

# **Отчет по лабораторной работе No 8**

**Дисциплины: Архитектура компьютера**

Ракутуманандзара Цантамписедрана Сарубиди

# 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>17</b>
	<b>Список литературы</b>	<b>18</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 . . . . .	13
3.14	рис 14 . . . . .	13
3.15	рис 15 . . . . .	13
3.16	рис 16 . . . . .	14
3.17	рис 17 . . . . .	14

## **Список таблиц**

# 1 Цель работы

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

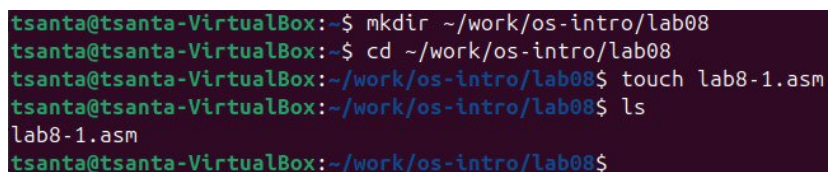
## 2 Задание

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

## 3 Выполнение лабораторной работы

### 1. Реализация циклов в NASM

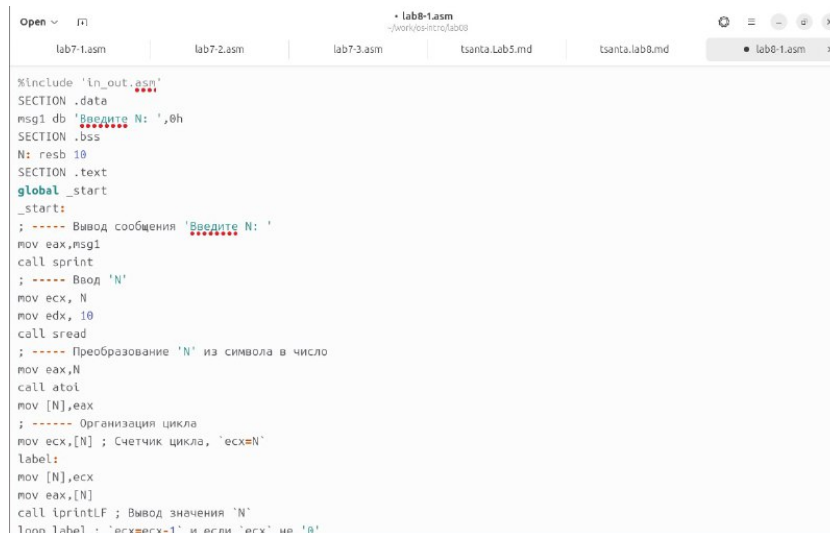
Я создам каталог для программ лабораторных работ 8, зайду в него и создам файл lab8-1.asm(рис 1)

A screenshot of a terminal window with a dark purple background. The text is as follows:

```
tsanta@tsanta-VirtualBox:~$ mkdir ~/work/os-intro/lab08
tsanta@tsanta-VirtualBox:~$ cd ~/work/os-intro/lab08
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ touch lab8-1.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ls
lab8-1.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$
```

Рис. 3.1: рис 1

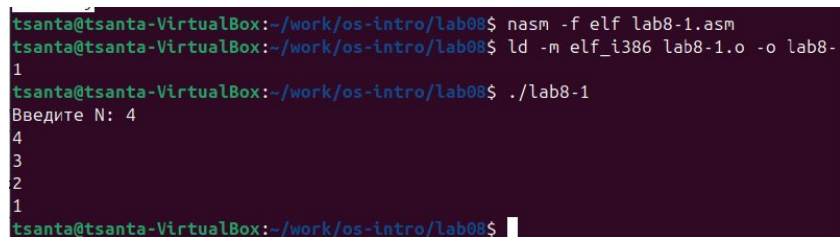
При реализации циклов в NASM с помощью инструкции цикла важно помнить, что эта инструкция использует регистр esx в качестве счетчика и уменьшает его значение на единицу на каждом шаге. Теперь я открою созданный мной файл, затем скопирую и изучу текст данной программы(рис 2)



```
%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 read
; ----- 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 iprintf ; Вывод значения 'N'
loop label ; "ecx=ecx-1" и если "ecx" не '0'
```

