

Отчёт по лабораторной работе 8

дисциплина: Архитектура компьютера

Кылыш Гоктюрк

Содержание

1 Цель работы	5
2 Выполнение лабораторной работы	6
3 Выводы	19

Список иллюстраций

2.1 Программа lab8-1.asm	7
2.2 Запуск программы lab8-1.asm	8
2.3 Программа lab8-1.asm	9
2.4 Запуск программы lab8-1.asm	10
2.5 Программа lab8-1.asm	11
2.6 Запуск программы lab8-1.asm	12
2.7 Программа lab8-2.asm	13
2.8 Запуск программы lab8-2.asm	13
2.9 Программа lab8-3.asm	14
2.10 Запуск программы lab8-3.asm	15
2.11 Программа lab8-3.asm	16
2.12 Запуск программы lab8-3.asm	16
2.13 Программа lab8-4.asm	17
2.14 Запуск программы lab8-4.asm	18

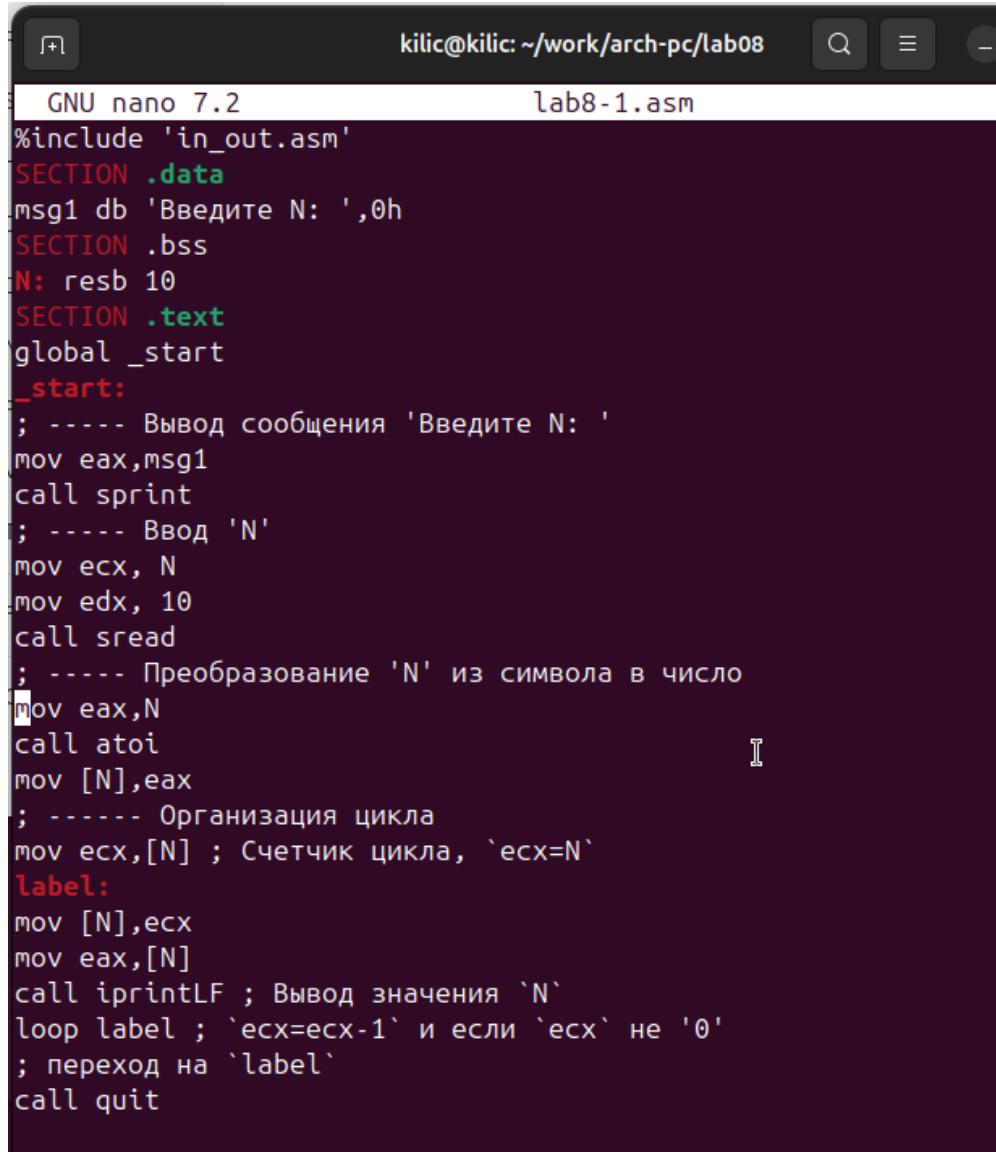
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1. Создал каталог для программам лабораторной работы № 8, перешел в него и создал файл lab8-1.asm
2. Написал в файл lab8-1.asm текст программы из листинга 8.1. Создал исполняемый файл и проверил его работу.



