

# **Отчёт по лабораторной работе №8**

**дисциплина: Архитектура компьютеров**

Ведьмина Александра Сереевна

# Содержание

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Цель работы</b>                                   | <b>5</b>  |
| <b>2</b> | <b>Задание</b>                                       | <b>6</b>  |
| <b>3</b> | <b>Теоретическое введение</b>                        | <b>7</b>  |
| <b>4</b> | <b>Выполнение лабораторной работы</b>                | <b>8</b>  |
| <b>5</b> | <b>Выполнение заданий для самостоятельной работы</b> | <b>16</b> |
| <b>6</b> | <b>Выводы</b>  | <b>17</b> |

## Список иллюстраций

|      |  |    |
|------|--|----|
| 4.1  | Создание lab8-1.asm . . . . .                        | 8  |
| 4.2  | Ввод программы в lab8-1.asm . . . . .                | 9  |
| 4.3  | Запуск lab8-1 . . . . .                              | 9  |
| 4.4  | Изменение программы в lab8-1.asm . . . . .           | 10 |
| 4.5  | Запуск изменённого lab8-1 . . . . .                  | 11 |
| 4.6  | Повторное изменение программы в lab8-1.asm . . . . . | 12 |
| 4.7  | Запуск вновь изменённого lab8-1 . . . . .            | 12 |
| 4.8  | Ввод программы в lab8-2.asm . . . . .                | 13 |
| 4.9  | Запуск lab8-2 . . . . .                              | 13 |
| 4.10 | Ввод программы в lab8-3.asm . . . . .                | 14 |
| 4.11 | Запуск lab8-3 . . . . .                              | 14 |
| 4.12 | Изменение программы в lab8-3.asm . . . . .           | 15 |
| 4.13 | Запуск изменённого lab8-3 . . . . .                  | 15 |
| 5.1  | Создание файла sumrub.asm . . . . .                  | 16 |
| 5.2  | Ввод программы в sumrub.asm . . . . .                | 16 |
| 5.3  | Запуск sumrub . . . . .                              | 16 |

## Список таблиц

# 1 Цель работы

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

## 2 Задание

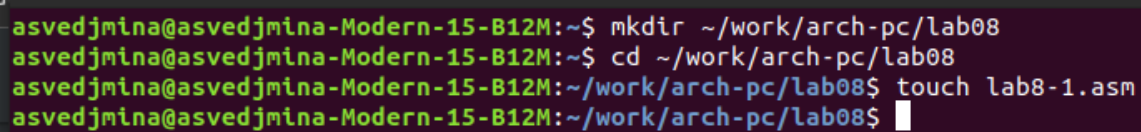
1. Изучить теорию по организации стека
2. Ознакомиться с реализацией циклов в NASM
3. Выполнить задание для самостоятельной работы

### 3 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO. Его основная функция - сохранение адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров. Вершина стека - адрес последнего добавленного элемента, противоположный конец стека именуется дном. В работе со стеками есть две основные операции: добавление элемента в вершину, извлечение элемента из вершины.

## 4 Выполнение лабораторной работы

Создаю каталог lab08, перехожу в него и создаю там файл lab8-1.asm.

