

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8**

дисциплина: *Архитектура компьютера*

Студент: Трофимов Владислав Алексеевич

Группа: НКАбд-06-25

**МОСКВА**

2025 г.

# **Содержание**

Список иллюстраций .....	3
1 Цель работы .....	4
2 Задание.....	5
3 Теоретическое введение .....	6
4 Выполнение лабораторной работы .....	7
4.1 Реализация циклов в NASM .....	7
4.2 Обработка аргументов командной строки.....	10
4.3 Задание для самостоятельной работы.....	13
5 Выводы .....	16
6 Список литературы .....	17

# **Список иллюстраций**

Рис. 4.1: Создание каталога

Рис. 4.2: Копирование программы из листинга 8.1

Рис. 4.3: Запуск программы

Рис. 4.4: Изменение программы

Рис. 4.5: Запуск измененной программы

Рис. 4.6: Добавление push и pop в цикл программы

Рис. 4.7: Запуск измененной программы

Рис. 4.8: Копирование программы из листинга

Рис. 4.9: Запуск второй программы

Рис. 4.10: Копирование программы из третьего листинга

Рис. 4.11: Запуск третьей программы

Рис. 4.12: Изменение третьей программы

Рис. 4.13: Запуск измененной третьей программы

Рис. 4.14: Написание программы для самостоятельной работы

Рис. 4.15: Запуск программы для самостоятельной работы

# **1 Цель работы**

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

## **2 Задание**

1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. Самостоятельное написание программы по материалам лабораторной работы

### **3 Теоретическое введение**

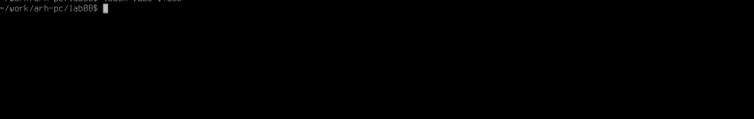
Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды.

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.

## **4 Выполнение лабораторной работы**

## 4.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8 (рис. 4.1).



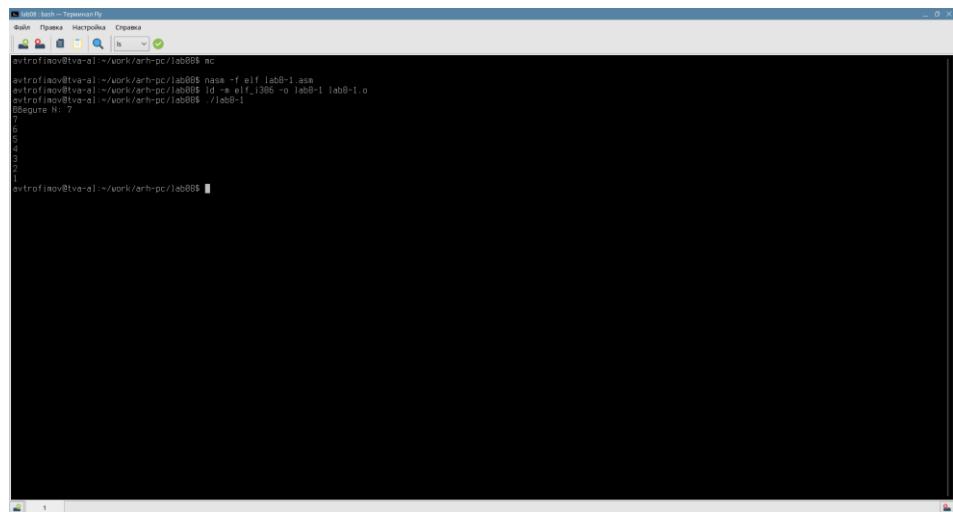
```
avtrolimov@vive-al: ~$ mkdir ~/work/arth-pc/lab88  
avtrolimov@vive-al: ~$ cd ~/work/arth-pc/lab88  
avtrolimov@vive-al:~/work/arth-pc/lab88$ touch lab88-1.ass  
avtrolimov@vive-al:~/work/arth-pc/lab88$
```

Рис. 4.1: Создание каталога

Копирую в созданный файл программу из листинга. (рис. 4.2).