The screenshot shows a terminal window titled "lab8-1.asm" running on a Linux system. The code is written in assembly language using the AT&T syntax. It includes sections for data and text, defines a global variable N, and implements a loop to read an integer from the user and print it. The assembly code is as follows:

```
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
```

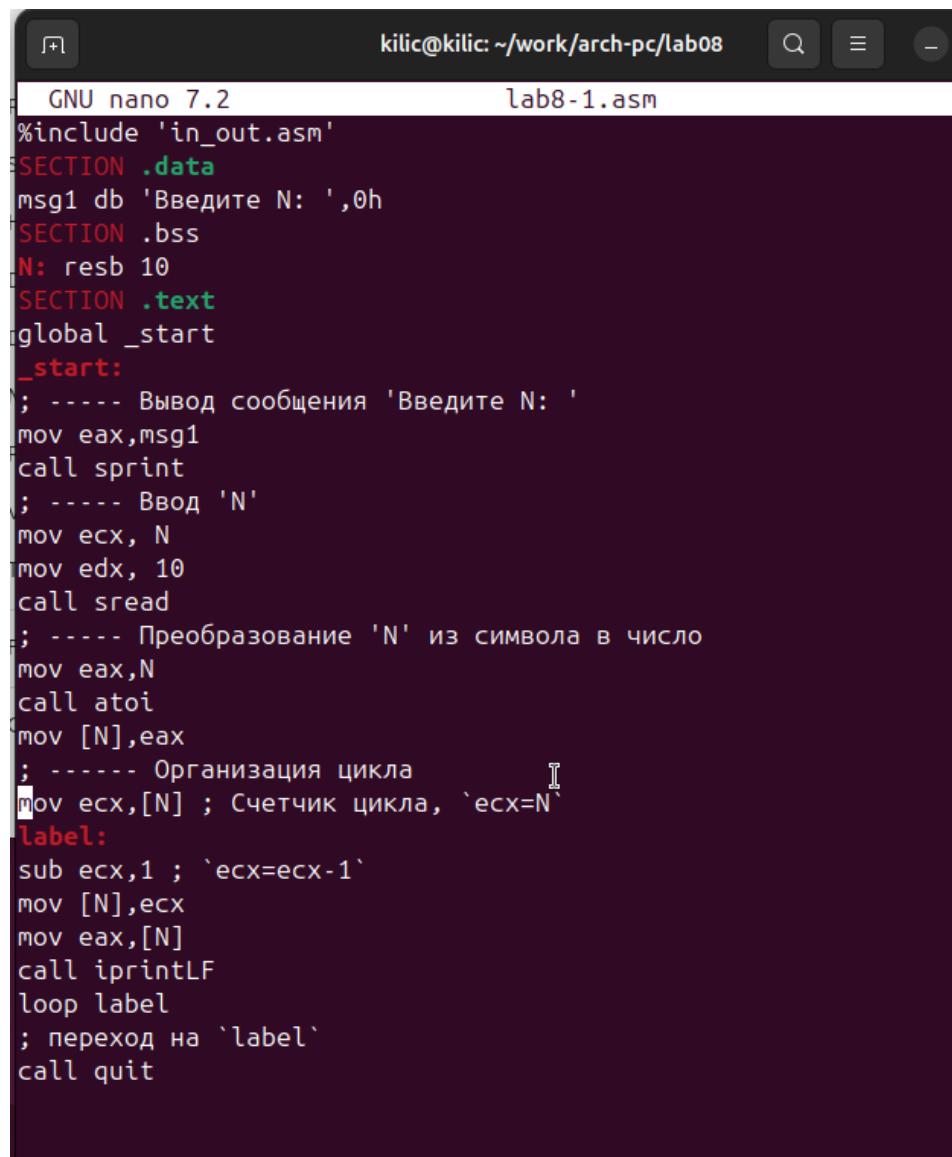
Рисунок 2.1: Программа lab8-1.asm

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

Рисунок 2.2: Запуск программы lab8-1.asm

3. Данный пример показывает, что использование регистра есх в теле цикла loop может привести к некорректной работе программы. Изменил текст программы добавив изменение значение регистра есх в цикле: Создайте исполняемый файл и проверьте его работу. Какие значения принимает регистр есх в цикле? Соответствует ли число проходов цикла значению N, введенному с клавиатуры?

Программа запускает бесконечный цикл при нечетном N и выводит только нечетные числа при четном N.



The screenshot shows a terminal window titled "GNU nano 7.2" with the command "kilic@kilic:~/work/arch-pc/lab08" at the top. The window displays assembly code for a program named "lab8-1.asm". The code includes sections for data and text, defines a variable N, and implements a loop to print numbers from 1 to N. The assembly instructions are color-coded: blue for labels, red for section headers, green for include files, and black for comments and data.

```
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 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintf
loop label
; переход на `label`
call quit
```

Рисунок 2.3: Программа lab8-1.asm

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

Рисунок 2.4: Запуск программы lab8-1.asm

4. Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внеси изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop. Создал исполняемый файл и проверьте его работу. Соответствует ли в данном случае число проходов цикла значению N введенному с клавиатуры?

Программа выводит числа от N-1 до 0, число проходов цикла соответствует N.

```
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 ; 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
```

