

# **Отчет по лабораторной работе 9**

**Программирование цикла. Обработка аргументов командной строки.**

Гисматуллин Артём Вадимович НПИбд-01-22

# 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>  |
| 3.1      | Организация стека . . . . .                          | 7         |
| 3.1.1    | Добавление элемента в стек. . . . .                  | 7         |
| 3.1.2    | Извлечение элемента из стека. . . . .                | 8         |
| 3.2      | Инструкции организации циклов . . . . .              | 8         |
| <b>4</b> | <b>Выполнение лабораторной работы</b>                | <b>9</b>  |
| <b>5</b> | <b>Выполнение задания для самостоятельной работы</b> | <b>17</b> |
| <b>6</b> | <b>Выводы</b>  | <b>20</b> |
|          | <b>Список литературы</b>                             | <b>21</b> |

## Список иллюстраций

|     |   |    |
|-----|---|----|
| 4.1 | Командная строка. каталог ~/work/arch-pc/lab09 . . . . .      | 9  |
| 4.2 | Командная строка. Выполнение программы lab9-1.asm . . . . .   | 10 |
| 4.3 | Командная строка. Выполнение измененной программы lab9-1.asm  | 11 |
| 4.4 | Командная строка. Измененная программа lab9-1.asm . . . . .   | 12 |
| 4.5 | Командная строка. Измененная программа lab9-1.asm 2 . . . . . | 13 |
| 4.6 | Командная строка. Выполнение программы lab9-2.asm . . . . .   | 14 |
| 4.7 | Командная строка. Выполнение программы lab9-3.asm . . . . .   | 15 |
| 4.8 | Командная строка. Выполнение измененной программы lab9-3.asm  | 16 |
| 5.1 | Командная строка. Программа lab9-4.asm . . . . .              | 18 |
| 5.2 | Командная строка. Выполнение программы lab9-4.asm . . . . .   | 19 |

## Список таблиц

# 1 Цель работы

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

## 2 Задание

1. Реализовать цикл с помощью листинга 9.1. Выполнить соответствующее задание с использованием регистра `esx` в теле цикла
2. Изменить цикл так, чтоб использовать внутри него регистр `esx` вкуче со стеком.
3. Обработать аргументы командной строки на основе листинга 9.2.
4. Реализовать программу по вычислению суммы поступающих аргументов. Переписать программу так, чтобы она вычисляла на их основе произведение.
5. Написать программу, которая находит сумму значений функции (на основе варианта 11).

## 3 Теоретическое введение

### 3.1 Организация стека

Стек — это структура данных, организованная по принципу LIFO. Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды.

Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается.

Для стека существует две основные операции:

- добавление элемента в вершину стека (push);
- извлечение элемента из вершины стека (pop).

#### 3.1.1 Добавление элемента в стек.

Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Данная команда имеет один операнд — значение, которое необходимо поместить в стек.

Существует ещё две команды для добавления значений в стек. Это команда `pusha`, которая помещает в стек содержимое всех регистров общего назначения в следующем порядке: `ax`, `cx`, `dx`, `bx`, `sp`, `bp`, `si`, `di`. А также команда `pushf`, которая служит для перемещения в стек содержимого регистра флагов. Обе эти команды не имеют операндов.

### **3.1.2 Извлечение элемента из стека.**

Команда `pop` извлекает значение из стека, т.е. извлекает значение из ячейки памяти, на которую указывает регистр `esp`, после этого уменьшает значение регистра `esp` на 4. У этой команды также один операнд, который может быть регистром или переменной в памяти.

Нужно помнить, что извлечённый из стека элемент не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек.

Аналогично команде записи в стек существует команда `popa`, которая восстанавливает из стека все регистры общего назначения, и команда `popf` для перемещения значений из вершины стека в регистр флагов.

## **3.2 Инструкции организации циклов**

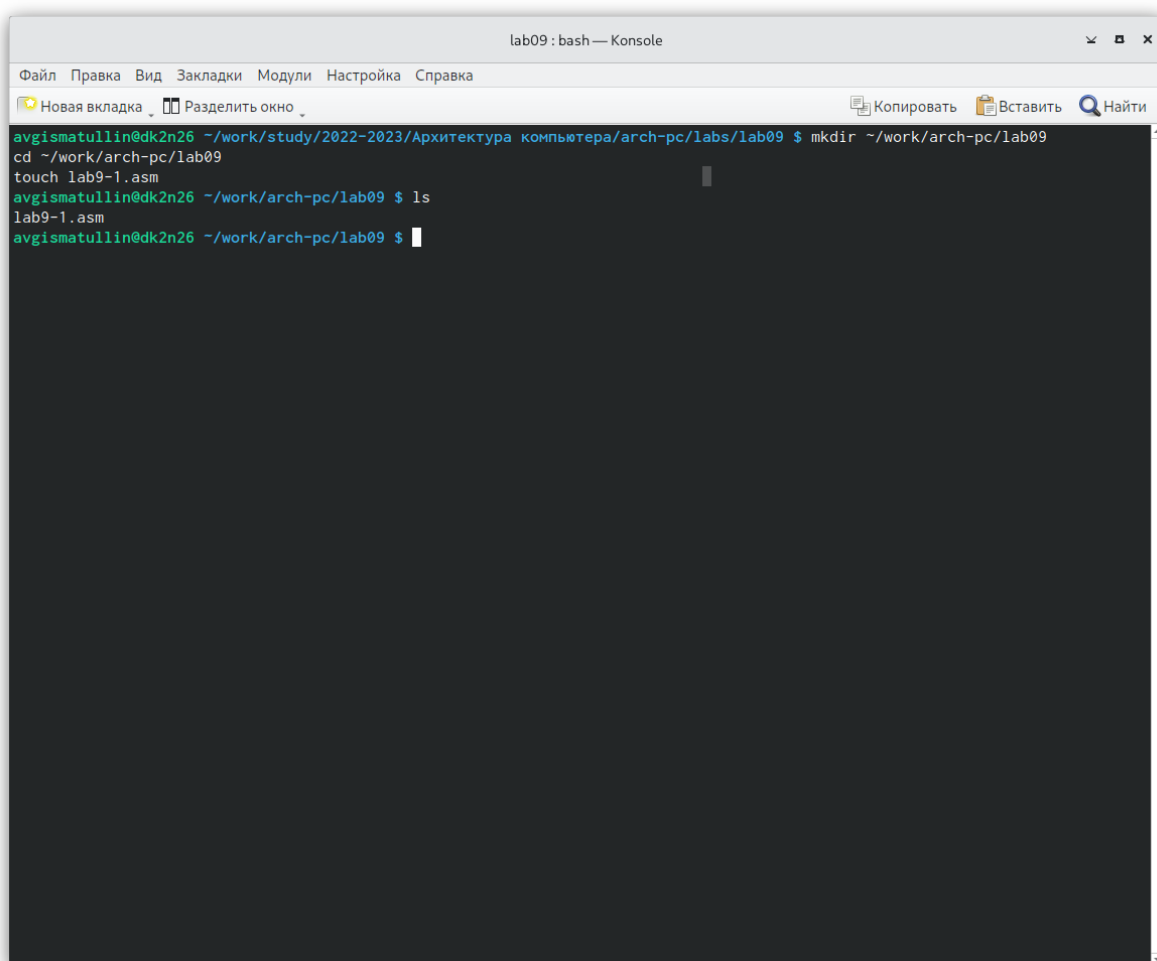
Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре `ecx`. Наиболее простой является инструкция `loop`. Она позволяет организовать безусловный цикл.

