

Отчёт по лабораторной работе №8

Дисциплина: архитектура компьютера

Учаева Алёна Сергеевна

Содержание

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
	Список литературы	17

Список иллюстраций

4.1	Создание каталога	7
4.2	Редактирование файла	8
4.3	Запуск исполняемого файла	8
4.4	Редактирование файла	9
4.5	Запуск исполняемого файла	9
4.6	Редактирование файла	10
4.7	Запуск исполняемого файла	10
4.8	Создание файла	11
4.9	Запуск исполняемого файла	11
4.10	Создание файла	12
4.11	Запуск исполняемого файла	12
4.12	Редактирование файла	13
4.13	Запуск исполняемого файла	13
4.14	Создание файла	14
4.15	Запуск исполняемого файла	14

1 Цель работы

Приобрести навыки написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Задание для самостоятельной работы

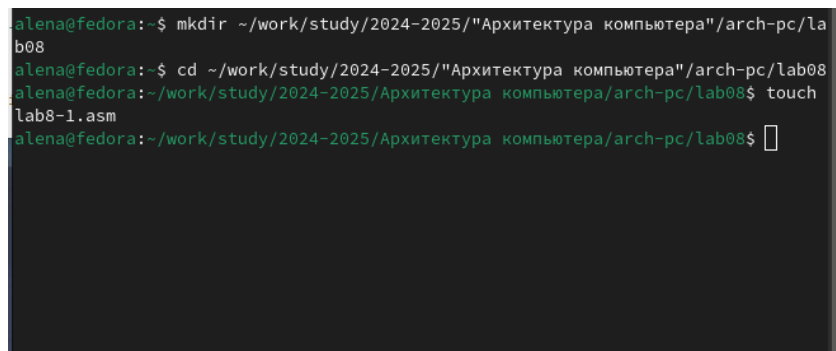
3 Теоретическое введение

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. 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).

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

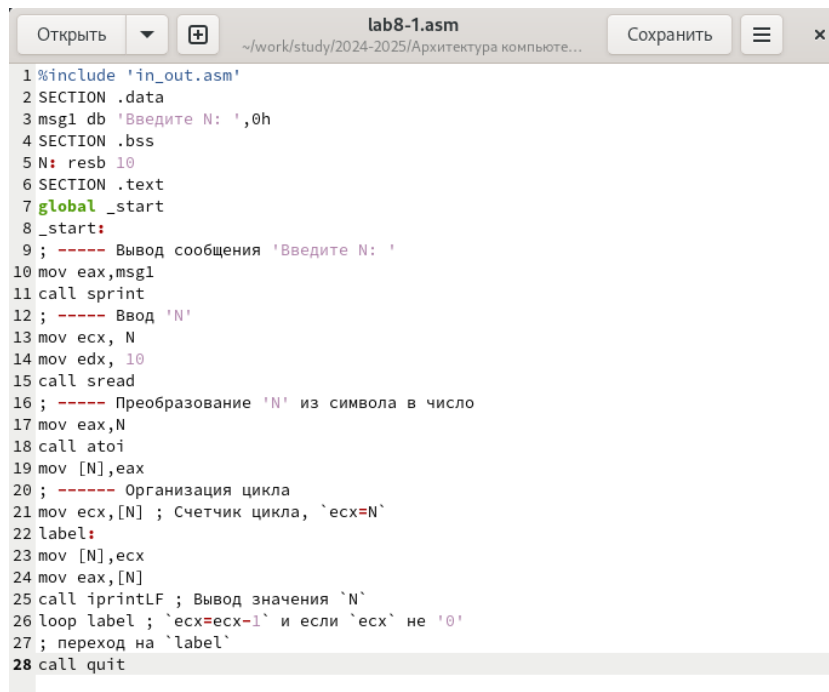
Создаю каталог для программ лабораторной работы №8(рис. 4.1).



```
alena@fedora: ~$ mkdir ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/lab08
alena@fedora: ~$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/lab08
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ touch lab8-1.asm
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

Рис. 4.1: Создание каталога

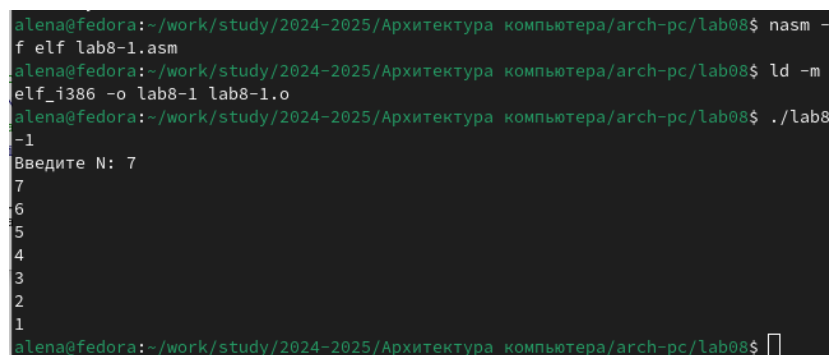
В созданный файл копирую программу из листинга(рис. 4.2).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 mov [N],ecx
24 mov eax,[N]
25 call iprintLF ; Вывод значения `N`
26 loop label ; `ecx=ecx-1` и если `ecx` не `0`
27 ; переход на `label`
28 call quit
```