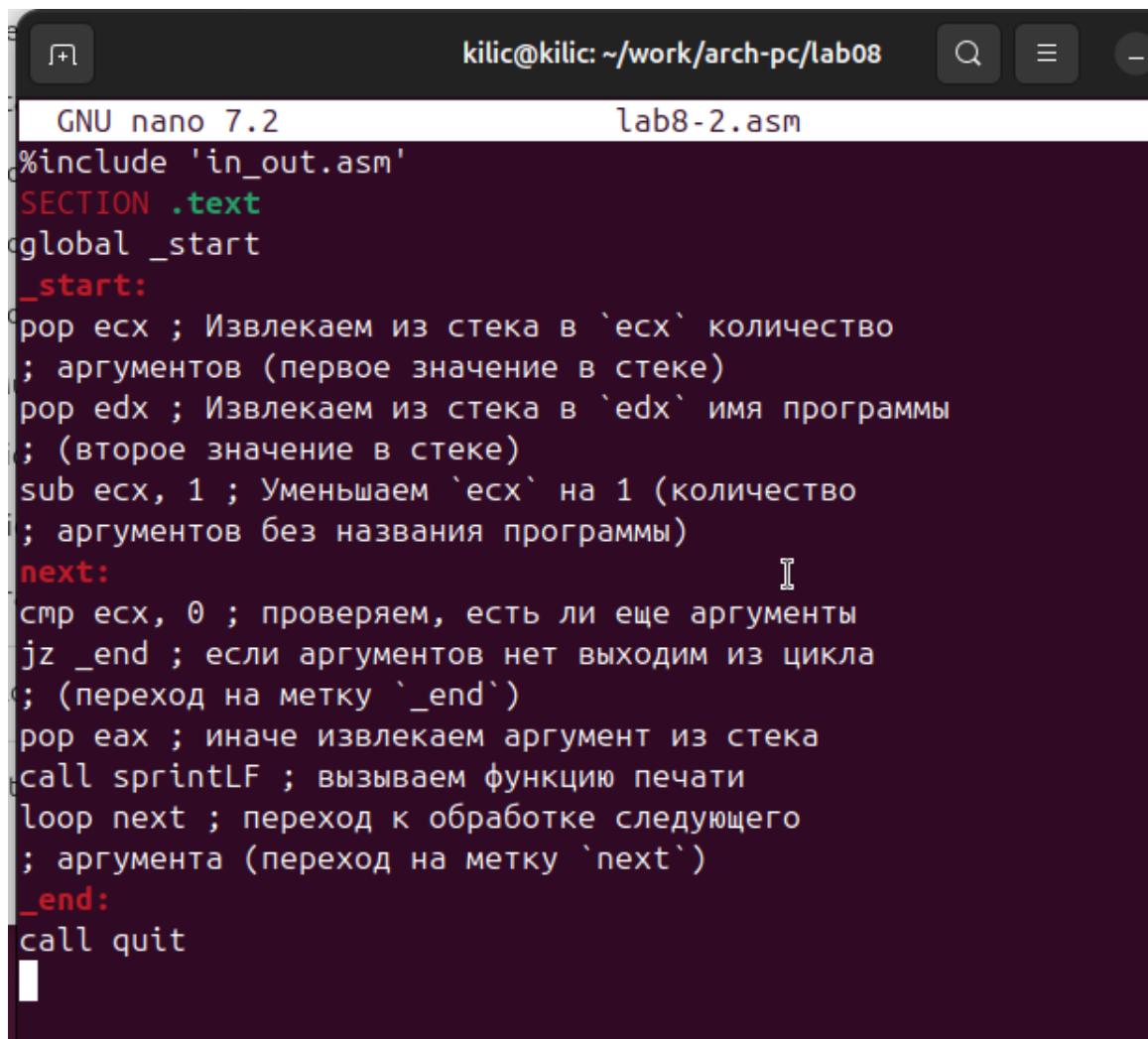
Рисунок 2.5: Программа lab8-1.asm

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

Рисунок 2.6: Запуск программы lab8-1.asm

5. Создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввел в него текст программы из листинга 8.2. Создал исполняемый файл и запустил его, указав аргументы. Сколько аргументов было обработано программой?

Программа обработала 4 аргумента.