Инструкция `loop` выполняется в два этапа. Сначала из регистра `ecx` вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды `loop`.



## 4 Выполнение лабораторной работы

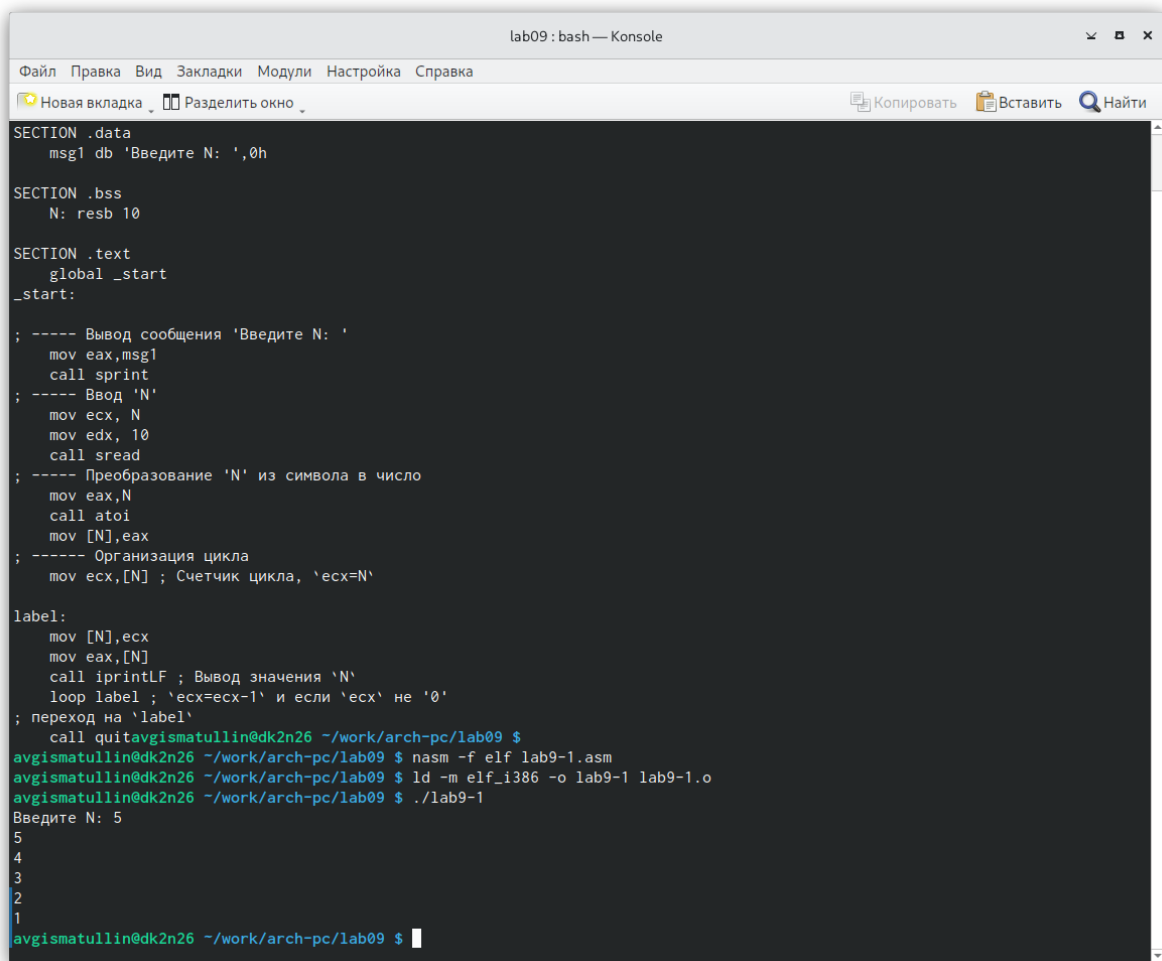
1. Создаем каталог для программ лабораторной работы No 9, перейдем в него и создаем файл lab9-1.asm. (рис. 4.1)



```
lab09: bash — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно Копировать Вставить Найти
avgismatullin@dk2n26 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab09 $ mkdir ~/work/arch-pc/lab09
cd ~/work/arch-pc/lab09
touch lab9-1.asm
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ ls
lab9-1.asm
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $
```