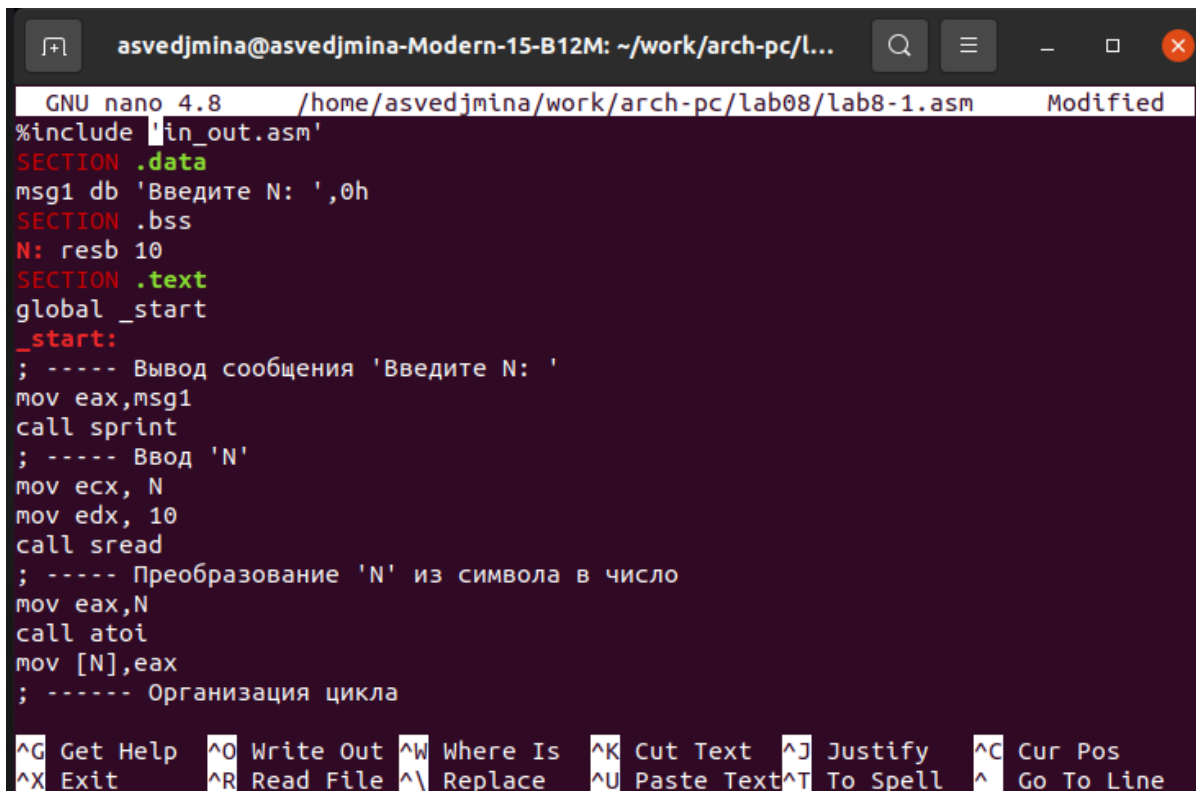
A screenshot of a terminal window with a dark purple background and green text. It shows four lines of commands and their execution: 'mkdir ~/work/arch-pc/lab08', 'cd ~/work/arch-pc/lab08', 'touch lab8-1.asm', and a prompt for the next command.

```
asvedjmina@asvedjmina-Modern-15-B12M:~$ mkdir ~/work/arch-pc/lab08
asvedjmina@asvedjmina-Modern-15-B12M:~$ cd ~/work/arch-pc/lab08
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ touch lab8-1.asm
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$
```

Рис. 4.1: Создание lab8-1.asm

В созданный файл ввожу текст программы вывода значений регистра есх.



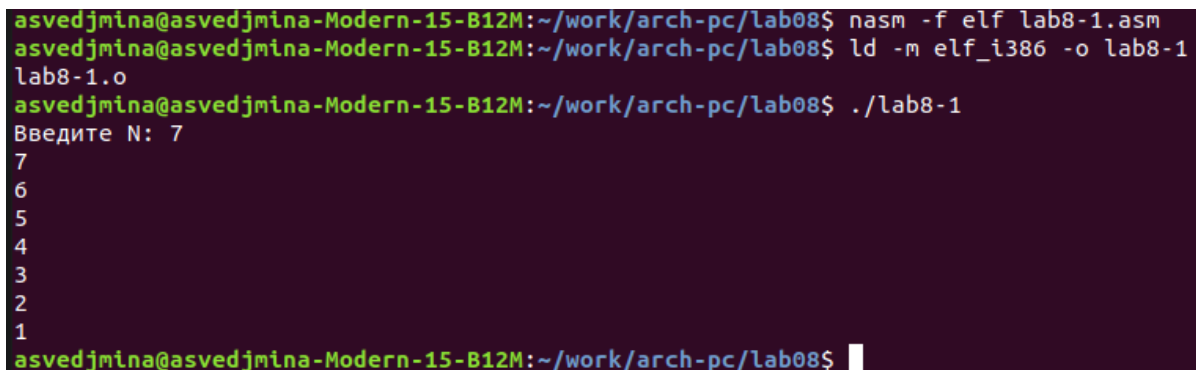


```
GNU nano 4.8 /home/asvedjmina/work/arch-pc/lab08/lab8-1.asm Modified
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace  ^U Paste Text^T To Spell  ^_ Go To Line
```

