

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8

Студент: Барханоева Раяна Магометовна

Ст.билет: 1032252468

Группа: НКАбд-01-25

МОСКВА

2025 г

Содержание

1. Цель работы.....	3
2. Выполнение работы.....	4
3. Самостоятельная работа.....	10
Вывод.....	12

1. Цель работы

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

2. Выполнение работы

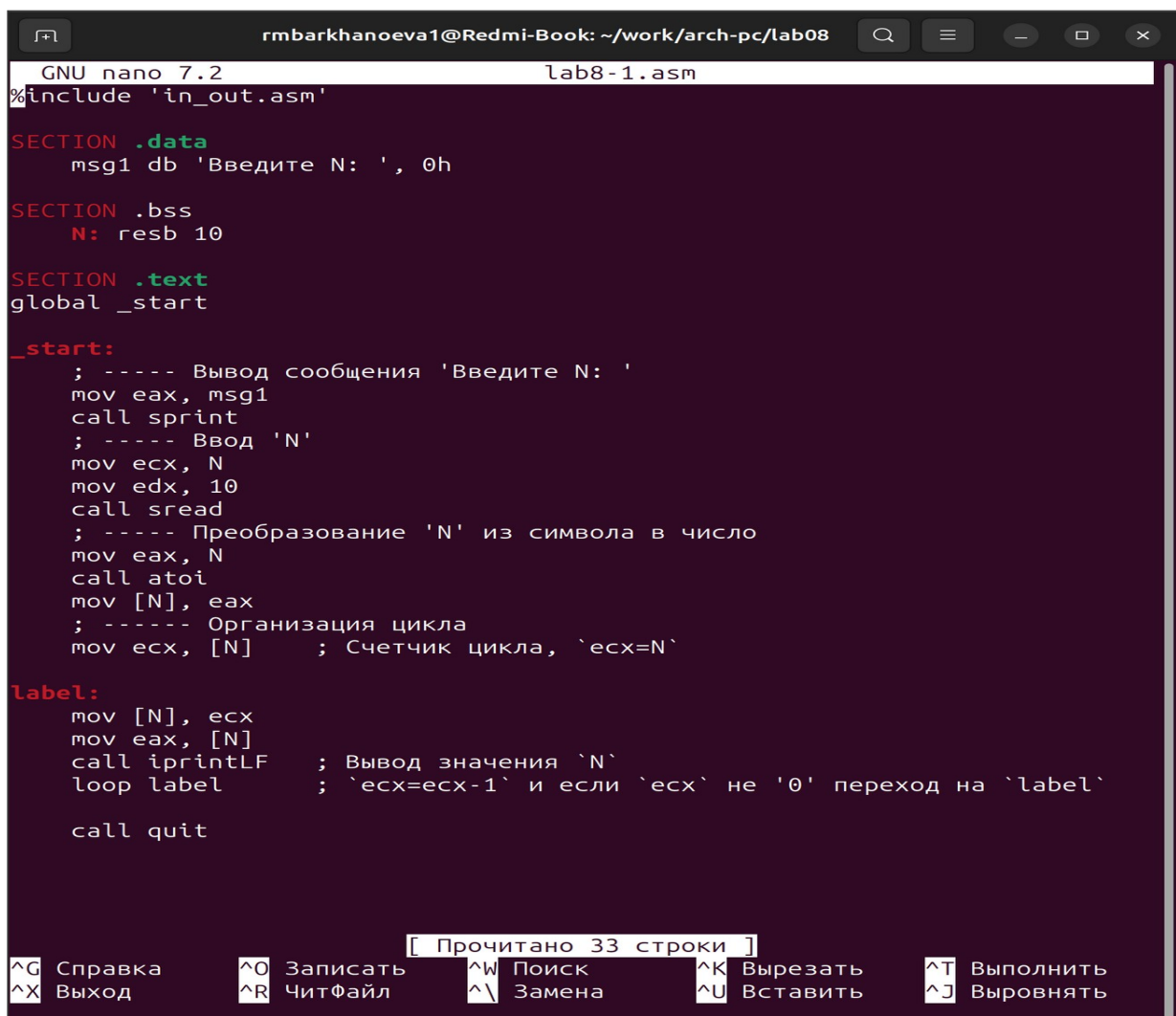
Создание каталога (рисунок 1).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc$ mkdir lab08
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc$ cd lab08/
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ touch lab8-1.asm
```

Рисунок 1.

Создание каталога и файла lab8-1.asm

Программа (рисунок 2).



