

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).

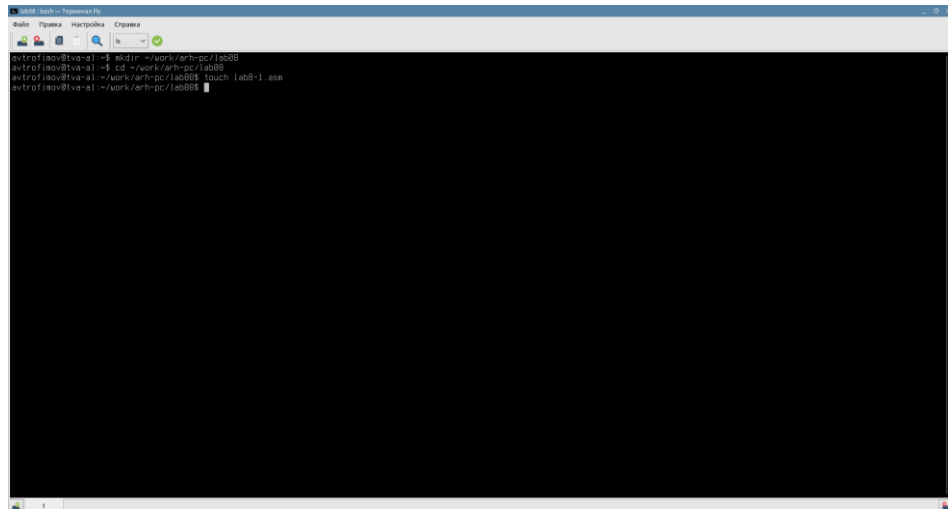


Рис. 4.1: Создание каталога

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

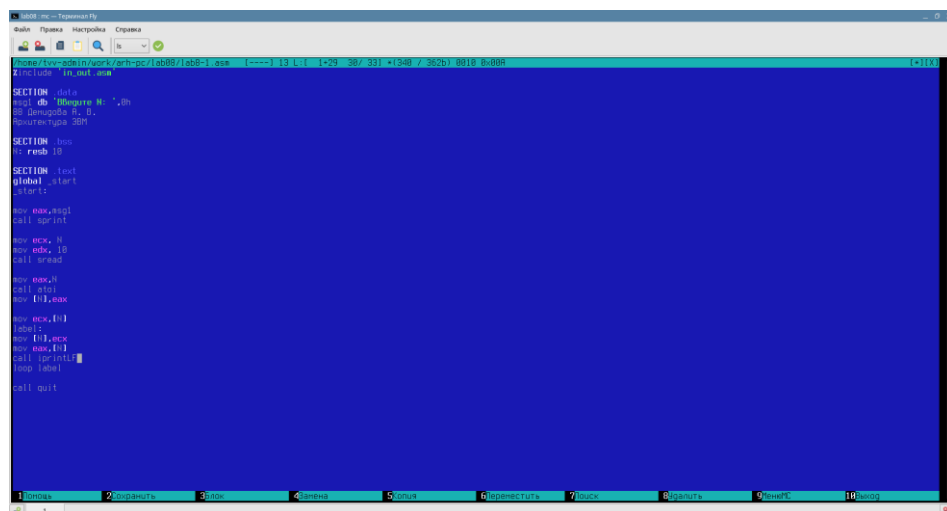
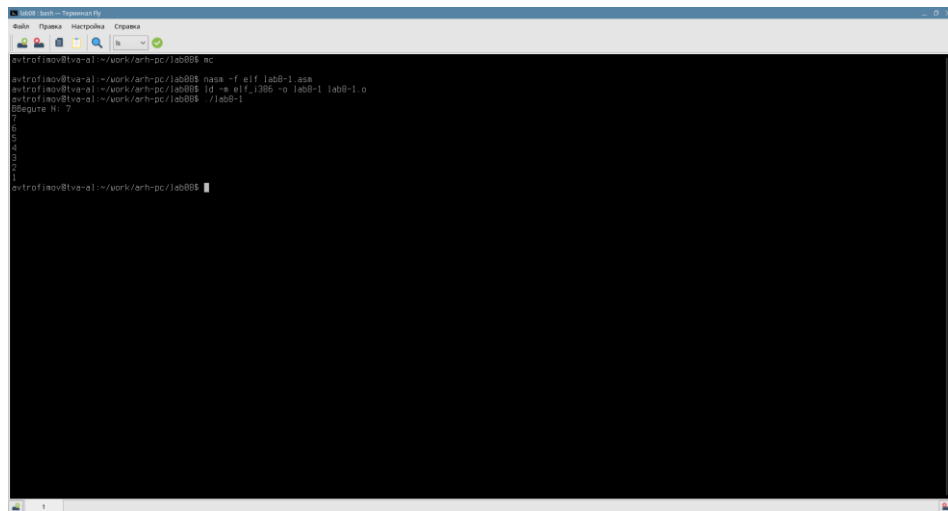