Рис. 4.2: Копирование программы из листинга 8.1

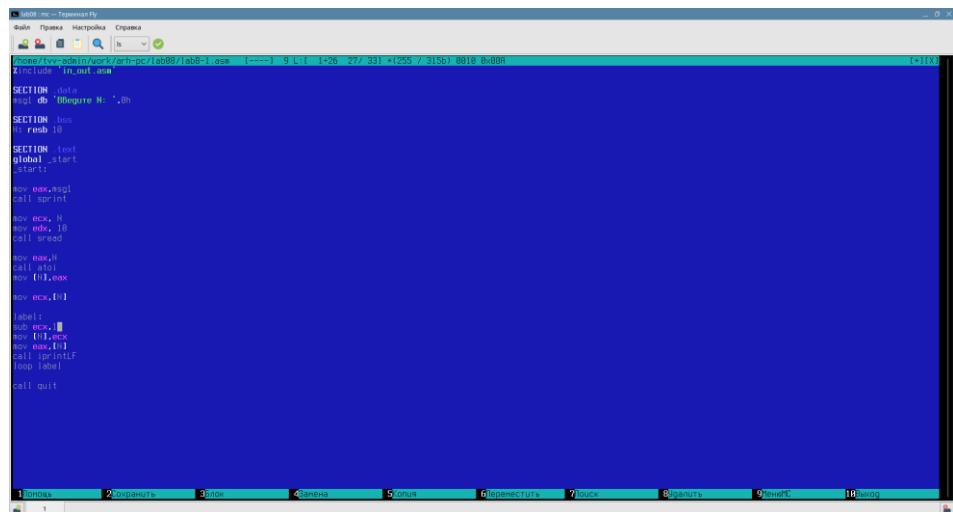
Запускаю программу, она показывает работу циклов в NASM (рис. 4.3).



```
lab08_1c - Терминал РУ
Файл Правка Настройка Справка
[?] [?] [?] [?] [?] [?]
astroimov@tv-a1:~/work/arth-pc/lab08$ nasm -f elf lab08-1.asm
astroimov@tv-a1:~/work/arth-pc/lab08$ ld -e elf_i386 -o lab08-1 lab08-1.o
astroimov@tv-a1:~/work/arth-pc/lab08$ ./lab08-1
Beginne N: 7
5
4
3
2
1
astroimov@tv-a1:~/work/arth-pc/lab08$
```

Рис. 4.3: Запуск программы

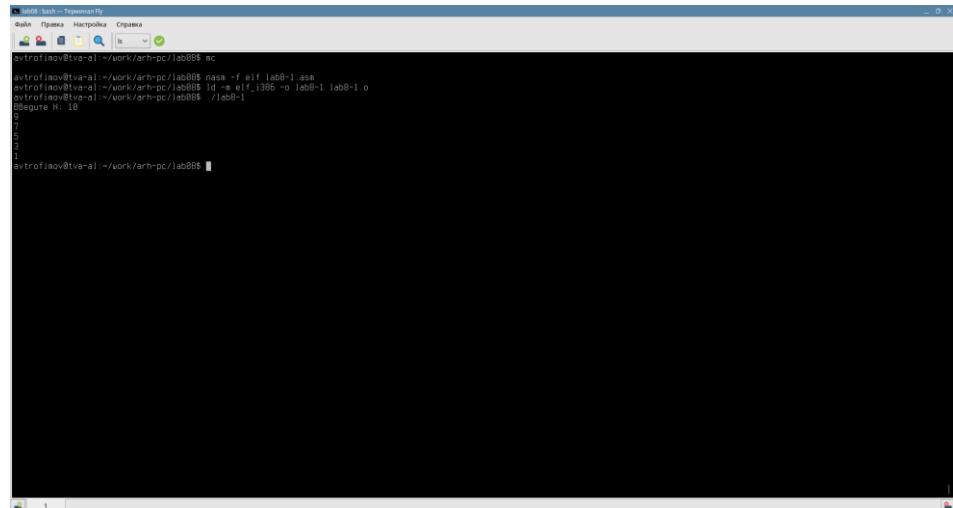
Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра ecx (рис. 4.4).



```
lab08_1c - Терминал РУ
Файл Правка Настройка Справка
[?] [?] [?] [?] [?] [?]
astroimov@tv-a1:~/work/arth-pc/lab08$ nasm -f elf lab08-1.asm
astroimov@tv-a1:~/work/arth-pc/lab08$ include 'in_out.asm'
SECTION data
msg db "Beginne N: ",0h
SECTION bss
resb 10
SECTION text
global _start
_start:
    mov eax,0
    call sprint
    mov ebx,0
    mov edx,10
    call read
    mov ecx,0
    call atoi
    mov ebx,0
    call quit
label:
    db ecx,0
    mov ebx,0
    mov ecx,0
    call writeLF
    loop label
    call quit
astroimov@tv-a1:~/work/arth-pc/lab08$
```