Рис. 4.2: Ввод программы в lab8-1.asm

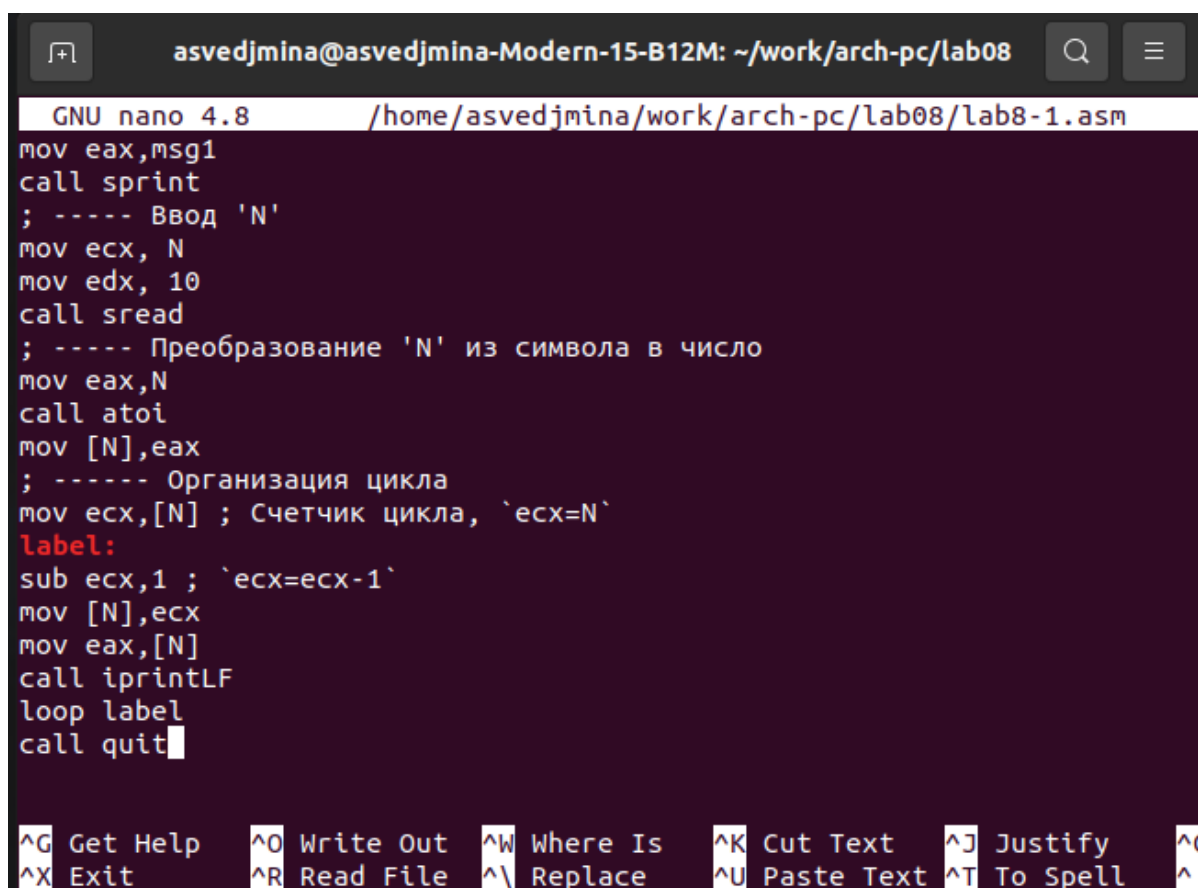
Создаю исполняемый файл и проверяю его работу.



```
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1
lab8-1.o
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 7
7
6
5
4
3
2
1
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$
```

Рис. 4.3: Запуск lab8-1

Изменяю текст программы в данном файле, добавляя именование значения регистра ecx в цикле.