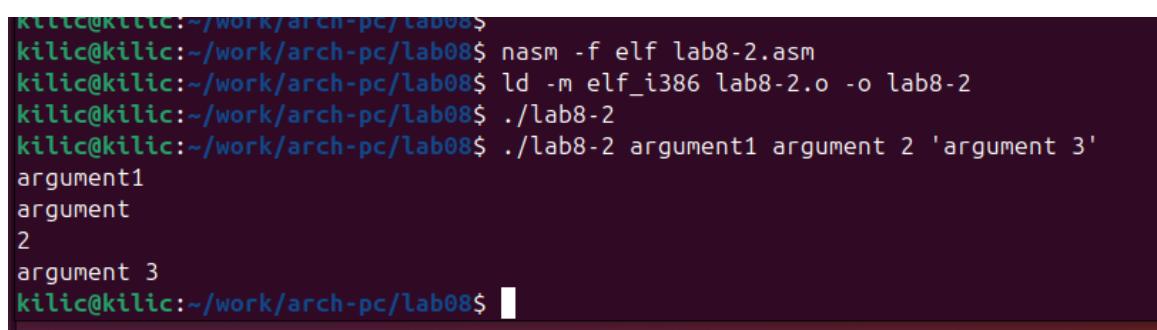


GNU nano 7.2 lab8-2.asm

```
%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 2.7: Программа lab8-2.asm

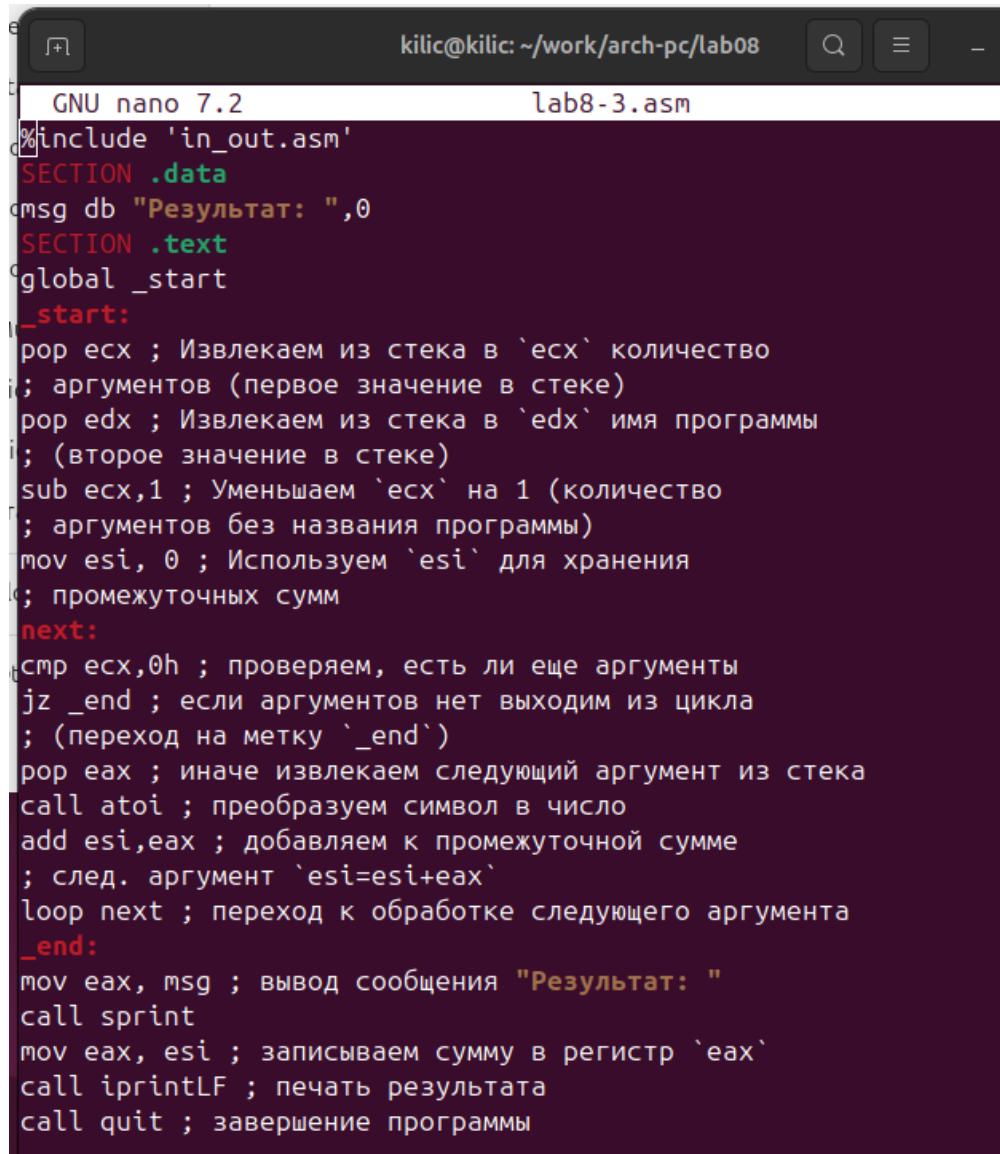