Рис. 4.1: Командная строка. каталог ~/work/arch-pc/lab09

2. Ввели в файл lab9-1.asm текст программы листинга, создали исполняемый файл и посмотрели результат программы. (рис. 4.2)



```
lab09: bash — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно Копировать Вставить Найти
SECTION .data
    msg1 db 'Введите N: ',0h

SECTION .bss
    N: resb 10

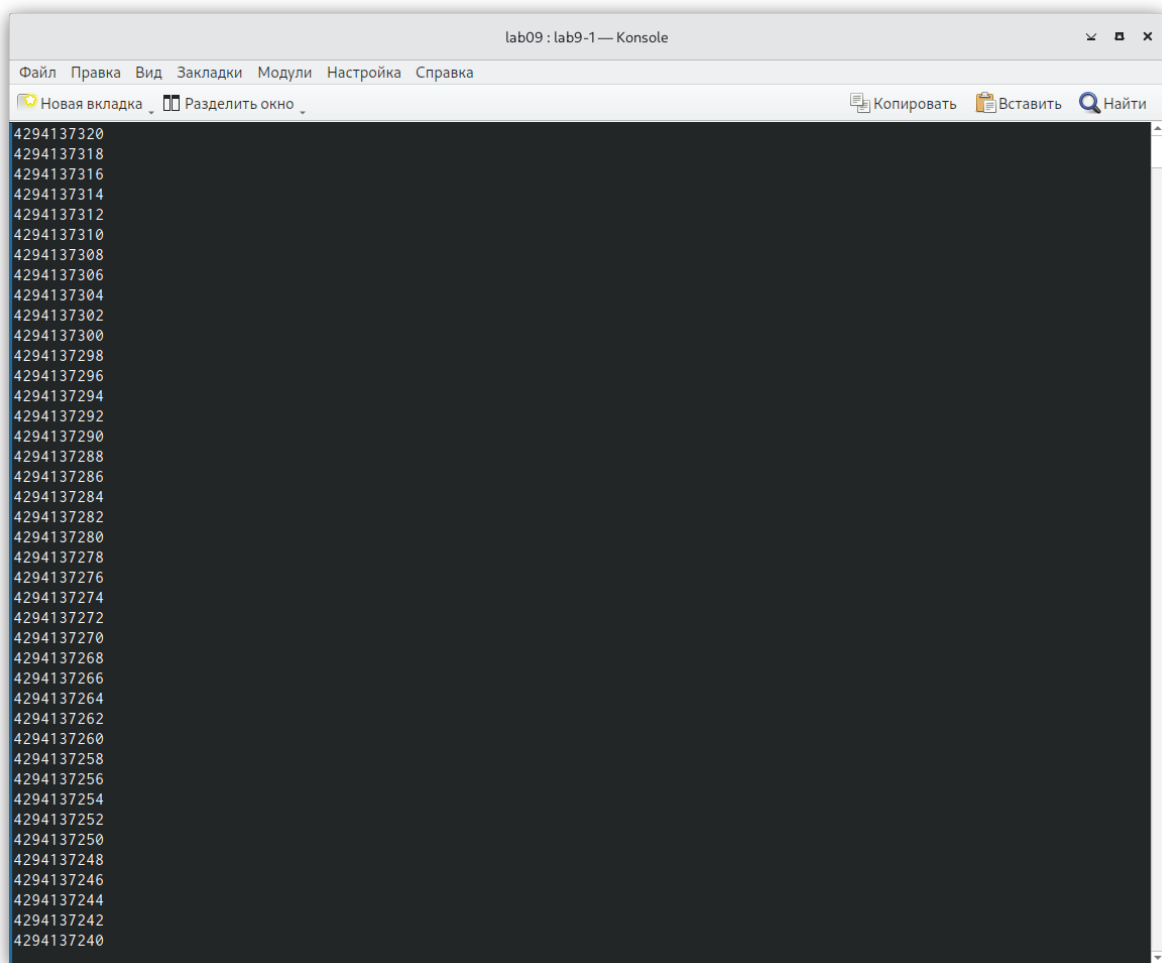
SECTION .text
    global _start
_start:

; ---- 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 quitavgismatullin@dk2n26 ~/work/arch-pc/lab09 $
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ ./lab9-1
Введите N: 5
5
4
3
2
1
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $
```

