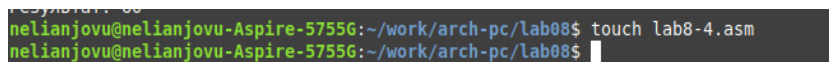


```
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$ ./lab8-3 4 5 3
Результат: 60
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$
```

Рис. 3.15: Рис 15

### 3. Задание для самостоятельной работы

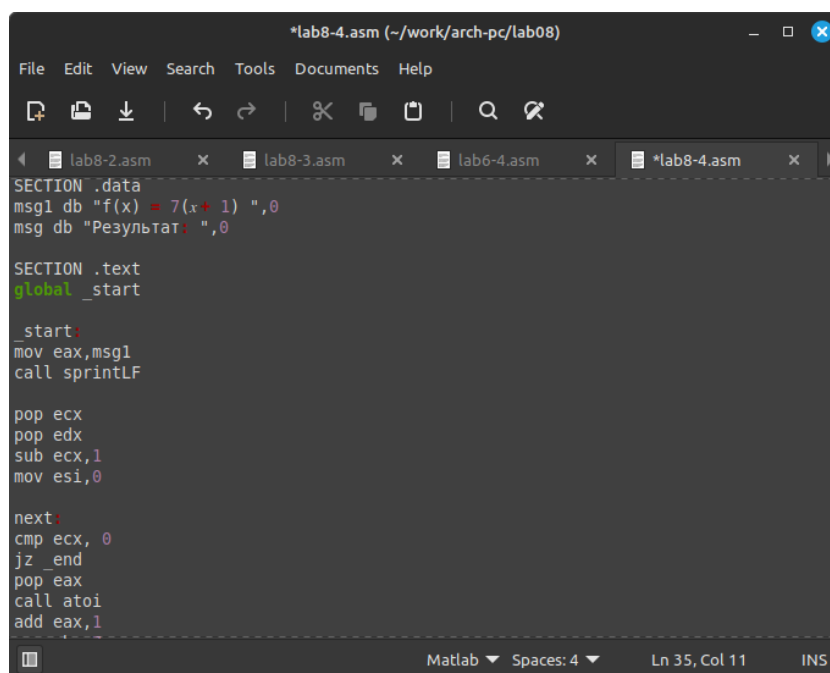
Я создам файл lab8-4.asm с помощью команды touch(рис 16)



```
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$ touch lab8-4.asm
nelianjovu@nelianjovu-Aspire-5755G:~/work/arch-pc/lab08$
```

Рис. 3.16: Рис 16

В созданном мной файле я напишу программу, которая находит сумму значений функции  $f(x)$  для  $x = x_1, x_2, x_3, \dots$  и т. д. Программа должна вывести значение  $f(x_1) + f(x_2) + \dots + f(x_n)$ . Значения  $x$  передаются в качестве аргументов. Тип функции  $f(x)$  я выберу из данной таблицы вариантов задания в соответствии с вариантом, полученным мной в ходе лабораторной работы 6. Мой вариант — вариант 14;  $f(x) = 7(x + 1)$  (рис 17)



```

*lab8-4.asm (~/.work/arch-pc/lab08)
File Edit View Search Tools Documents Help
lab8-2.asm lab8-3.asm lab6-4.asm *lab8-4.asm
SECTION .data
msg1 db "f(x) = 7(x + 1) ",0
msg db "Результат: ",0

SECTION .text
global _start

_start:
mov eax,msg1
call sprintf

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

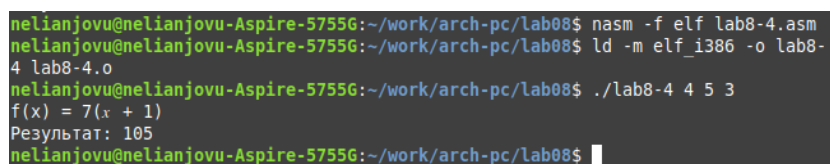
next:
cmp ecx, 0
jz _end
pop eax
call atoi
add eax,1

_end:

```

Рис. 3.17: Рис 17

Я создам исполняемый файл и проверю его работу(рис 18)



```

nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$ ./lab8-4 4 5 3
f(x) = 7(x + 1)
Результат: 105
nelianjovu@nelianjovu-Aspire-57556:~/work/arch-pc/lab08$

```

Рис. 3.18: Рис 18

*Текстовая программа для самостоятельной работы*

%include 'in\_out.asm'

```

SECTION .data
msg1 db "f(x) = 7(x + 1) ",0
msg db "Результат: ",0
SECTION .text
global _start
_start:
mov eax,msg1
call sprintfLF
pop ecx
pop edx
sub ecx,1
mov esi,0
next:
cmp ecx, 0
jz _end
pop eax
call atoi
add eax,1
mov ebx,7
mul ebx
add esi,eax
loop next
_end:
mov eax, msg
call sprintf
mov eax, esi
call sprintfLF
call quit

```

## 4 Выводы

Выполняя эту лабораторную работу, я приобрел навыки написания программ с использованием циклов и обработки аргументов командной строки.



# Список литературы

Архитектура ЭВМ