```
GNU nano 7.2 lab8-1.asm
%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
    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' переход на `label`

    call quit
```

[Прочитано 33 строки]

^G Справка	^O Записать	^W Поиск	^K Вырезать	^T Выполнить
^X Выход	^R ЧитФайл	^N Замена	^U Вставить	^J Выводить

Рисунок 2.

Программа вывода значений регистра ecx

Тест программы (рисунок 3).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$
```

Рисунок 3.

Данный пример показывает, что использование регистра ecx в теле цикла loop может привести к некорректной работе программы.

Программа (рисунок 4).

```
GNU nano 7.2 lab8-1.asm *
%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
    mov ecx, [N] ; Счетчик цикла, 'ecx=N'

label:
    sub ecx, 1
    mov [N], ecx
    mov eax, [N]
    call iprintLF ; Вывод значения 'N'
    loop label ; 'ecx=ecx-1' и если 'ecx' не '0' переход на 'label'
    call quit

^G Справка      ^O Записать
^X Выход        ^R ЧитФайл
^W Поиск       ^N Вырезать
^_ Замена      ^U Вставить
^T Выполнить   ^J Выровнять
```

Рисунок 4.

Добавили изменение значения регистра ecx в цикле

Тест программы (рисунок 5.)

```
rmbarkhanoeva1@Redmi-Book: ~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
rmbarkhanoeva1@Redmi-Book: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
rmbarkhanoeva1@Redmi-Book: ~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
rmbarkhanoeva1@Redmi-Book: ~/work/arch-pc/lab08$
```

Рисунок 5.

Создали исполняемый файл и проверили на работоспособность. Не соответствует количеству проходов.

Программа (рисунок 6).

```
GNU nano 7.2 lab8-1.asm *
%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
mov ecx, [N] ; Счетчик цикла, 'ecx=N'

label:
push ecx
sub ecx, 1
mov [N], ecx
mov eax, [N]
call iprintlnLF ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0' переход на 'label'
pop ecx
call quit

Имя файла для записи: lab8-1.asm
^G Справка M-D Формат DOS M-A Доп. в начало M-B Резерв. копия
^C Отмена M-M Формат Mac M-P Доп. в конец ^T Обзор
```

Рисунок 6.

Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Внесли изменения в текст программы добавив команды `push` и `pop`.

Тест программы (рисунок 7).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$
```

Рисунок 7.

Не соответствует число проходов.

Программа (рисунок 8).