Рис. 4.4: Изменение программы

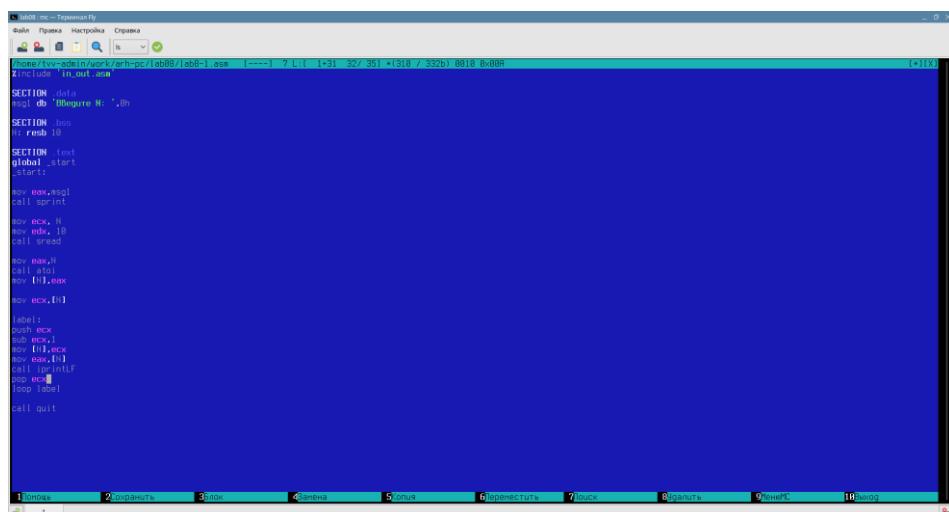
Из-за того, что теперь регистр `esx` на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. 4.5).



```
autrofimov@tv-a1:~/work/arch-pc/lab08$ nasm -f elf lab08-1.asm
autrofimov@tv-a1:~/work/arch-pc/lab08$ ld -e elf_i386 -o lab08-1 lab08-1.o
autrofimov@tv-a1:~/work/arch-pc/lab08$ ./lab08-1
autrofimov@tv-a1:~/work/arch-pc/lab08$
```

Рис. 4.5: Запуск измененной программы

Добавляю команды `push` и `pop` в программу (рис. 4.6).



```
autrofimov@tv-a1:~/work/arch-pc/lab08$ nasm -f elf lab08-1.asm
autrofimov@tv-a1:~/work/arch-pc/lab08$ ld -e lab08-1 lab08-1.o
autrofimov@tv-a1:~/work/arch-pc/lab08$ ./lab08-1
autrofimov@tv-a1:~/work/arch-pc/lab08$
```

Рис. 4.6: Добавление `push` и `pop` в цикл программы

Теперь количество итераций совпадает введенному  $N$ , но произошло смещение выводимых чисел на -1 (рис. 4.7).

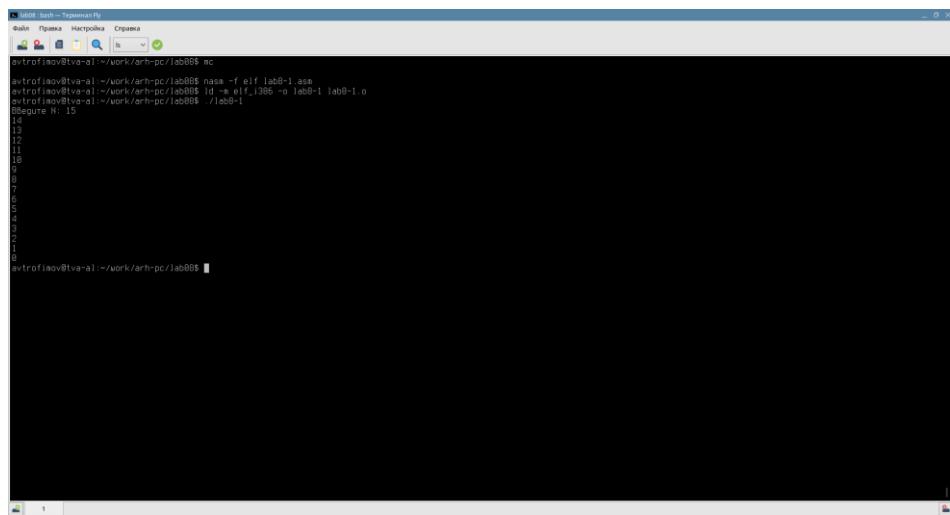


Рис. 4.7: Запуск измененной программы

## 4.2 Обработка аргументов командной строки

Создаю новый файл для программы и копирую в него код из следующего листинга (рис. 4.8).