Рис. 3.2: рис 2

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

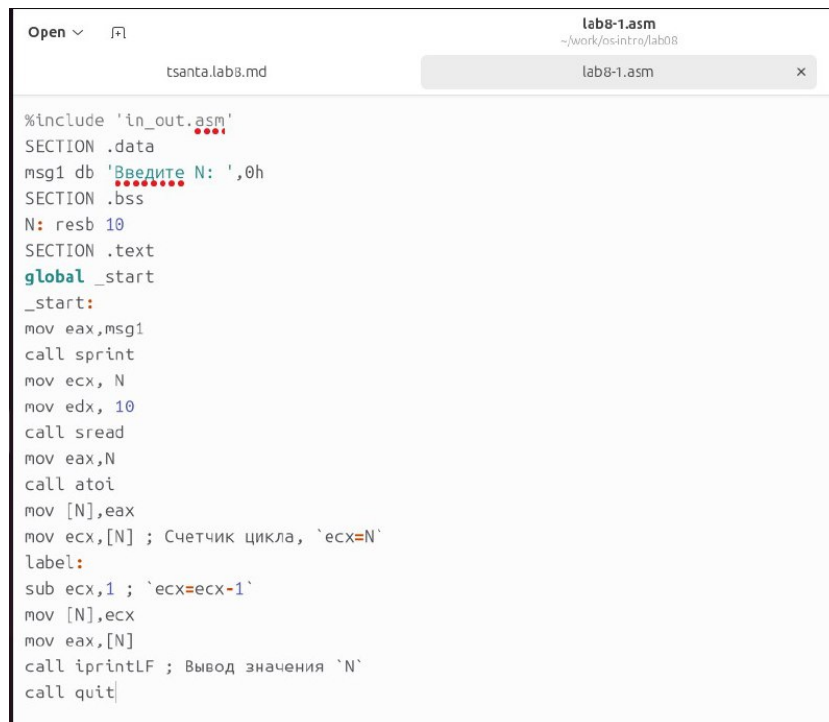


```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ nasm -f elf lab8-1.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ./lab8-1
Введите N: 4
4
3
2
1
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$
```

Рис. 3.3: рис 3

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



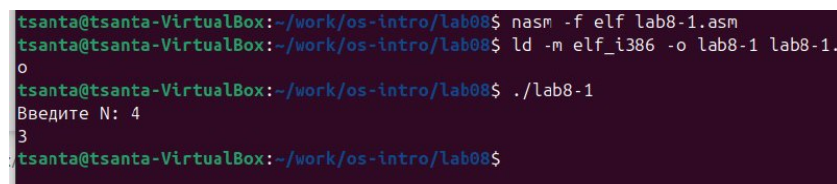


```
Open  tsanta.lab8.md  lab8-1.asm  x
~/work/osintro/lab08
lab8-1.asm

%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`
call quit
```

Рис. 3.4: рис 4

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

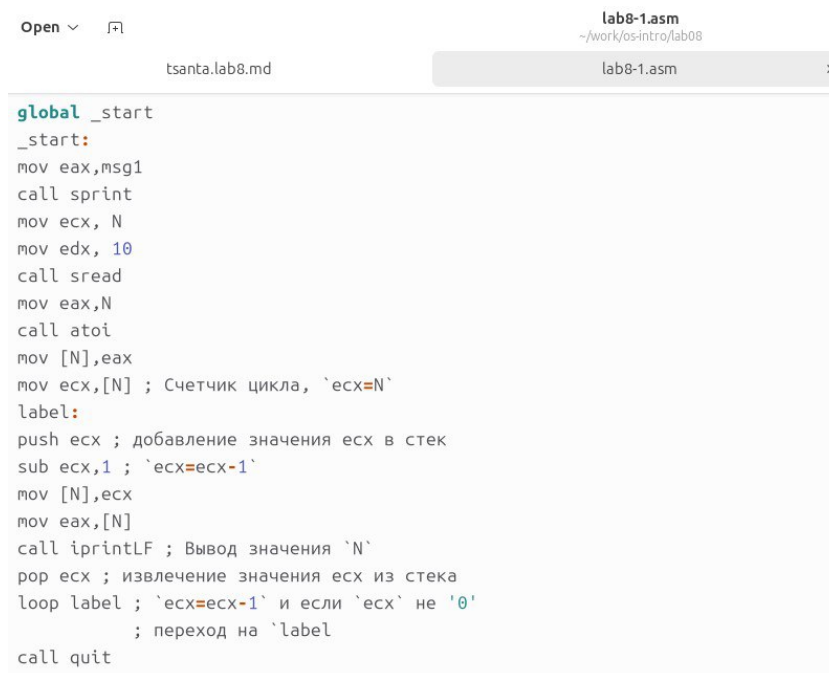


```
tsanta@tsanta-VirtualBox: ~/work/os-intro/lab08$ nasm -f elf lab8-1.asm
tsanta@tsanta-VirtualBox: ~/work/os-intro/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
tsanta@tsanta-VirtualBox: ~/work/os-intro/lab08$ ./lab8-1
Введите N: 4
3
tsanta@tsanta-VirtualBox: ~/work/os-intro/lab08$
```