```
kilic@kilic:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
kilic@kilic:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
kilic@kilic:~/work/arch-pc/lab08$ ./lab8-2
kilic@kilic:~/work/arch-pc/lab08$ ./lab8-2 argument1 argument 2 'argument 3'
argument1
argument
2
argument 3
kilic@kilic:~/work/arch-pc/lab08$
```

Рисунок 2.8: Запуск программы lab8-2.asm

6. Рассмотрим еще один пример программы которая выводит сумму чисел,

которые передаются в программу как аргументы.



The screenshot shows a terminal window titled 'GNU nano 7.2' with the file path 'kilic@kilic: ~/work/arch-pc/lab08'. The current file is 'lab8-3.asm'. The code is written in assembly language and includes comments in Russian explaining the purpose of each instruction. The code reads arguments from the stack, initializes a sum in ESI, loops through arguments to add them to ESI, prints the result, and exits.

```
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
```

Рисунок 2.9: Программа lab8-3.asm

```
kilic@kilic:~/work/arch-pc/lab08$  
kilic@kilic:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm  
kilic@kilic:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3  
kilic@kilic:~/work/arch-pc/lab08$ ./lab8-3  
Результат: 0  
kilic@kilic:~/work/arch-pc/lab08$ ./lab8-3 3 6 9  
Результат: 18  
kilic@kilic:~/work/arch-pc/lab08$ █
```

Рисунок 2.10: Запуск программы lab8-3.asm

7. Изменл текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.

GNU nano 7.2 lab8-3.asm

```
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
    mov ebx,eax
    mov eax,esi
    mul ebx
    mov 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
    █
```

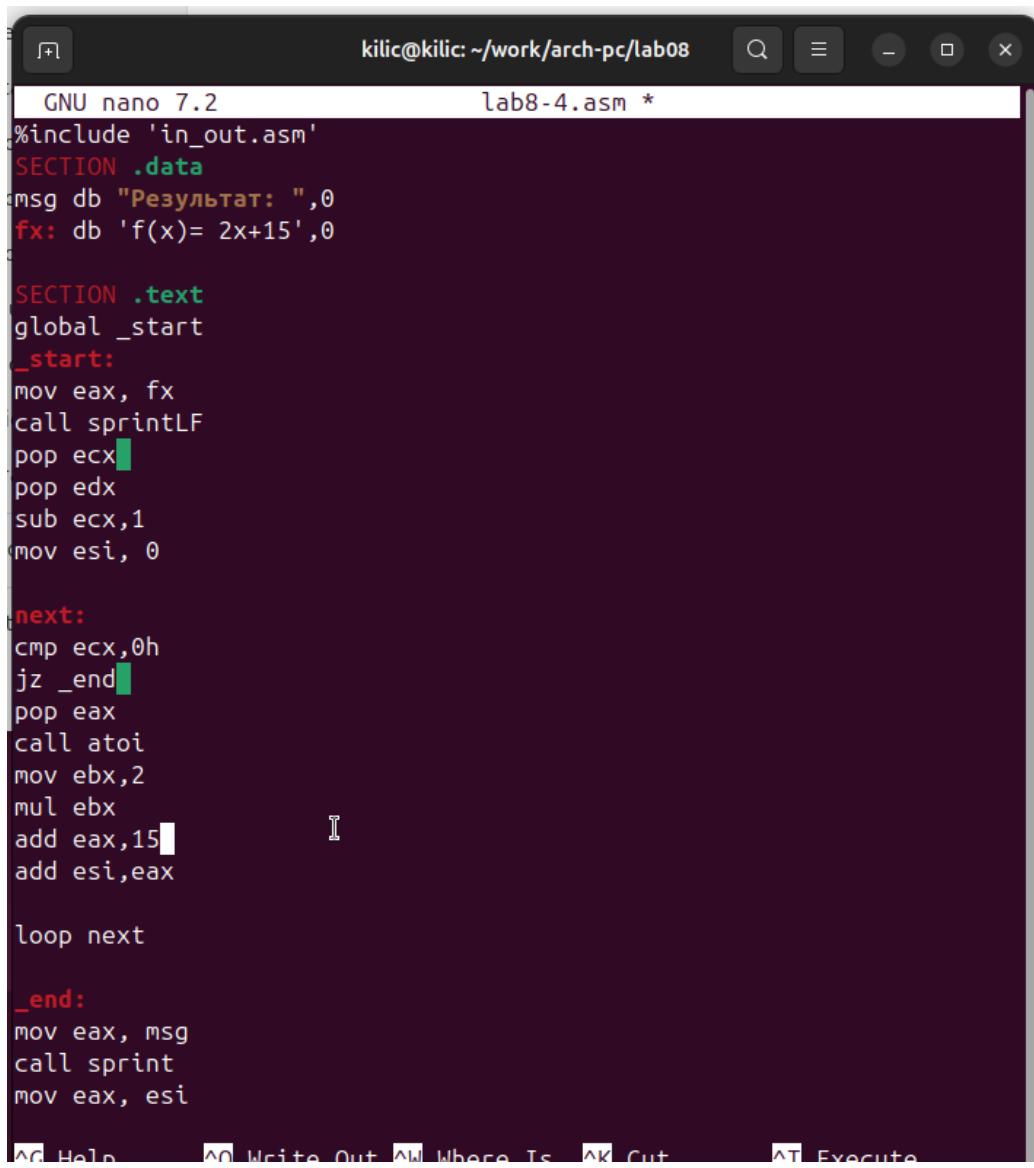
Рисунок 2.11: Программа lab8-3.asm