```
GNU nano 7.2 lab8-2.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 sprintLF    ; вызываем функцию печати
    loop next       ; переход к обработке следующего
                  ; аргумента (переход на метку `next`)
_end:
    call quit
```

Рисунок 8.

Программа выводящая на экран аргументы командной строки

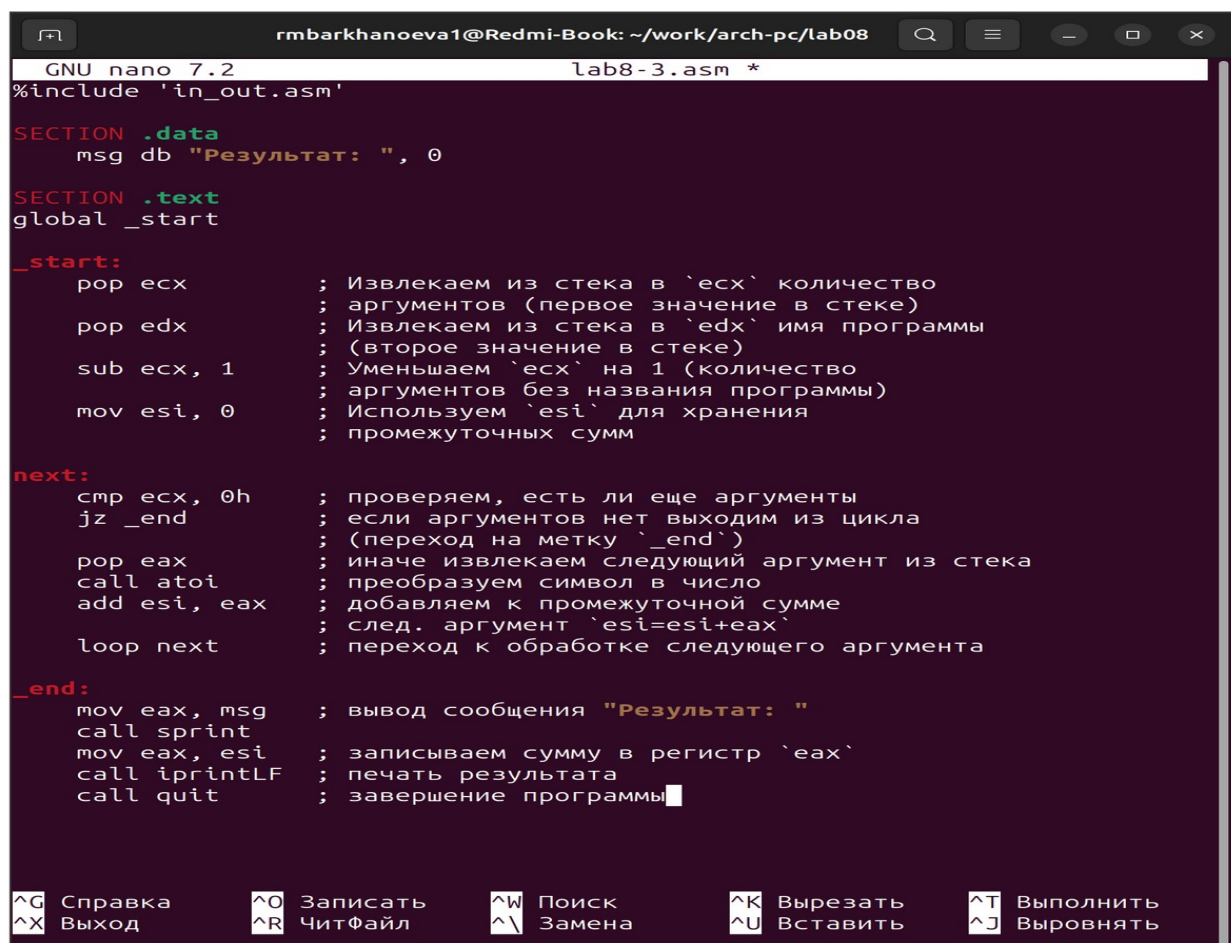
Тест программы (рисунок 9).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ./lab8-2 10 20 30
10
20
30
```

Рисунок 9.

Скомпелировал файл lab8-2.asm и выводом стало три аргумента, которые я ввел.

Программа (рисунок 10).



```
GNU nano 7.2 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, 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
    mov eax, esi     ; записываем сумму в регистр `eax`
    call iprintLF    ; печать результата
    call quit        ; завершение программы
```

GNU nano 7.2 interface with keyboard shortcuts at the bottom:

^G Справка	^O Записать	^W Поиск	^K Вырезать	^T Выполнить
^X Выход	^R ЧитФайл	^N Замена	^U Вставить	^J Выводить

Рисунок 10.

Программа, которая выводит сумму чисел, которые передаются в программу как аргументы.

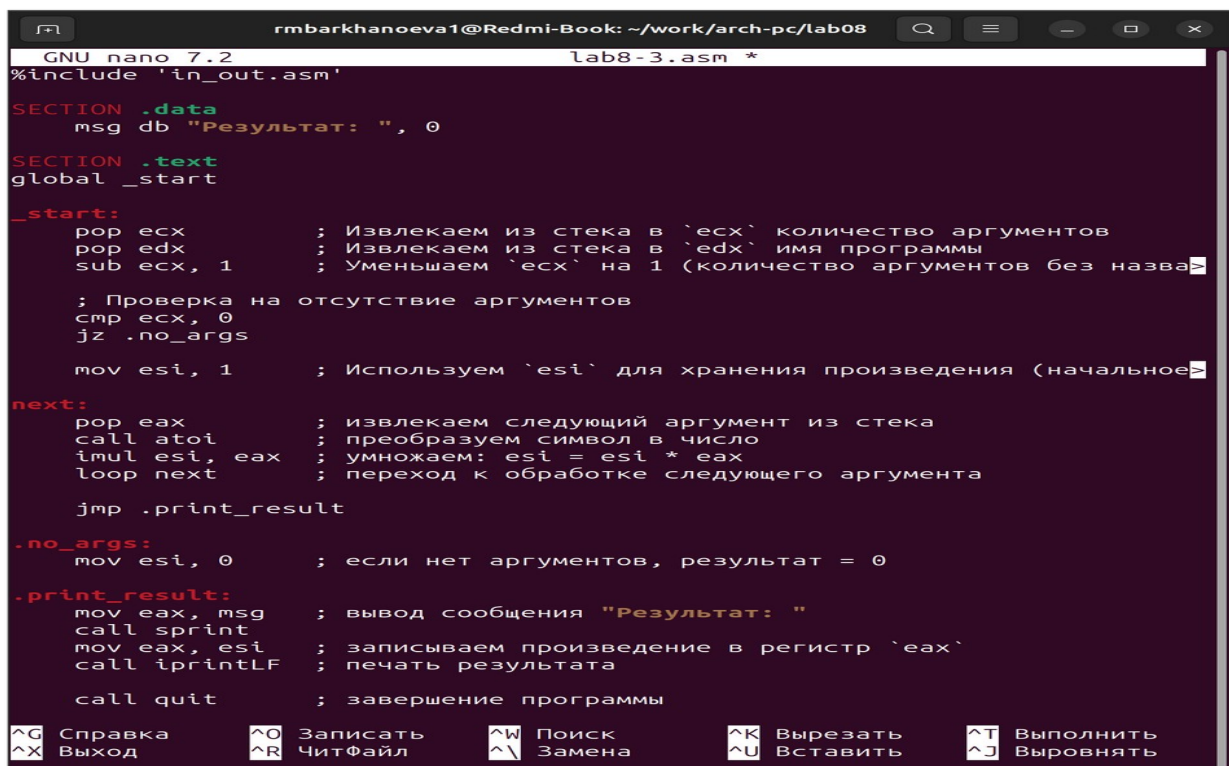
Тест программы (рисунок 11).

```
rmbarkhanoeva1@Redmi-Book: ~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
rmbarkhanoeva1@Redmi-Book: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
rmbarkhanoeva1@Redmi-Book: ~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
rmbarkhanoeva1@Redmi-Book: ~/work/arch-pc/lab08$
```