Рис. 3.5: рис 5

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

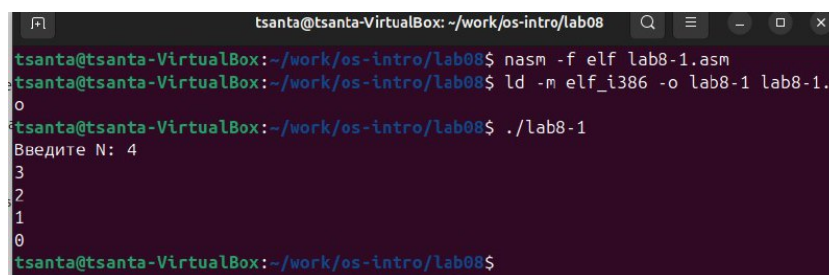
Чтобы использовать регистр ecx в цикле и обеспечить правильную работу программы, мне нужно использовать стек. Поэтому я внесу изменения в текст программы, добавив команды push и pop (добавление в стек и извлечение из стека), чтобы сохранить значение счетчика цикла(рис 6)



```
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
```

Рис. 3.6: рис 6

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



```
tsanta@tsanta-VirtualBox: ~/work/os-intro/lab08
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ nasm -f elf lab8-1.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ./lab8-1
Введите N: 4
3
2
1
0
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$
```

Рис. 3.7: рис 7

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

## 2. Обработка аргументов командной строки

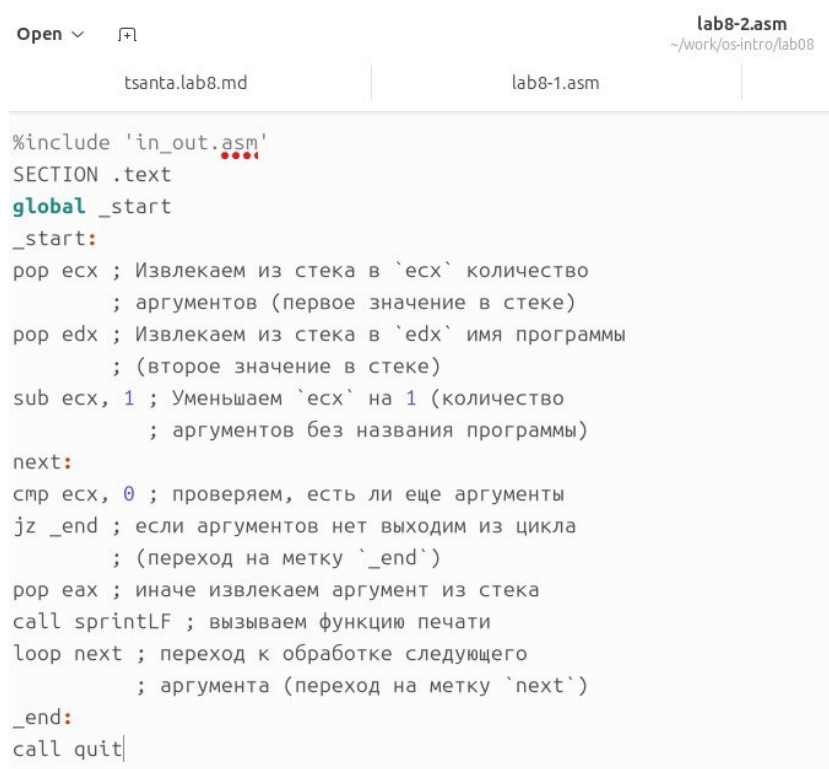
Я создам файл lab8-2.asm с помощью команды touch(рис 8)



```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ touch lab8-2.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/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)