Рис. 4.2: Копирование программы из листинга 8.1

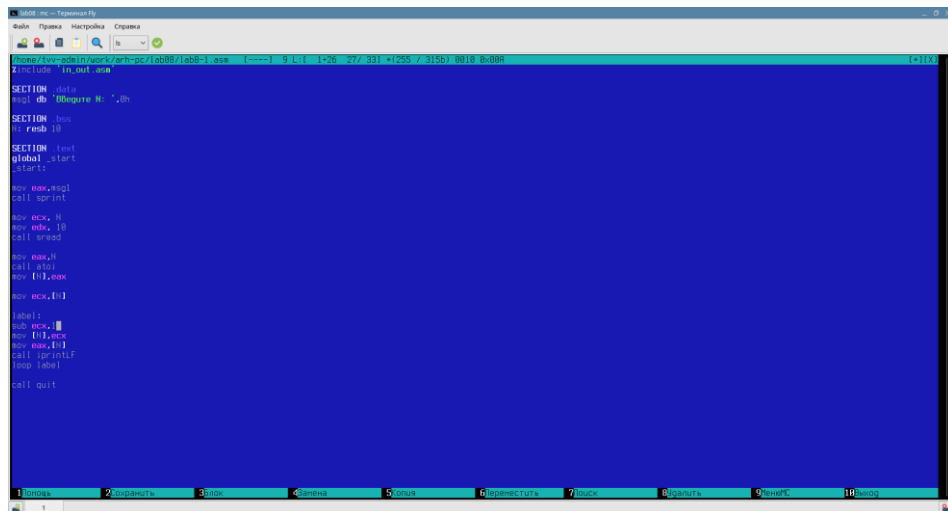
Запускаю программу, она показывает работу циклов в NASM (рис. 4.3).



```
avtrofiav@tva-al:~/work/armhpc/lab808$ mc
avtrofiav@tva-al:~/work/armhpc/lab808$ nasm -f elf lab8-1.asm
avtrofiav@tva-al:~/work/armhpc/lab808$ ld -s elf_1385 -o lab8-1 lab8-1.o
avtrofiav@tva-al:~/work/armhpc/lab808$ ./lab8-1
$debug N: 7
6
5
4
3
2
1
avtrofiav@tva-al:~/work/armhpc/lab808$
```

Рис. 4.3: Запуск программы

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



```
SECTION .data
msg db "Debug N: ",0h

SECTION .text
global _start
_start:
mov eax,msg
call printf

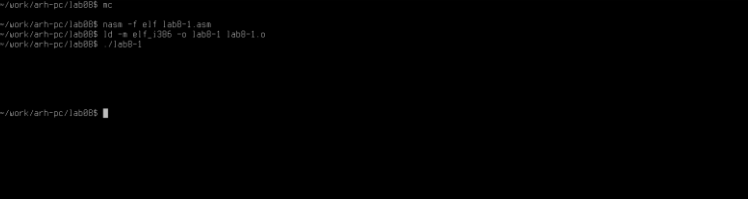
mov ecx,N
mov edx,10
call read

mov eax,N
call atoi
mov [ecx],eax

mov ecx,[ecx]
label:
sub ecx,1
mov [ecx],eax
call printf
call printf
loop label

call quit
```