Рис. 4.2: Командная строка. Выполнение программы lab9-1.asm

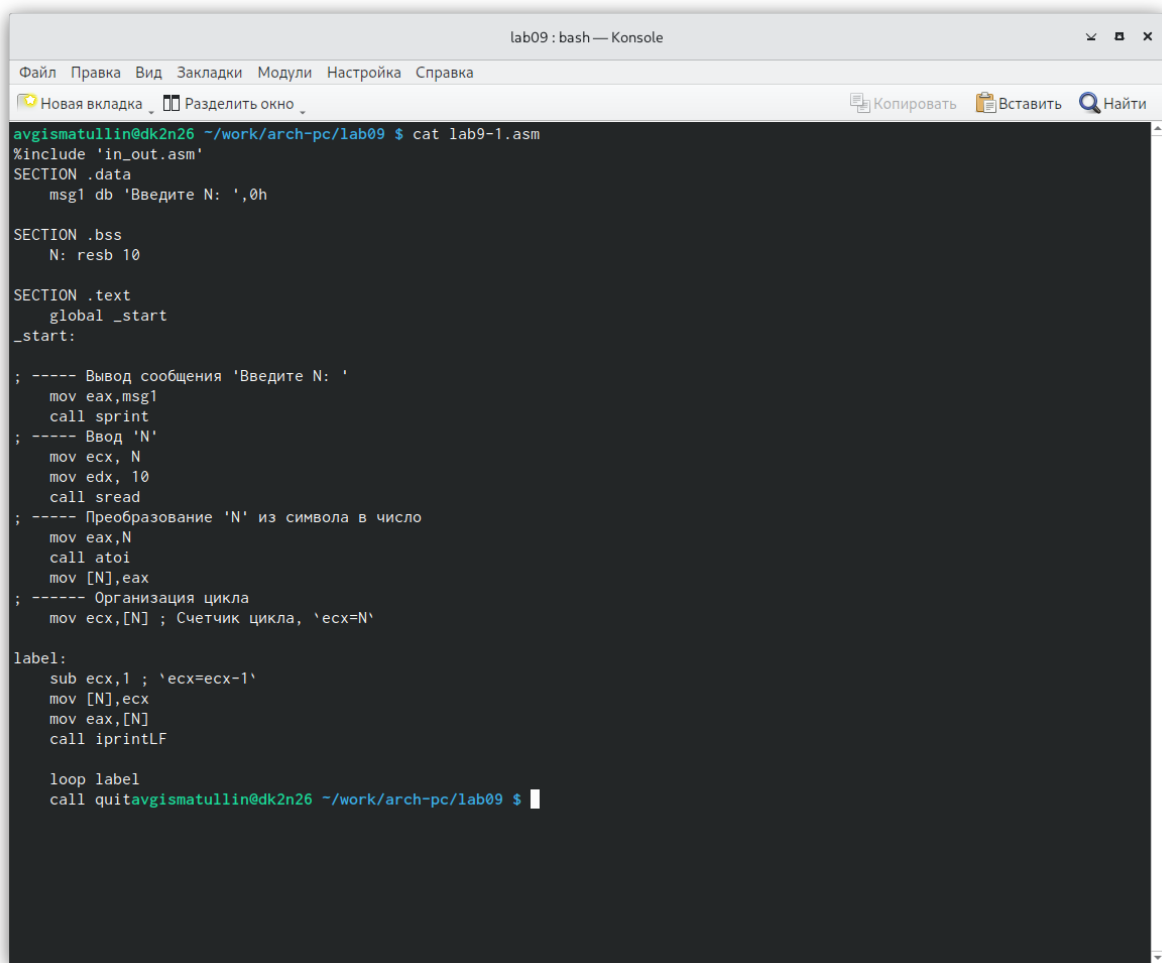
Программа циклично выводит N значений по убыванию. Работает корректно. Далее производим некоторые в ней изменения, начиная взаимодействовать с регистром ecx в самом цикле. Результат: (рис. 4.3)



```
lab09 : lab9-1 — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
Новая вкладка  Разделить окно  Копировать  Вставить  Найти
4294137320
4294137318
4294137316
4294137314
4294137312
4294137310
4294137308
4294137306
4294137304
4294137302
4294137300
4294137298
4294137296
4294137294
4294137292
4294137290
4294137288
4294137286
4294137284
4294137282
4294137280
4294137278
4294137276
4294137274
4294137272
4294137270
4294137268
4294137266
4294137264
4294137262
4294137260
4294137258
4294137256
4294137254
4294137252
4294137250
4294137248
4294137246
4294137244
4294137242
4294137240
```

Рис. 4.3: Командная строка. Выполнение измененной программы lab9-1.asm

При следующем коде: (рис. 4.4)



```
lab09: bash — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно Копировать Вставить Найти
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ cat lab9-1.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'

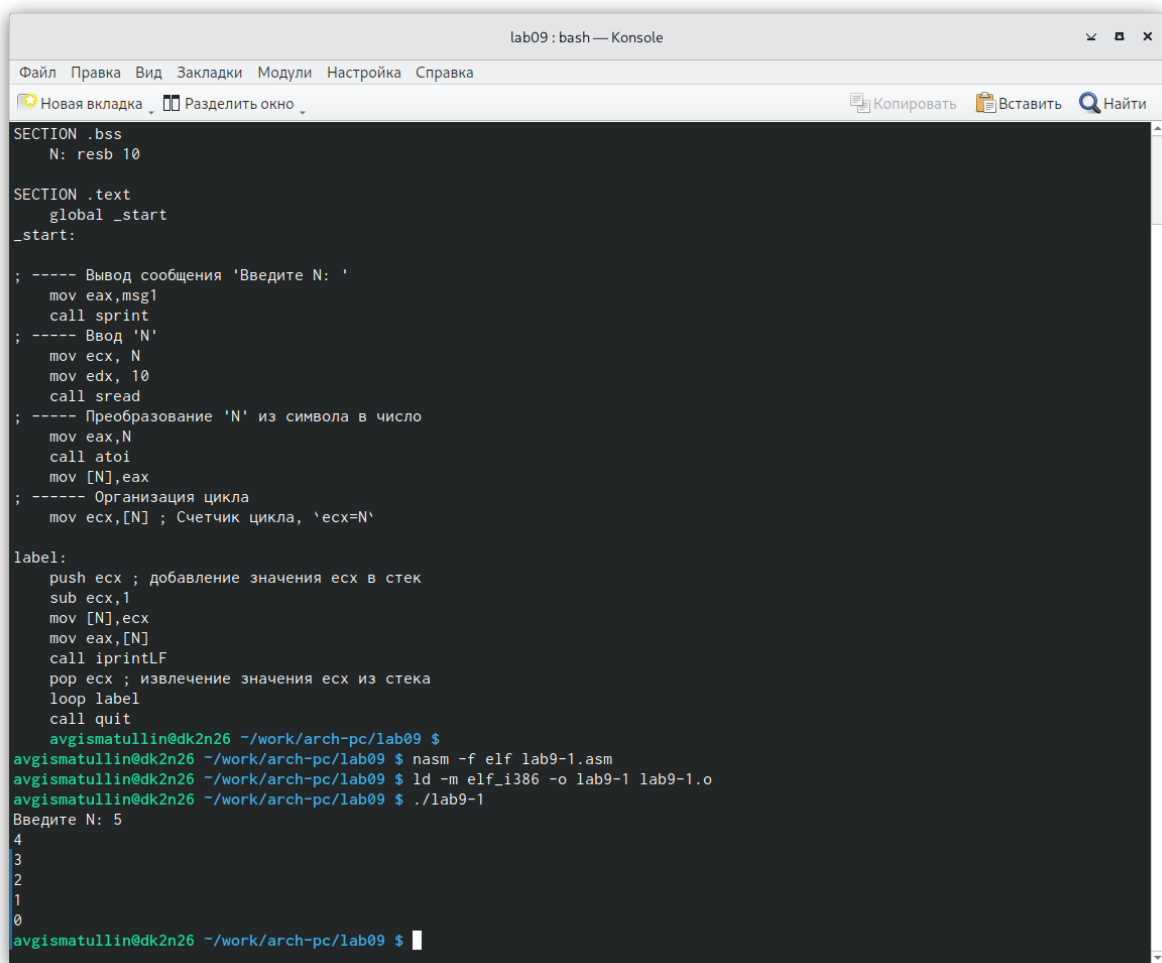
label:
    sub ecx,1 ; 'ecx=ecx-1'
    mov [N],ecx
    mov eax,[N]
    call iprintLF

    loop label
    call quitavgismatullin@dk2n26 ~/work/arch-pc/lab09 $
```