```

Open  ▾  [F1]
lab8-2.asm
~/work/os-intro/lab08

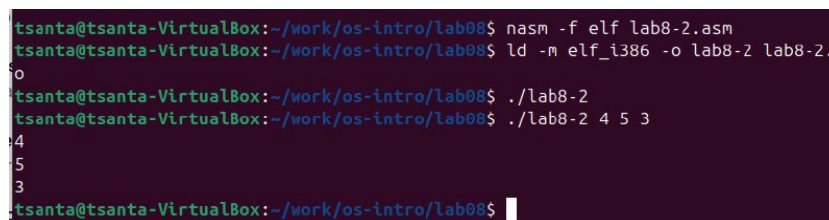
tsanta.lab8.md  lab8-1.asm

%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 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)



```

tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ nasm -f elf lab8-2.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ./lab8-2
4
5
3
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$

```

Рис. 3.10: рис 10

Я ввела три аргумента, и программа обработала количество введенных мной

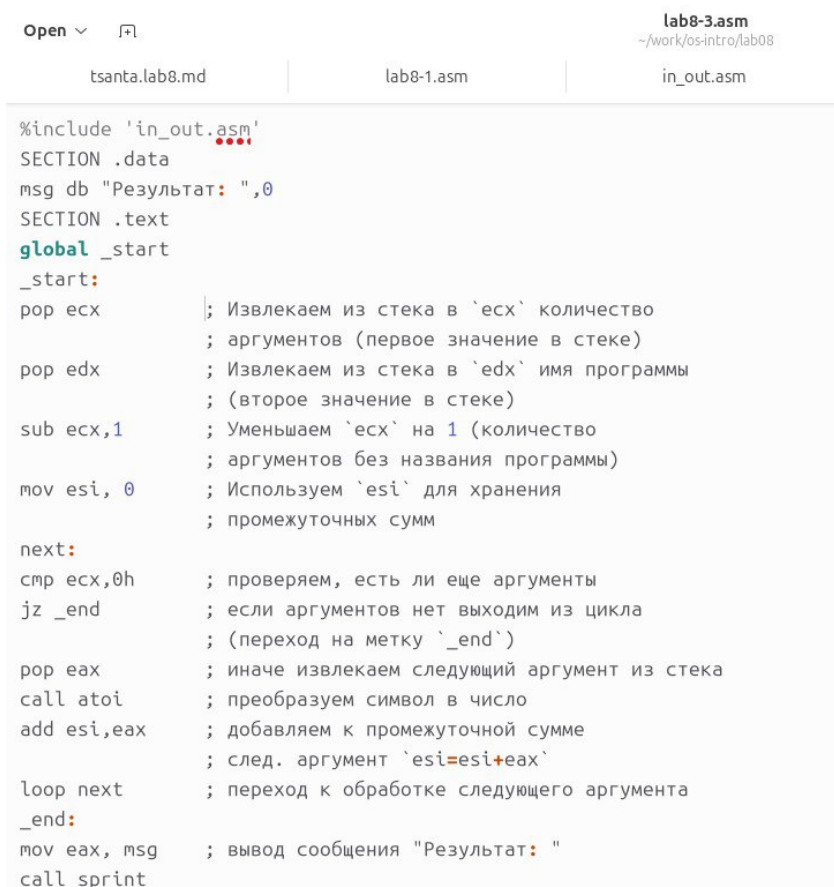
аргументов.

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

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ touch lab8-3.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/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)



```
Open ▾  tsanta.lab8.md  lab8-1.asm  lab8-3.asm
~/.work/os-intro/lab08
in_out.asm

%include 'in_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
call atoi    ; преобразуем символ в число
add esi,eax  ; добавляем к промежуточной сумме
              ; след. аргумент `esi=esi+eax`
loop next    ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
```


Рис. 3.12: рис 12

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

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ touch lab8-3.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ nasm -f elf lab8-3.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ./lab8-3 4 5 3
Результат: 12
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$
```

Рис. 3.13: рис 13

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



```
Open ▾  lab8-3.asm
~/work/os-intro/lab08
tsanta.lab8.md | lab8-1.asm | in_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,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 произведению
              ; след. аргумент `esi=esi*eax`
mov esi,eax
loop next    ; переход к обработке следующего аргумента

_end:
```

Рис. 3.14: рис 14

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

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ./lab8-3 4 5 3
Результат: 12
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ nasm -f elf lab8-3.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ./lab8-3 4 5 3
Результат: 60
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$
```