Рисунок 11.

Скомпилировали программу, ввели аргументы и получили сумму.

Программа (рисунок 12).



```
GNU nano 7.2 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
    pop edx          ; Извлекаем из стека в `edx` имя программы
    sub ecx, 1       ; Уменьшаем `ecx` на 1 (количество аргументов без названия)

    ; Проверка на отсутствие аргументов
    cmp ecx, 0
    jz .no_args

    mov esi, 1       ; Используем `esi` для хранения произведения (начальное значение)

next:
    pop eax          ; извлекаем следующий аргумент из стека
    call atoi        ; преобразуем символ в число
    imul esi, eax     ; умножаем: esi = esi * eax
    loop next        ; переход к обработке следующего аргумента

    jmp .print_result

.no_args:
    mov esi, 0       ; если нет аргументов, результат = 0

.print_result:
    mov eax, msg     ; вывод сообщения "Результат: "
    call sprint
    mov eax, esi      ; записываем произведение в регистр `eax`
    call iprintLF    ; печать результата

    call quit        ; завершение программы

^G Справка      ^O Записать
^X Выход        ^R ЧитФайл
^_              ^W Поиск
               ^\ Замена
               ^K Вырезать
               ^U Вставить
               ^T Выполнить
               ^J Выводить
```

Рисунок 12.

Программа которая перемножает аргументы.

Тест программы (рисунок 13).