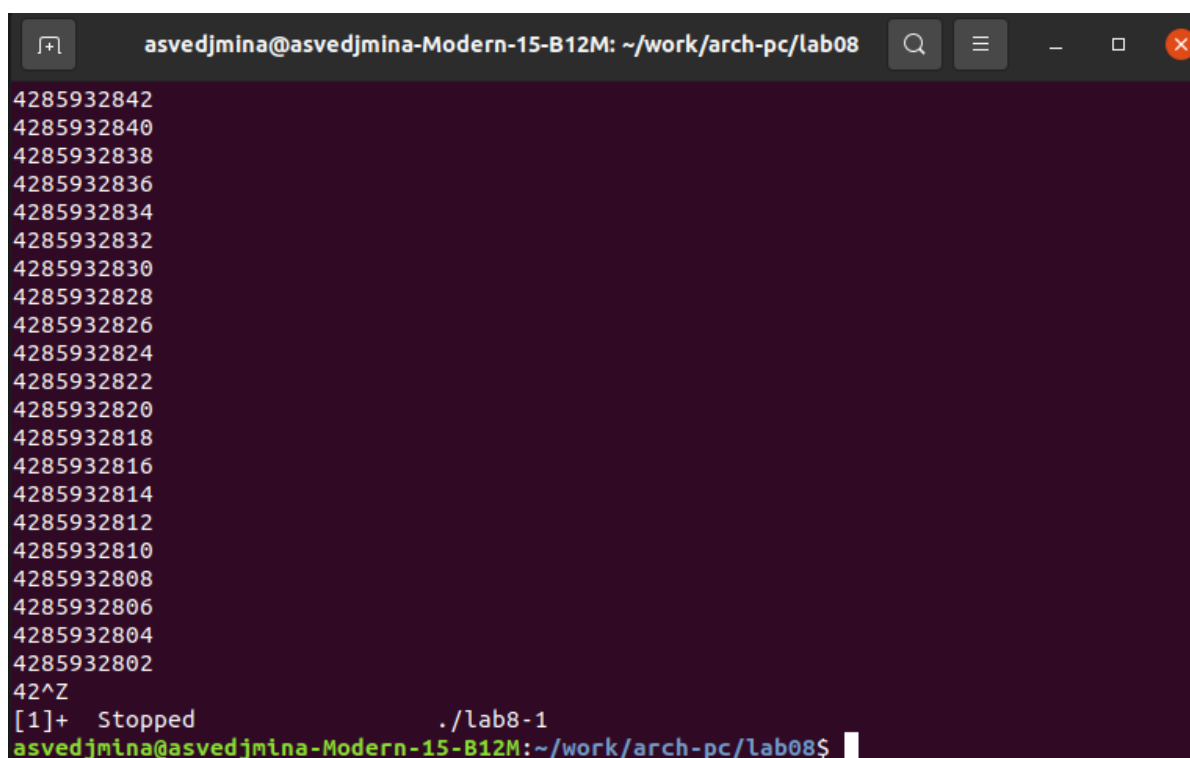


```
GNU nano 4.8 /home/asvedjmina/work/arch-pc/lab08/lab8-1.asm
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
call quit
```

^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C  
^X Exit   ^R Read File   ^\ Replace   ^U Paste Text   ^T To Spell   ^\_

Рис. 4.4: Изменение программы в lab8-1.asm

Создаю исполняемый файл и запускаю его.



```
asvedjmina@asvedjmina-Modern-15-B12M: ~/work/arch-pc/lab08
4285932842
4285932840
4285932838
4285932836
4285932834
4285932832
4285932830
4285932828
4285932826
4285932824
4285932822
4285932820
4285932818
4285932816
4285932814
4285932812
4285932810
4285932808
4285932806
4285932804
4285932802
42^Z
[1]+  Stopped                  ./lab8-1
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$
```

Рис. 4.5: Запуск изменённого lab8-1

Цикл получился бесконечным. Вновь вношу изменения в lab8-1.asm, добавляя команды push и pop.

```
asvedjmina@asvedjmina-Modern-15-B12M: ~/work/arch-pc/lab08
GNU nano 4.8 /home/asvedjmina/work/arch-pc/lab08/lab8-1.asm
; ----- 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:
push ecx ; добавление значения ecx в стек
sub ecx, 1
mov [N], ecx
mov eax, [N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop label
call quit
```

Рис. 4.6: Повторное изменение программы в lab8-1.asm

Создаю исполняемый файл и запускаю его. Теперь число проходов соответствует значению N, введенному с клавиатуры.

```
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1
lab8-1.o
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 7
6
5
4
3
2
1
0
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$
```