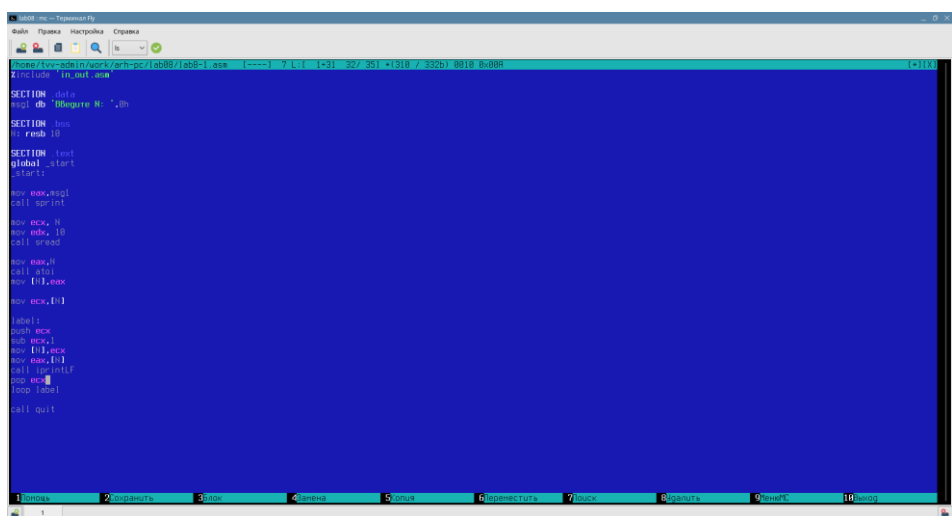
Рис. 4.4: Изменение программы



The screenshot shows a terminal window with the following commands and output:

```
evtrofiasv@tva-al:~/work/arth-pc/lab005$ nc
evtrofiasv@tva-al:~/work/arth-pc/lab005$ nasm -f elf lab0-1.asm
evtrofiasv@tva-al:~/work/arth-pc/lab005$ ld -s elf.386 -o lab0-1 lab0-1.o
evtrofiasv@tva-al:~/work/arth-pc/lab005$ ./lab0-1
Segmentation fault
0
0
0
0
1
evtrofiasv@tva-al:~/work/arth-pc/lab005$
```