Рис. 4.4: Командная строка. Измененная программа lab9-1.asm

Инструкция loop сама следит за изменениями значений в регистре ecx. Реализованное выше вмешательство приводит к тому, что цикл продолжался бесконечно.

Конечно, для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Реализация этого следующая: (рис. 4.5)



```
lab09: bash — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно
Копировать Вставить Найти

SECTION .bss
N: resb 10

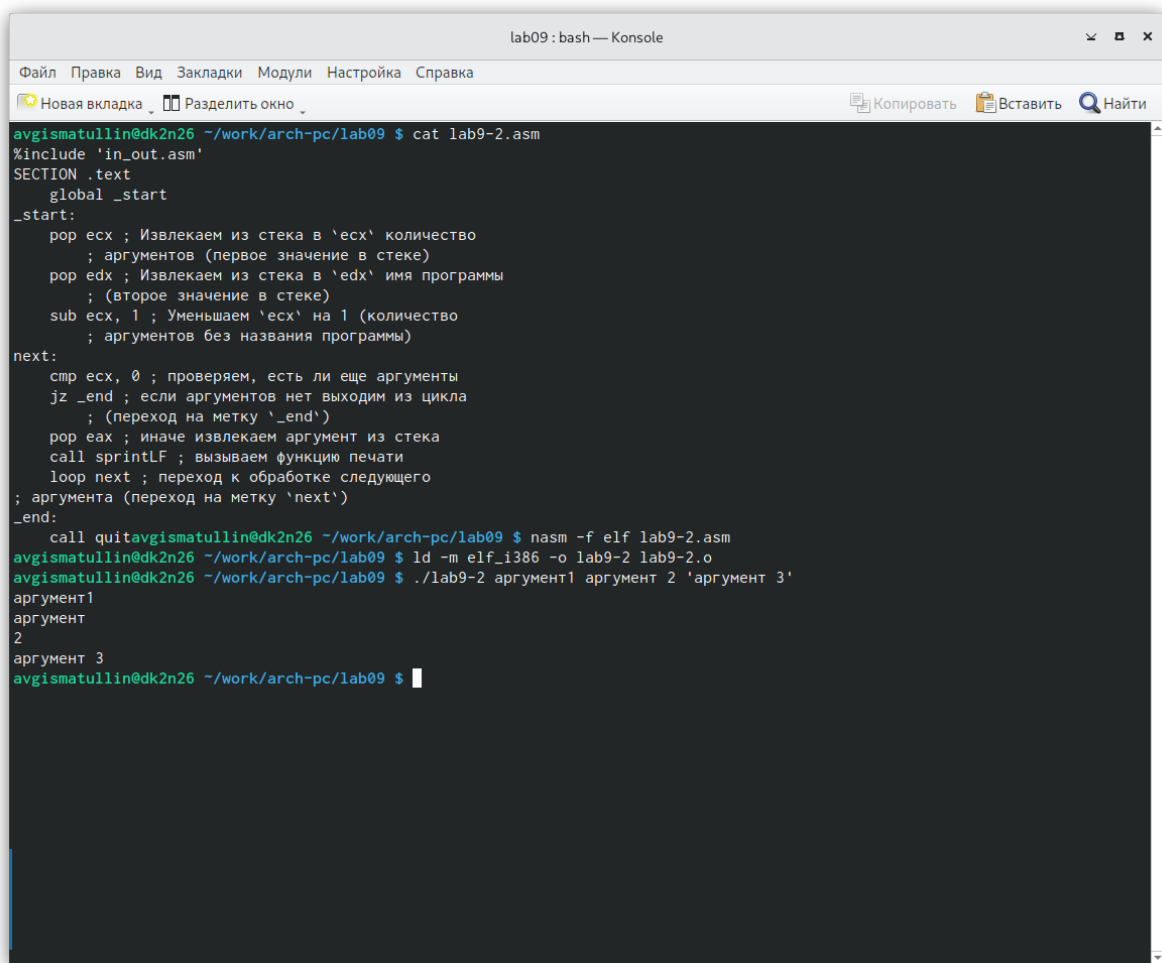
SECTION .text
global _start
_start:

; ---- 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
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ ./lab9-1
Введите N: 5
4
3
2
1
0
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $
```

Рис. 4.5: Командная строка. Измененная программа lab9-1.asm 2

3. Далее мы сталкиваемся в обработке аргументов. Для этого изучим текст программы листинга 9.2 и запустим его (в lab9-2.asm): (рис. 4.6)