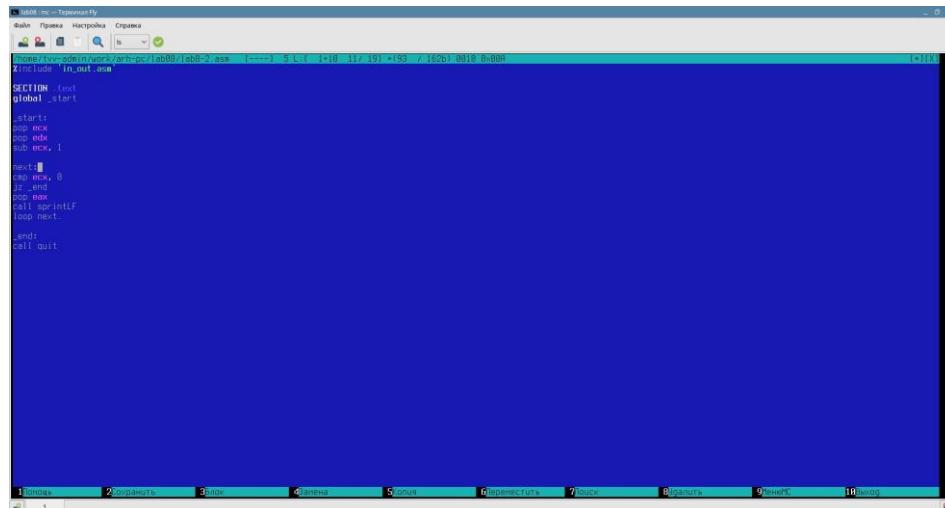


Рис. 4.8: Копирование программы из листинга

Компилирую программу и запускаю, указав аргументы. Программой было выведено обратно тоже количество аргументов, что и было введено (рис. 4.9).

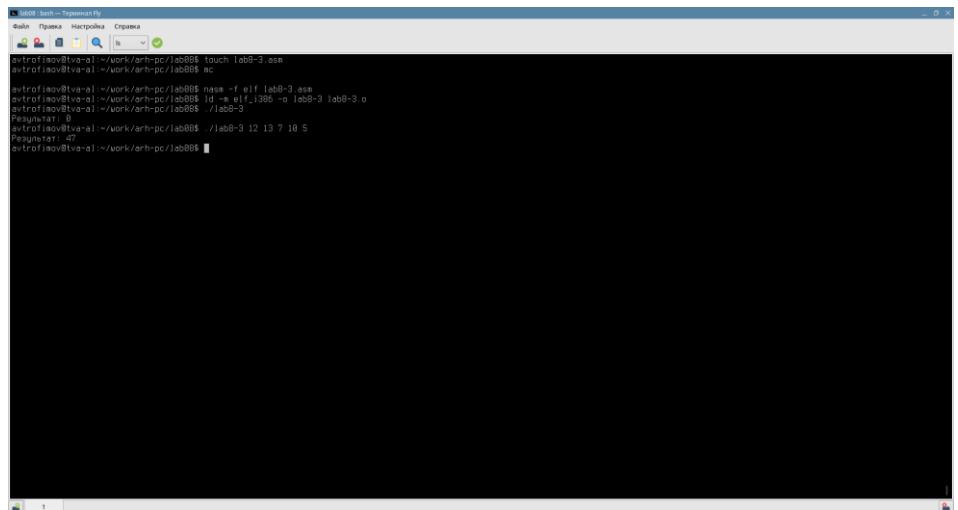
```
avtrolimov@tva-s1: ~$ /work/arh-pc/lab888 ./lab8-2 arg1 arg2 arg3
avtrolimov@tva-s1: ~$
```

Рис. 4.9: Запуск второй программы

Создаю новый файл для программы и копирую в него код из третьего листинга (рис. 4.10).