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



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

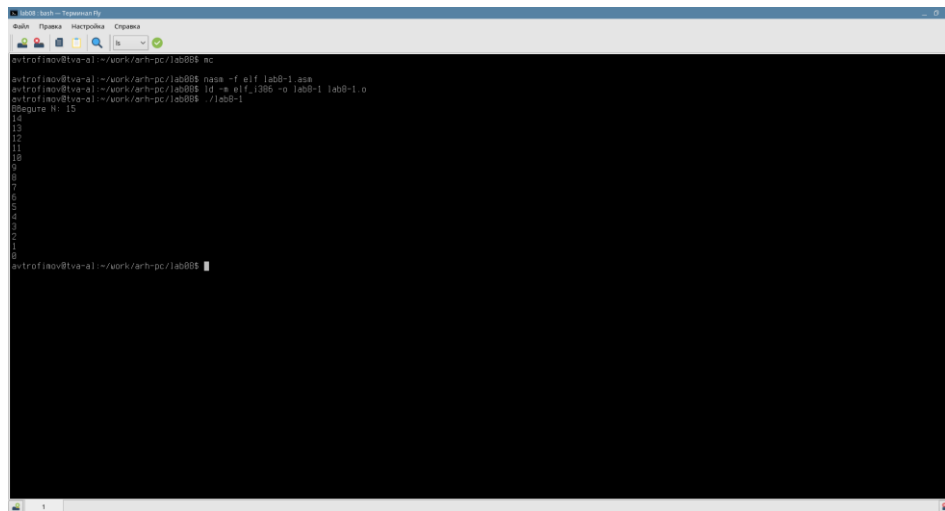


Рис. 4.7: Запуск измененной программы

4.2 Обработка аргументов командной строки

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

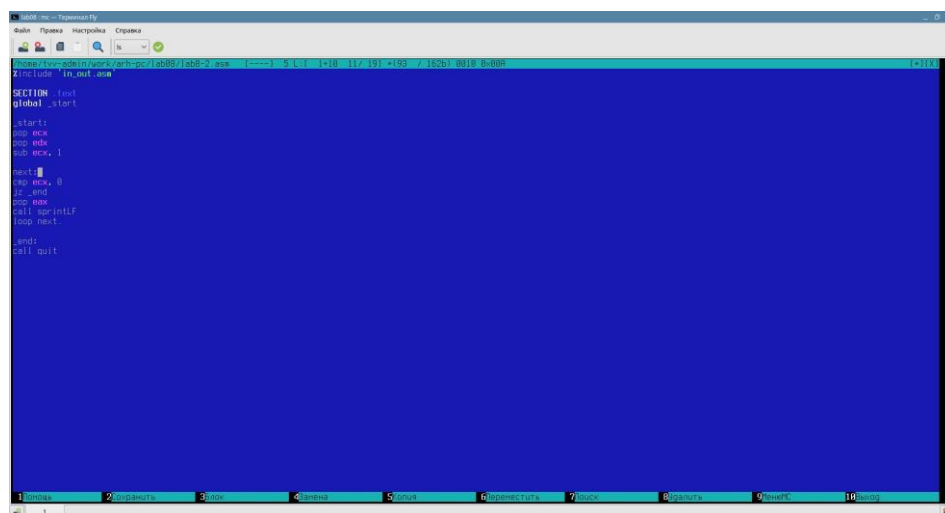