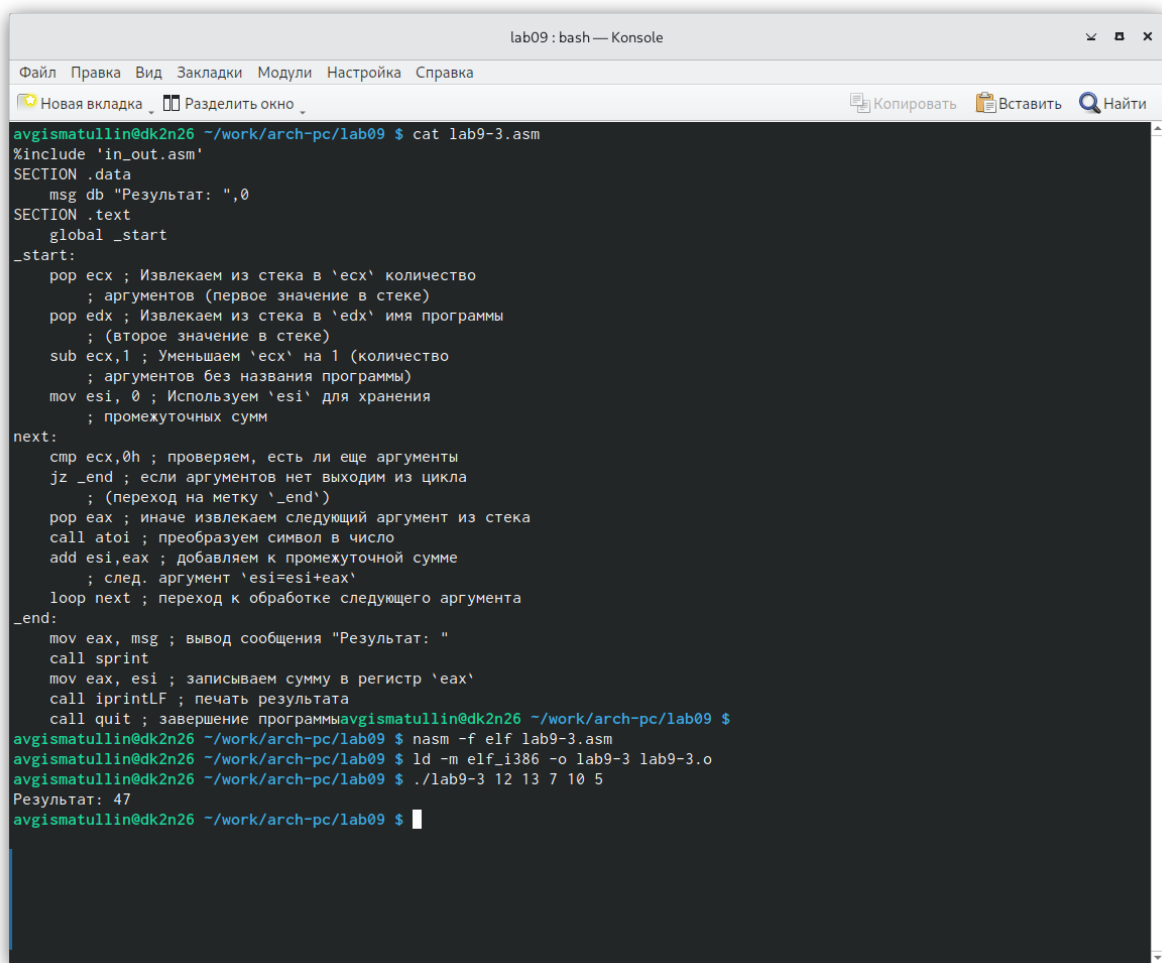


```
lab09: bash — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно
Копировать Вставить Найти
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ cat lab9-2.asm
%include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx ; Извлекаем из стека в 'ecx' количество
              ; 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 printf ; вызываем функцию печати
    loop next ; переход к обработке следующего
; аргумента (переход на метку 'next')
_end:
    call quitavgismatullin@dk2n26 ~/work/arch-pc/lab09 $ nasm -f elf lab9-2.asm
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ ./lab9-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $
```

Рис. 4.6: Командная строка. Выполнение программы lab9-2.asm

При данном вводе программой было обработано 4 аргумента, обозначенные переходом на следующую строку.