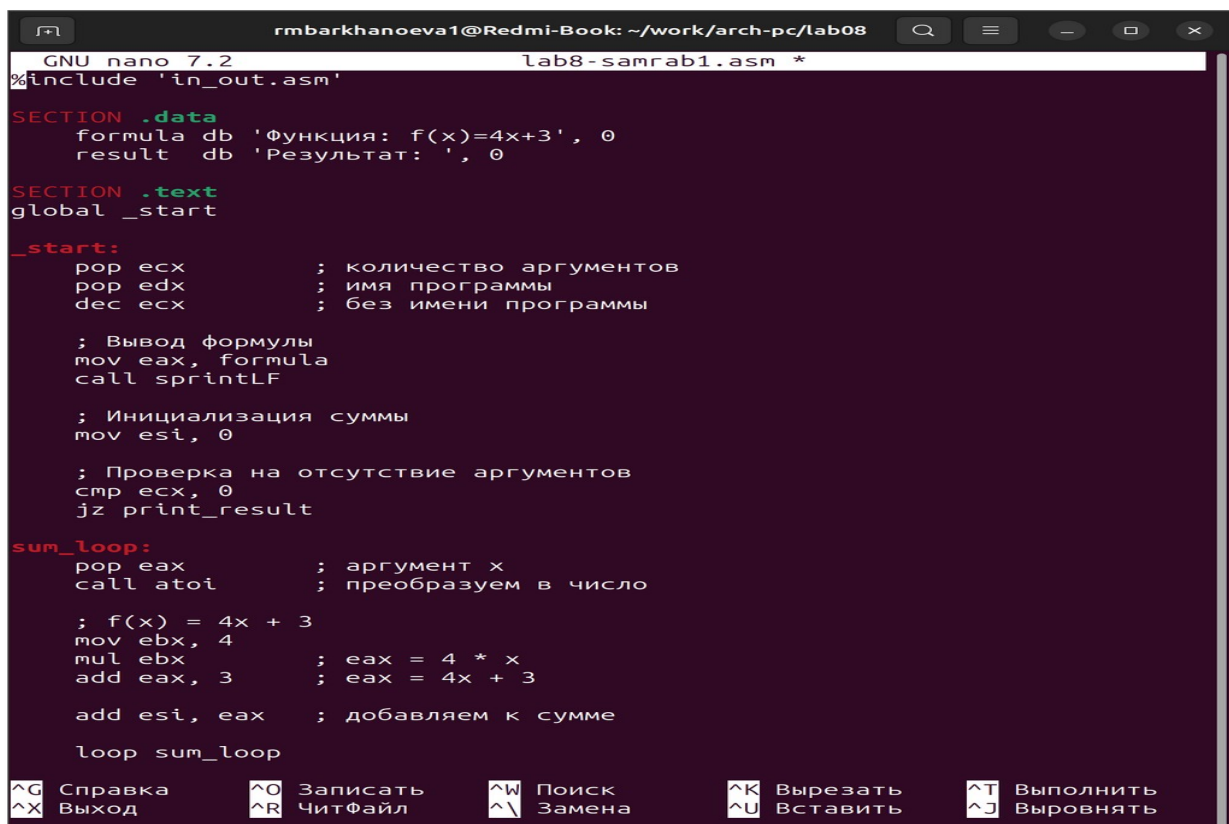
```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 la
b8-3.o
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ./lab8-3 5 5 3
Результат: 75
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$
```

Рисунок 13.

Скомпелировали файл, ввели аргументы и программа корректно их перемножила.

3. Самостоятельная работа.

Программа (рисунок 14).



```
GNU nano 7.2 lab8-samrab1.asm *
%include 'in_out.asm'

SECTION .data
    formula db 'Функция: f(x)=4x+3', 0
    result db 'Результат: ', 0

SECTION .text
global _start

_start:
    pop ecx          ; количество аргументов
    pop edx          ; имя программы
    dec ecx          ; без имени программы

    ; Вывод формулы
    mov eax, formula
    call printf

    ; Инициализация суммы
    mov esi, 0

    ; Проверка на отсутствие аргументов
    cmp ecx, 0
    jz print_result

sum_loop:
    pop eax          ; аргумент x
    call atoi        ; преобразуем в число

    ; f(x) = 4x + 3
    mov ebx, 4
    mul ebx          ; eax = 4 * x
    add eax, 3       ; eax = 4x + 3

    add esi, eax     ; добавляем к сумме

    loop sum_loop

; print_result:
;     mov eax, result
;     call printf
;     exit
```

Рисунок 14.

Программа, 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.

Тест программы (рисунок 15).

```
гmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ nasm -f elf lab8-samrab1.asm
гmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-samrab1 lab8-samrab1.o
гmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ./lab8-samrab1
Функция:  $f(x)=4x+3$ 
Результат: 0
гmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ./lab8-samrab1 1 2 3 4
Функция:  $f(x)=4x+3$ 
Результат: 52
гmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ./lab8-samrab1 2 2 2 2
Функция:  $f(x)=4x+3$ 
Результат: 44
гmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab08$ ./lab8-samrab1 4 3 2 1
Функция:  $f(x)=4x+3$ 
Результат: 52
```

Рисунок 15.

Скомпелировали файл и ввели аргументы.

Вывод

В ходе лабораторной работы №8 были изучены принципы программирования циклов в NASM и обработки аргументов командной строки. Освоена организация стека, его назначение и операции `push` и `pop`, а также роль регистра `ecx` как счётчика цикла при использовании инструкции `loop`. На практике рассмотрены ошибки, возникающие при изменении регистра `ecx` внутри цикла, и способы их устранения с помощью стека. Также были получены навыки извлечения аргументов командной строки из стека, их последовательной обработки в цикле и выполнения над ними арифметических операций. В результате выполненных заданий сформировано понимание работы стека, циклов и механизмов передачи данных в программах на языке ассемблера NASM.