Рис. 4.8: Копирование программы из листинга

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

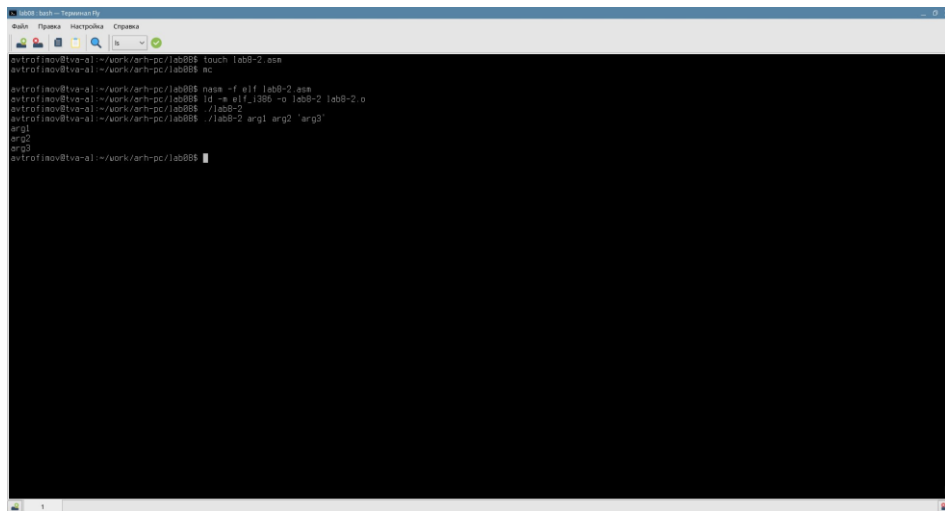


Рис. 4.9: Запуск второй программы

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

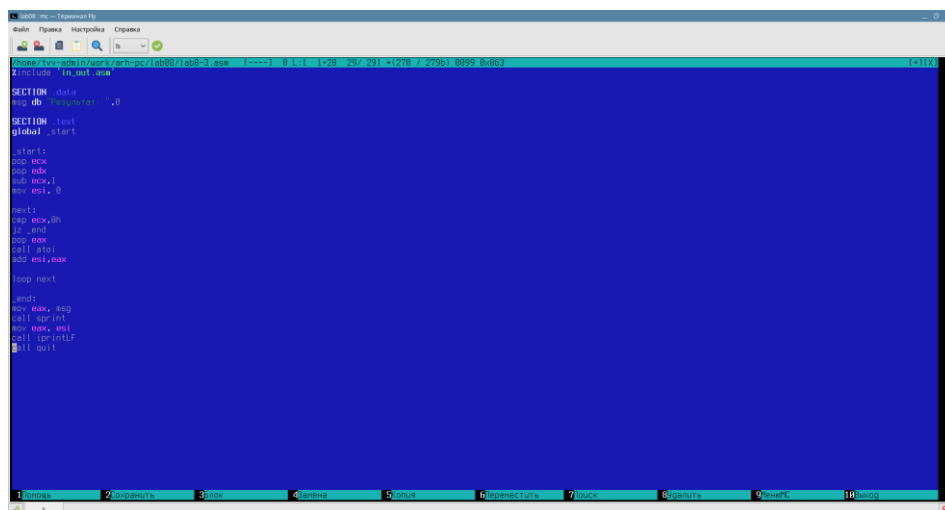
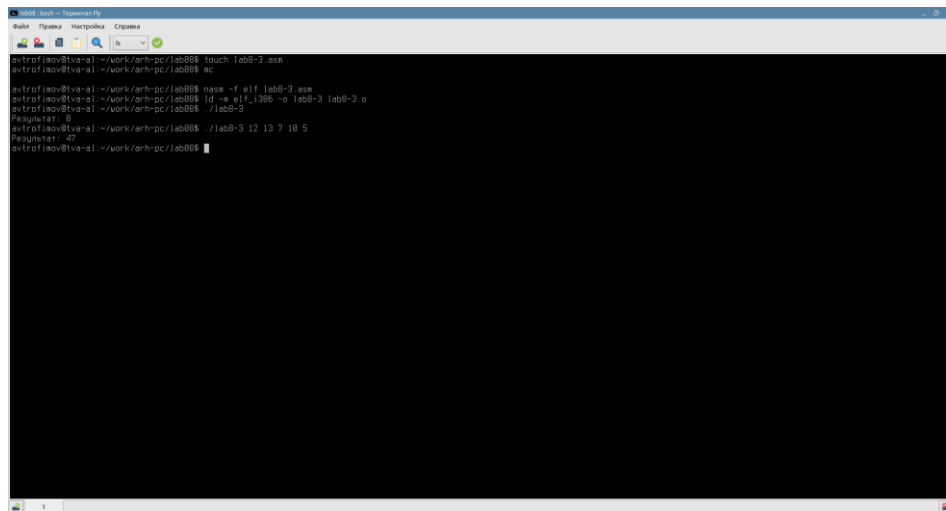