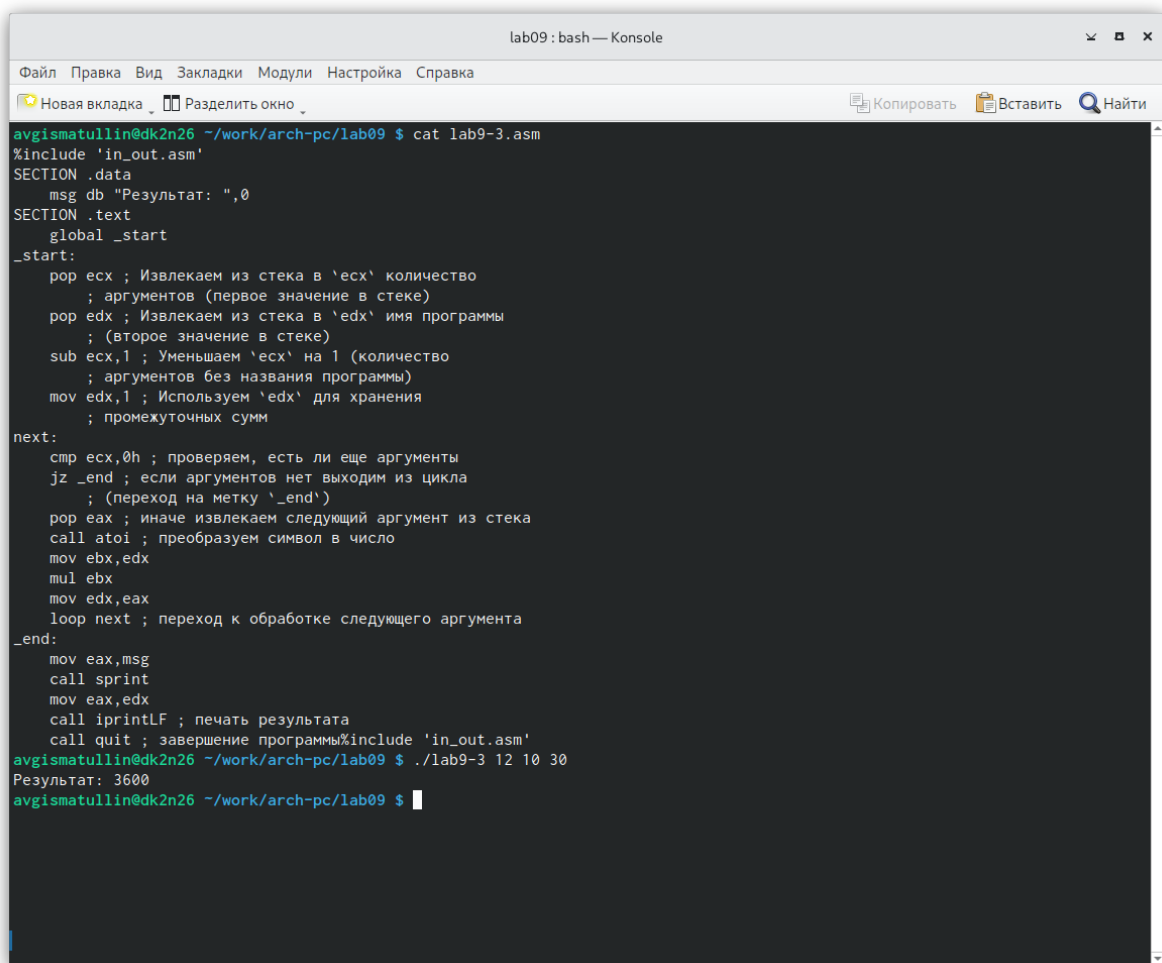
4. Затем создаем файл lab9-3.asm, изучаем программу листинга 9.3 по суммированию всех аргументов и запускаем ее: (рис. 4.7)



```
lab09: bash — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно
Копировать Вставить Найти
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ cat lab9-3.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 iprintf ; печать результата
    call quit ; завершение программы
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ nasm -f elf lab9-3.asm
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-3 lab9-3.o
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ ./lab9-3 12 13 7 10 5
Результат: 47
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $
```

Рис. 4.7: Командная строка. Выполнение программы lab9-3.asm

После этого требуется написать похожую программу, но для нахождения произведения всех элементов. Я реализовал ее следующим образом (изменения внес в самом файле lab9-3.asm - новый не создавал): (рис. 4.8)



```
lab09 : bash — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно
Копировать Вставить Найти
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ cat lab9-3.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 edx,1 ; Используем 'edx' 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,edx
    mul ebx
    mov edx,eax
    loop next ; переход к обработке следующего аргумента
_end:
    mov eax,msg
    call sprint
    mov eax,edx
    call iprintLF ; печать результата
    call quit ; завершение программы%include 'in_out.asm'
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ ./lab9-3 12 10 30
Результат: 3600
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $
```

Рис. 4.8: Командная строка. Выполнение измененной программы lab9-3.asm

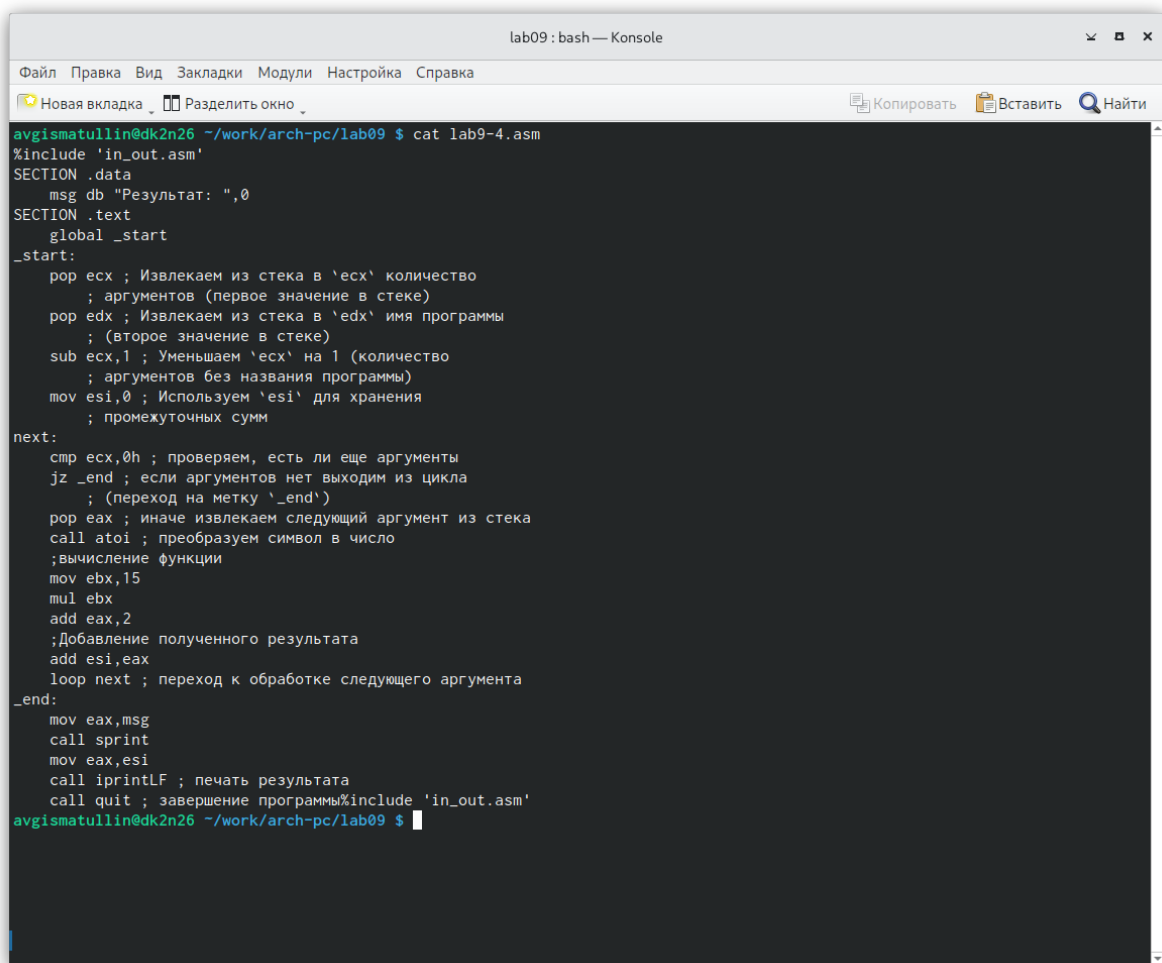
Здесь же мы можем увидеть демонстрацию работоспособности программы.