Рис. 4.7: Запуск вновь изменённого lab8-1

Создаю файл lab8-2.asm и ввожу в него текст программы, выводящей на экран аргументы командной строки.

```
GNU nano 4.8 /home/asvedjmina/work/arch-pc/lab08/lab8-2.asm Modified
%include 'in_out.asm'
SECTION .text
global _start
_start:
por ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
por 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`)
por 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
```

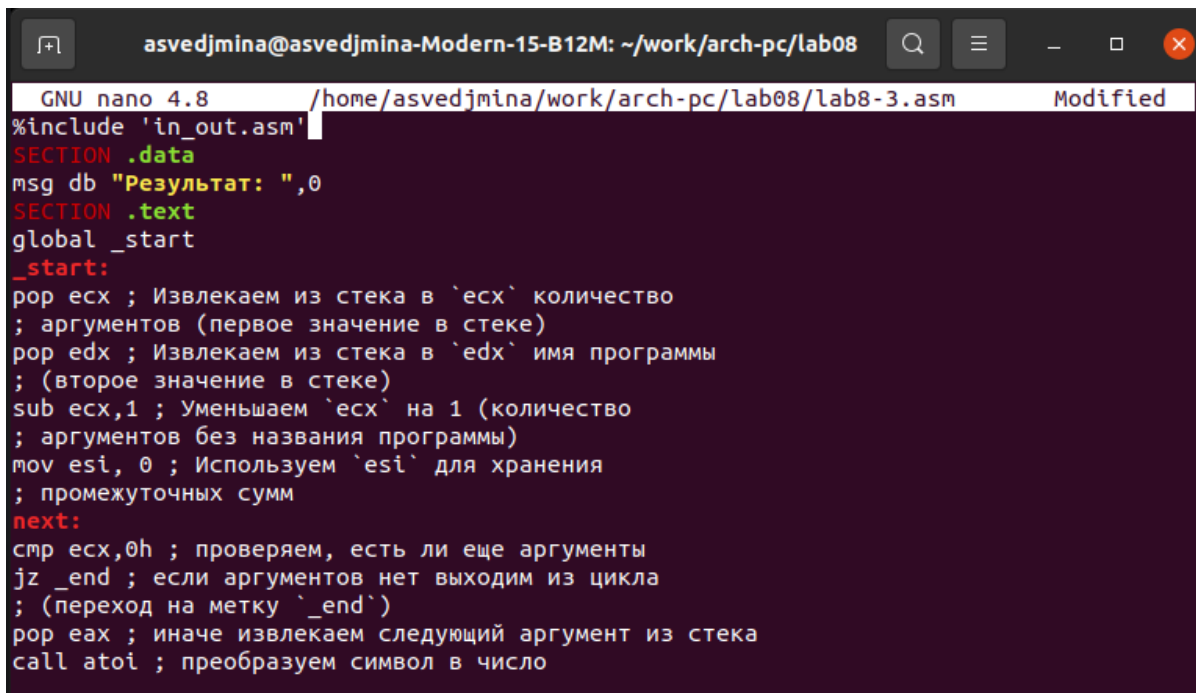
Рис. 4.8: Ввод программы в lab8-2.asm

Создаю исполняемый файл и запускаю его, указав “аргумент1 аргумент 2 ‘аргумент 3’”. Таким образом, программой будет обработано три аргумента.

```
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ touch lab8-2.asm
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ mc
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2
lab8-2.o
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ ./lab8-2
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$ ./lab8-2 3 5 'roar'
3
5
roar
asvedjmina@asvedjmina-Modern-15-B12M:~/work/arch-pc/lab08$
```

Рис. 4.9: Запуск lab8-2

Далее создаю файл lab8-3.asm и ввожу текст программы, вычисляющую сумму аргументов командной строки.



```
GNU nano 4.8 /home/asvedjmina/work/arch-pc/lab08/lab8-3.asm Modified
#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
```

Рис. 4.10: Ввод программы в lab8-3.asm

Создаю исполняемый файл и запускаю его.