Рис. 4.10: Копирование программы из третьего листинга

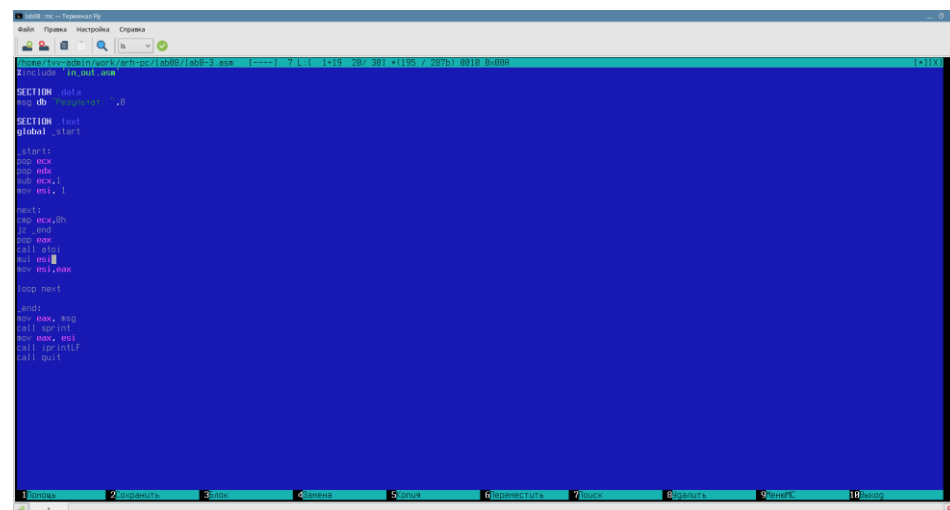
Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. 4.11).



```
avtrofiav@vavr-1:~/work/arm-pc/lab08$ touch lab0-3.asm
avtrofiav@vavr-1:~/work/arm-pc/lab08$ nc
avtrofiav@vavr-1:~/work/arm-pc/lab08$ nasm -f elf lab0-3.asm
avtrofiav@vavr-1:~/work/arm-pc/lab08$ ld -m elf_386 -o lab0-3 lab0-3.o
avtrofiav@vavr-1:~/work/arm-pc/lab08$ ./lab0-3
Rezultat: 0
avtrofiav@vavr-1:~/work/arm-pc/lab08$ ./lab0-3 12 13 7 10 5
Rezultat: 47
avtrofiav@vavr-1:~/work/arm-pc/lab08$
```

Рис. 4.11: Запуск третьей программы

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



```
lab08.asm - Text Editor
/home/avtrofiav/work/arm-pc/lab08/lab0-3.asm 7 L 1 1+19 20/ 30 1+195 / 20767 8918 8-808
#include "in_out.asm"

SECTION .data
msg db "Rezultat: ",0

SECTION .text
global _start

_start:
    mov ecx,0
    mov ebx,1
    mov esi,1

next:
    mov ecx,ebx
    jmp end
    mov ecx,esi
    call atoi
    mov esi,esi
    mov ebx,ecx
    jmp next

end:
    mov ebx,esp
    call _write
    mov ebx,esi
    call _printf
    call _quit
```