Рис. 4.10: Копирование программы из третьего листинга

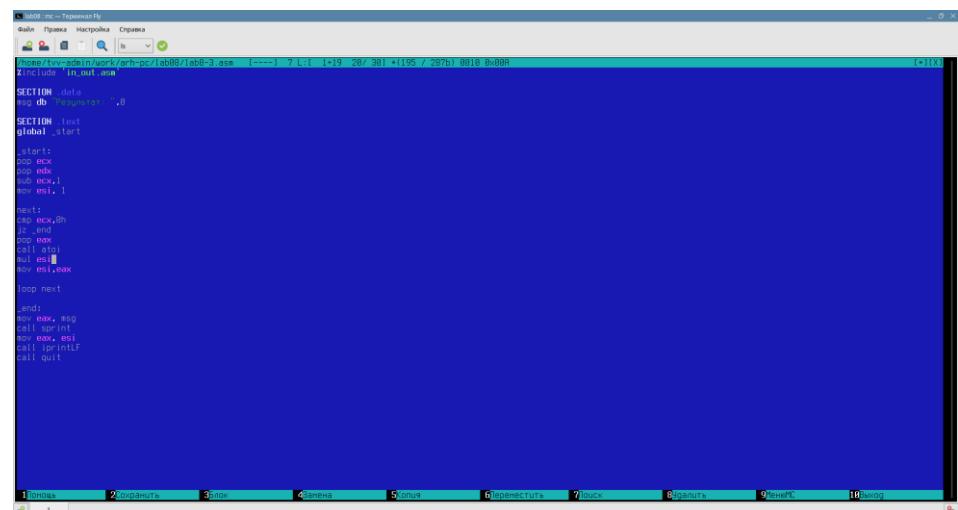
Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. 4.11).



```
estrofimov@tv-a-01:~/work/arm-h-pc/lab03$ touch lab03.asm
estrofimov@tv-a-01:~/work/arm-h-pc/lab03$ nasm -f elf lab03.asm
estrofimov@tv-a-01:~/work/arm-h-pc/lab03$ ld -e _start -o lab03 lab03.o
estrofimov@tv-a-01:~/work/arm-h-pc/lab03$ ./lab03
Input A: 8
Input B: 5
Result: 40
estrofimov@tv-a-01:~/work/arm-h-pc/lab03$
```

Рис. 4.11: Запуск третьей программы

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. 4.12).



```
estrofimov@tv-a-01:~/work/arm-h-pc/lab03$ nasm -f elf lab03.asm
estrofimov@tv-a-01:~/work/arm-h-pc/lab03$ ld -e _start -o lab03 lab03.o
estrofimov@tv-a-01:~/work/arm-h-pc/lab03$ ./lab03
Input A: 8
Input B: 5
Result: 40
estrofimov@tv-a-01:~/work/arm-h-pc/lab03$
```

```
ESTROFILOV@TV-A-01:~/work/arm-h-pc/lab03$ nasm -f elf lab03.asm
ESTROFILOV@TV-A-01:~/work/arm-h-pc/lab03$ ld -e _start -o lab03 lab03.o
ESTROFILOV@TV-A-01:~/work/arm-h-pc/lab03$ ./lab03
Input A: 8
Input B: 5
Result: 40
ESTROFILOV@TV-A-01:~/work/arm-h-pc/lab03$
```

Рис. 4.12: Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. 4.13).