```
kilic@kilic:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kilic@kilic:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
kilic@kilic:~/work/arch-pc/lab08$ ./lab8-3
Результат: 1
kilic@kilic:~/work/arch-pc/lab08$ ./lab8-3 3 6 9
Результат: 162
kilic@kilic:~/work/arch-pc/lab08$
```

Рисунок 2.12: Запуск программы lab8-3.asm

8. Напишите 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 передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x .

для варианта 1 $f(x) = 2x + 15$



The screenshot shows a terminal window titled "GNU nano 7.2" with the command "kilic@kilic: ~/work/arch-pc/lab08" at the top. The window contains assembly code for a program named "lab8-4.asm". The code defines a function $f(x) = 2x + 15$ and calculates the sum of values for x from 0 to $n-1$. The assembly instructions include `msg`, `fx`, `SECTION .text`, `global _start`, `_start:`, `mov eax, fx`, `call sprintLF`, `pop ecx`, `pop edx`, `sub ecx,1`, `mov esi, 0`, `next:`, `cmp ecx,0h`, `jz _end`, `pop eax`, `call atoi`, `mov ebx,2`, `mul ebx`, `add eax,15`, `add esi,eax`, `loop next`, and `_end:`. The file ends with `mov eax, msg`, `call sprint`, and `mov eax, esi`. At the bottom of the terminal window, there are several keyboard shortcut icons.

```
GNU nano 7.2          lab8-4.asm *
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 2x+15',0

SECTION .text
global _start
_start:
    mov eax, fx
    call sprintLF
    pop ecx
    pop edx
    sub ecx,1
    mov esi, 0

    next:
    cmp ecx,0h
    jz _end
    pop eax
    call atoi
    mov ebx,2
    mul ebx
    add eax,15
    add esi,eax

    loop next

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

Рисунок 2.13: Программа lab8-4.asm

```
kilic@kilic:~/work/arch-pc/lab08$  
kilic@kilic:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm  
kilic@kilic:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-4.o -o lab8-4  
kilic@kilic:~/work/arch-pc/lab08$ ./lab8-4  
f(x)= 2x+15  
Результат: 0  
kilic@kilic:~/work/arch-pc/lab08$ ./lab8-4 3  
f(x)= 2x+15  
Результат: 21  
kilic@kilic:~/work/arch-pc/lab08$ ./lab8-4 3 6 7 9 1 2 3  
f(x)= 2x+15  
Результат: 167  
kilic@kilic:~/work/arch-pc/lab08$
```

Рисунок 2.14: Запуск программы lab8-4.asm

3 Выводы

Освоили работы со стеком, циклом и аргументами на ассемблере nasm.