## 5 Выполнение задания для самостоятельной работы

Суть задания для самостоятельной работы заключается в том, чтобы написать программу для решения некоторой функции, но на вход получить сразу несколько аргументов этой функции. Финальным ответом же будет сумма всех полученных значений. Так как на прошлой лабораторной работе я нашел, что мой вариант - 11, то значения я буду брать соответствующие.

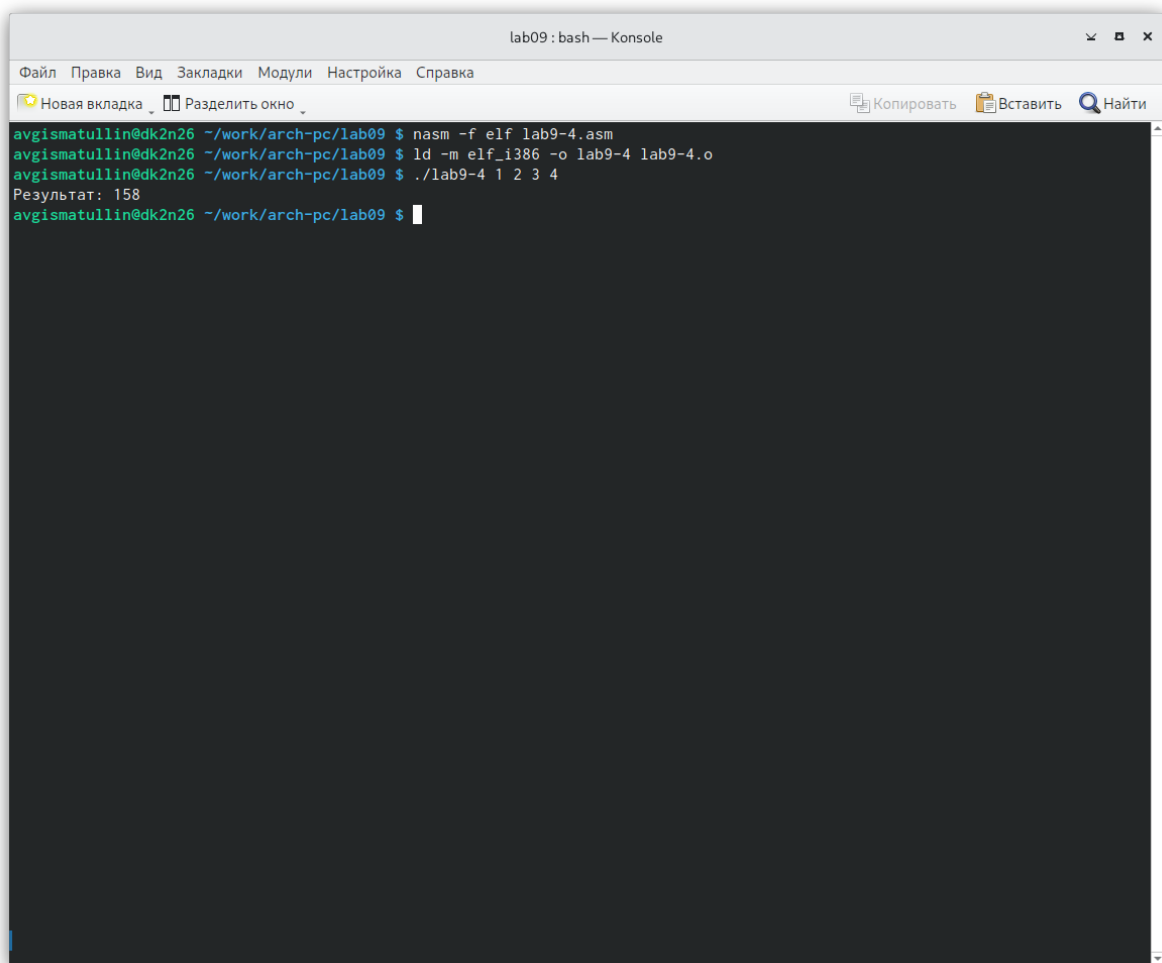
Решил же я это задание следующим образом (файл lab9-4.asm): (рис. 5.1)



```
lab09: bash — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно Копировать Вставить Найти
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ cat lab9-4.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
              ; вычисление функции
    mov ebx,15
    mul ebx
    add eax,2
              ; Добавление полученного результата
    add esi,eax
    loop next ; переход к обработке следующего аргумента
_end:
    mov eax,msg
    call sprint
    mov eax,esi
    call iprintLF ; печать результата
    call quit ; завершение программы%include 'in_out.asm'
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $
```

Рис. 5.1: Командная строка. Программа lab9-4.asm

Демонстрация работоспособности: (рис. 5.2)



```
lab09: bash — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно Копировать Вставить Найти
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ nasm -f elf lab9-4.asm
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-4 lab9-4.o
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $ ./lab9-4 1 2 3 4
Результат: 158
avgismatullin@dk2n26 ~/work/arch-pc/lab09 $
```

Рис. 5.2: Командная строка. Выполнение программы lab9-4.asm

## 6 Выводы

В ходе работы я приобрел навыки написания программ с использованием циклов и обработкой аргументов командной строки.

## **Список литературы**