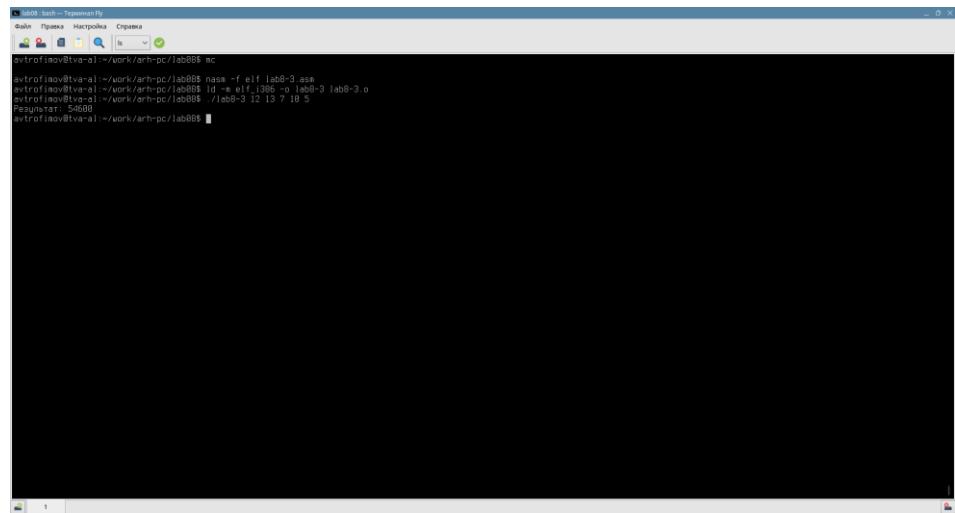


Рис. 4.13: Запуск измененной третьей программы

### 4.3 Задание для самостоятельной работы

Пишу программму, которая будет находить сумму значений для функции  $f(x) = 2x+15$ , которая совпадает с моим девытым вариантом (рис. 4.14).

```

%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 2x + 15, 0
msg_result db "Результат: ", 0h

SECTION .text
GLOBAL _start

_start:
    mov eax, msg_func
    call sprintLF
    pop ebx
    pop edx
    sub eax, 1
    mov esi, 0

.next:
    mov ebx, 0h
    jz .end
    pop eax
    call atoi
    mov ebx, 2
    mul ebx
    add ebx, 15
    add esi, ebx
    loop .next

.end:
    mov eax, msg_result
    call sprint
    mov eax, 0h
    call printLF
    call quit

```

Рис. 4.14: Написание программы для самостоятельной работы

Код программы:

```
%include 'in_out.asm'

SECTION .data

msg_func db "Функция: f(x) = 2x + 15, 0
msg_result db "Результат: ", 0
```

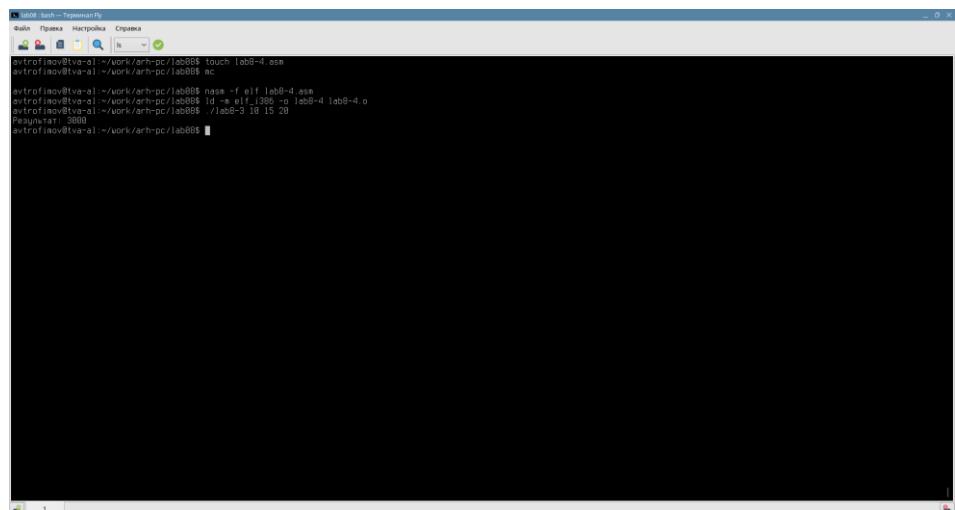
```
SECTION .text
GLOBAL _start

_start:
    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_result
    call sprint
    mov eax, esi
    call iprintLF
    call quit
```

Проверяю работу программы, указав в качестве аргумента  
несколько чисел (рис. 4.15).



```
labb08 [root] - Терминал 1
Файл Правка Настройка Справка ls
avtrolinov@tiva-a1:~/work/arm-pc/labb08$ touch labb-4.sas
avtrolinov@tiva-a1:~/work/arm-pc/labb08$ mc
avtrolinov@tiva-a1:~/work/arm-pc/labb08$ nasm -f elf labb-4.sas
avtrolinov@tiva-a1:~/work/arm-pc/labb08$ ld -m elf_i386 -o labb-4 labb-4.o
avtrolinov@tiva-a1:~/work/arm-pc/labb08$ ./labb-3 18 15 20
Program terminated.
avtrolinov@tiva-a1:~/work/arm-pc/labb08$
```

Рис. 4.15: Запуск программы для самостоятельной работы

## **5 Выводы**

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием циклов а также научился обрабатывать аргументы командной строки.

## **6 Список литературы**

1. Курс ТУИС

2. Лабораторная работа №7

3. Программирование на языке ассемблера NASM Столяров А.

В.