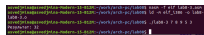
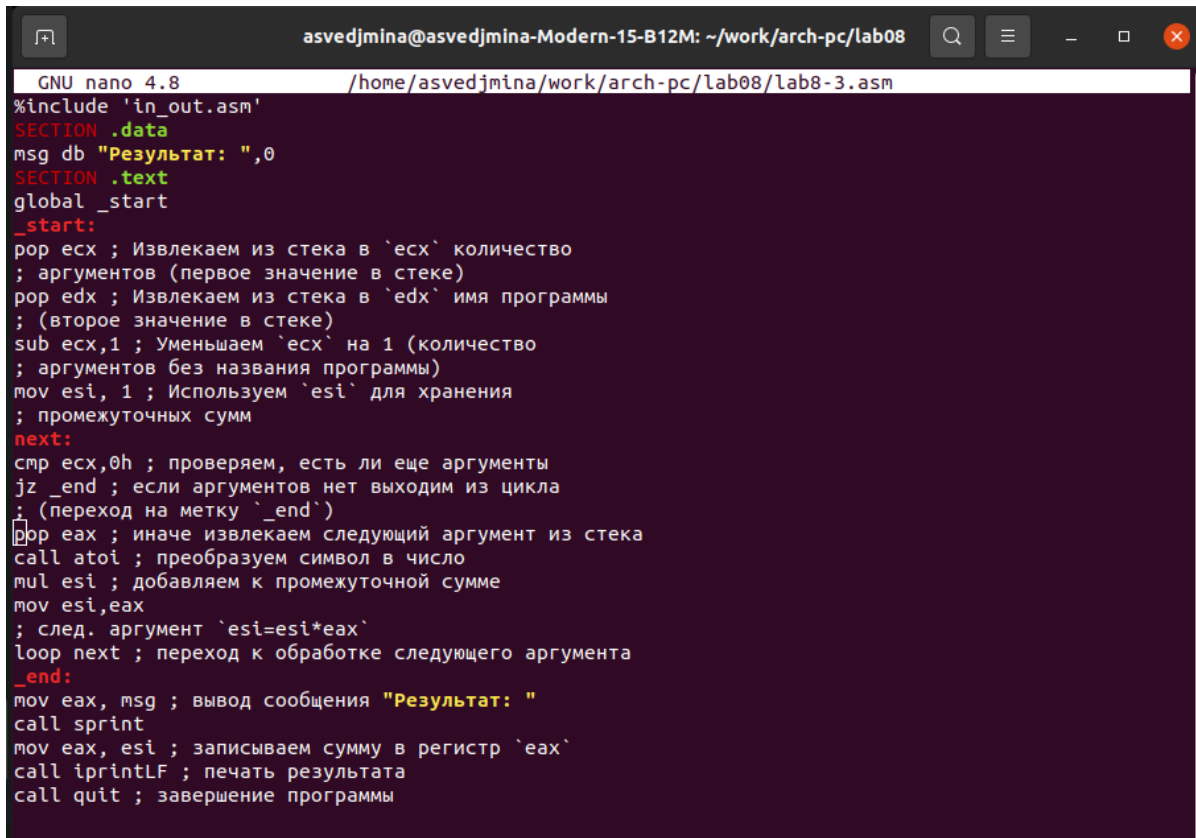


Рис. 4.11: Запуск lab8-3

Затем изменяю текст программы так, чтобы она вычисляла произведение аргументов командной строки.



```
GNU nano 4.8 /home/asvedjmina/work/arch-pc/lab08/lab8-3.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, 1 ; 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
mul esi ; добавляем к промежуточной сумме
mov esi,eax
; след. аргумент `esi=esi*eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем сумму в регистр `eax`
call iprintf ; печать результата
call quit ; завершение программы
```

Рис. 4.12: Изменение программы в lab8-3.asm

Создаю исполняемый файл и запускаю его.



Рис. 4.13: Запуск изменённого lab8-3

## 5 Выполнение заданий для самостоятельной работы

Для выполнения заданий для самостоятельной работы создаю файл sumrub.asm.



Рис. 5.1: Создание файла sumrub.asm

Ввожу в него программу, которая будет выводить сумму значений функции. Номер моего варианта - 4, поэтому я буду реализовывать  $2^*(x-1)$ .

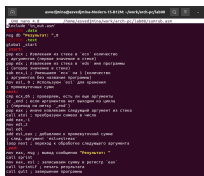


Рис. 5.2: Ввод программы в sumrub.asm

Создаю исполняемый файл и запускаю его.



Рис. 5.3: Запуск sumrub



## 6 Выводы

В ходе лабораторной работы я приобрела навыки написания программ, в которых используются циклы и обработка аргументов командной строки.