Рис. 3.15: рис 15

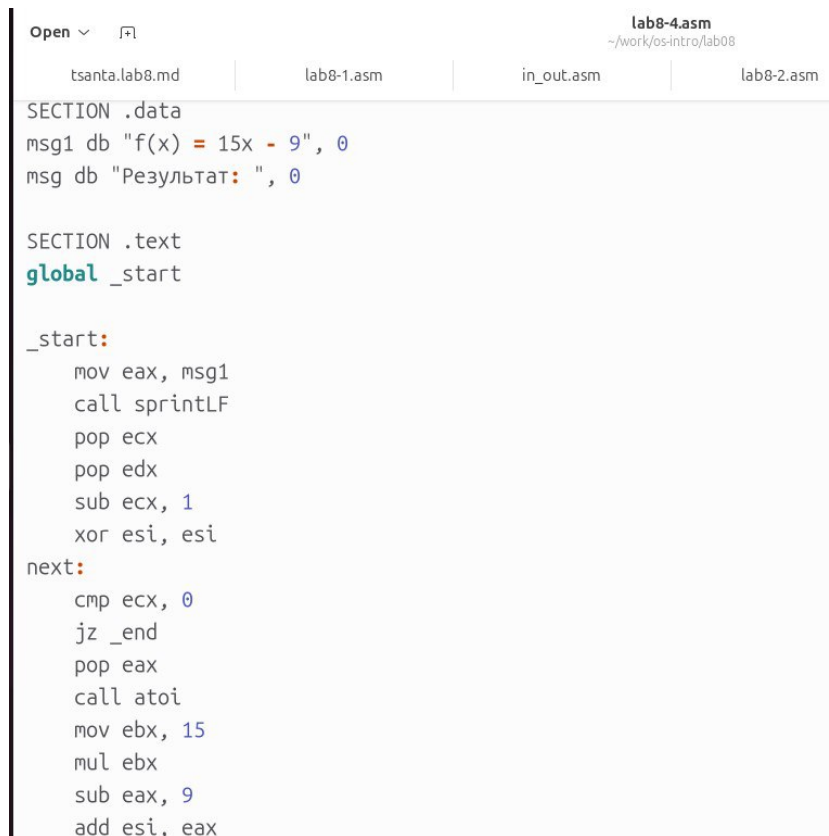
### 3. Задание для самостоятельной работы

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

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ touch lab8-4.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/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 12;  $f(x) = (15x-9)$ (рис 17)



```
Open ▾  lab8-4.asm
tsanta.lab8.md  lab8-1.asm  in_out.asm  lab8-2.asm
SECTION .data
msg1 db "f(x) = 15x - 9", 0
msg db "Результат: ", 0

SECTION .text
global _start

_start:
    mov eax, msg1
    call sprintf
    pop ecx
    pop edx
    sub ecx, 1
    xor esi, esi
next:
    cmp ecx, 0
    jz _end
    pop eax
    call atoi
    mov ebx, 15
    mul ebx
    sub eax, 9
    add esi, eax
```

Рис. 3.17: рис 17

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

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ nasm -f elf lab8-4.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$ ./lab8-4 4 5 3
f(x) = 15x - 9
Результат: 153
tsanta@tsanta-VirtualBox:~/work/os-intro/lab08$
```

{#fig:0018

width=70%

Текстовая программа для самостоятельной работы

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

```
SECTION .data
```

```
msg1 db "f(x) = 15x - 9", 0
```

```
msg db "Результат: ", 0
```

```
SECTION .text
```

```
global _start
```

```
_start:
```

```
    mov eax, msg1
```

```
    call sprintf
```

```
    pop ecx
```

```
    pop edx
```

```
    sub ecx, 1
```

```
    xor esi, esi
```

```
next:
```

```
    cmp ecx, 0
```

```
    jz _end
```

```
    pop eax
```

```
    call atoi
```

```
    mov ebx, 15
```

```
    mul ebx
```

```
    sub eax, 9
```

```
    add esi, eax
    dec ecx
    jmp next
_end:
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintLF
    call quit
```



## 4 Выводы

Здесь кратко описываются итоги проделанной работы. Выполняя эту лабораторную работу, я приобрел навыки написания программ с использованием циклов и обработки аргументов командной строки.

# **Список литературы**

Архитектура ЭВМ