Рис. 4.12: Изменение третьей программы

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

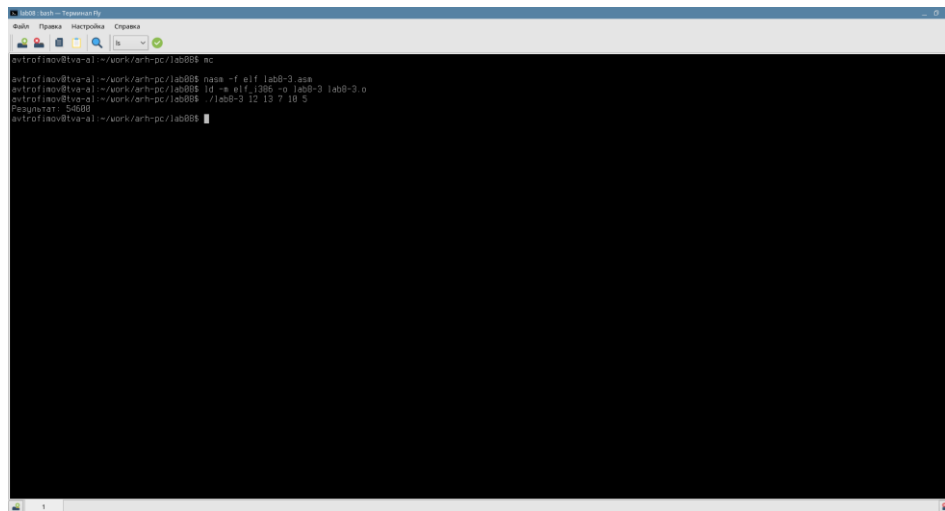


Рис. 4.13: Запуск измененной третьей программы

4.3 Задание для самостоятельной работы

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

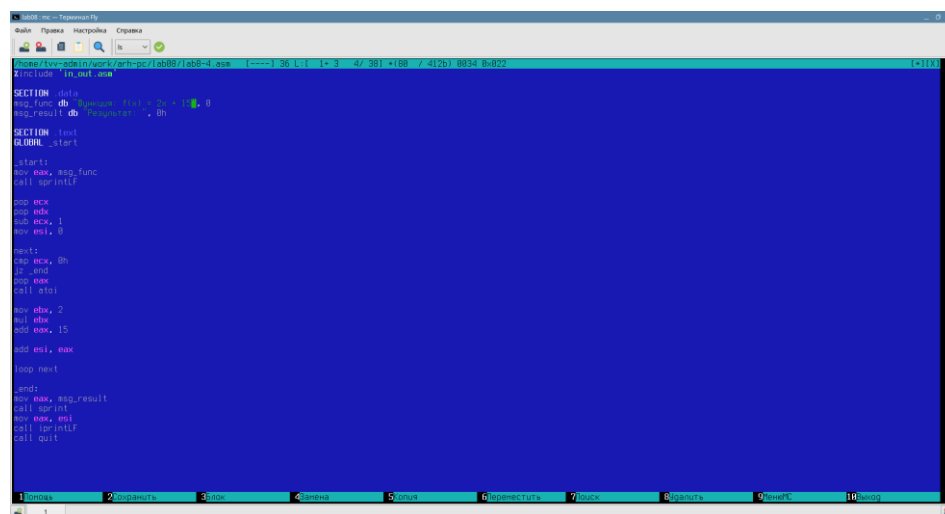


Рис. 4.14: Написание программы для самостоятельной работы

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

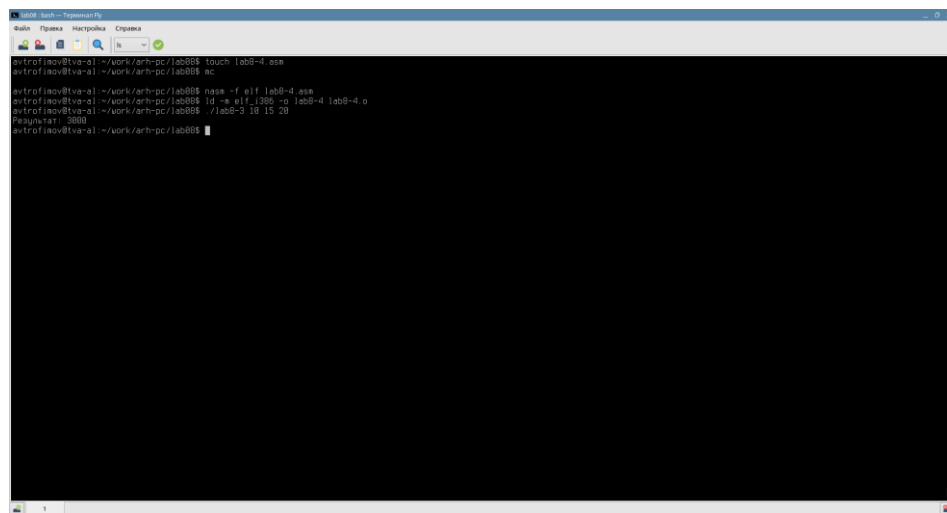
```
msg_func db "Функция:  $f(x) = 2x + 15$ ", 0
```

```
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).



The image shows a terminal window titled "lab08 - ssh - Terminals". The terminal displays the following commands and output:

```
astrofinaov@tva-1: ~/work/ark-ipc/lab008$ touch lab08-4.ssa
astrofinaov@tva-1: ~/work/ark-ipc/lab008$ nc

astrofinaov@tva-1: ~/work/ark-ipc/lab008$ nasm -f elf lab08-4.ssa
astrofinaov@tva-1: ~/work/ark-ipc/lab008$ ld -s -elf lab08-4.o
astrofinaov@tva-1: ~/work/ark-ipc/lab008$ ./lab08-3 10 15 20
Пауза: 2000
astrofinaov@tva-1: ~/work/ark-ipc/lab008$
```

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.