Рис. 4.2: Редактирование файла

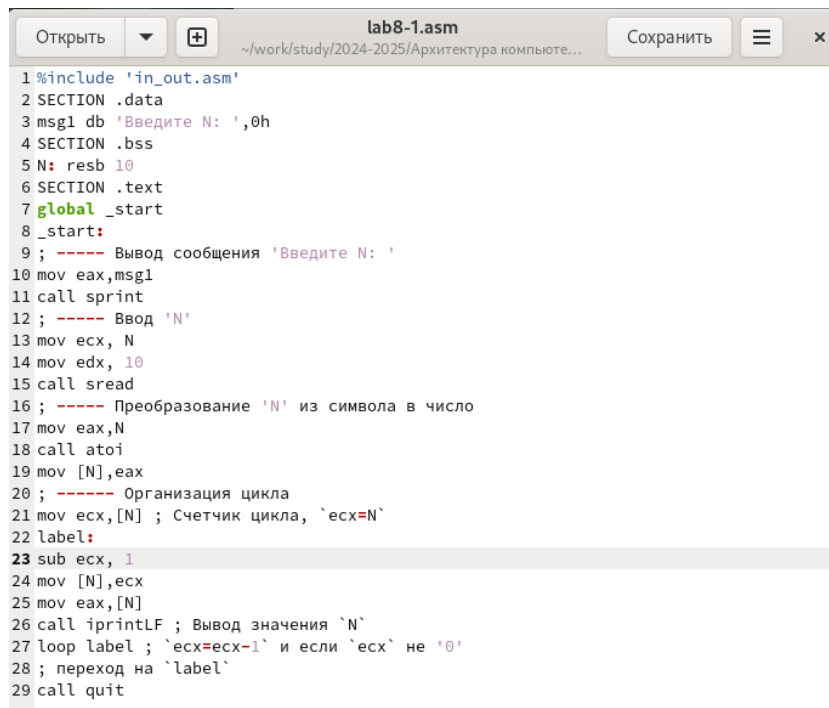
Создаю исполняемый файл и запускаю его, программа показывает работу циклов в NASM(рис. 4.3).



```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-1.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
-1
Введите N: 7
7
6
5
4
3
2
1
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

Рис. 4.3: Запуск исполняемого файла

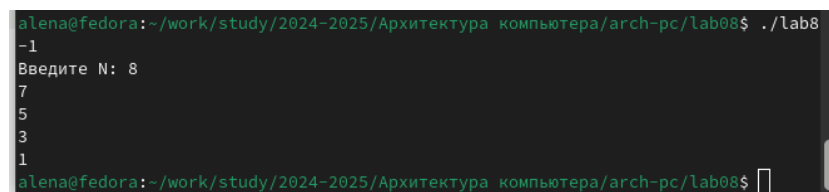
Изменяю текст программы, добавляю изменения значения регистра ecx в цикле(рис. 4.4).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 sub ecx, 1
24 mov [N],ecx
25 mov eax,[N]
26 call iprintf ; Вывод значения `N`
27 loop label ; `ecx=ecx-1` и если `ecx` не `0`
28 ; переход на `label`
29 call quit
```

Рис. 4.4: Редактирование файла

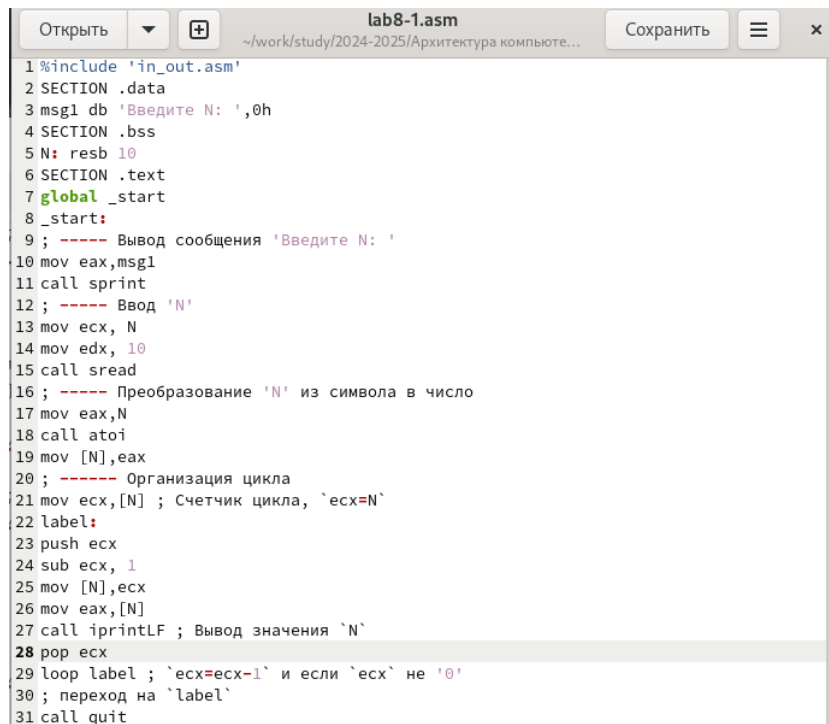
Создаю исполняемый файл и запускаю его, теперь ecx на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое(рис. 4.5).



```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab8
-1
Введите N: 8
7
5
3
1
1
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

Рис. 4.5: Запуск исполняемого файла

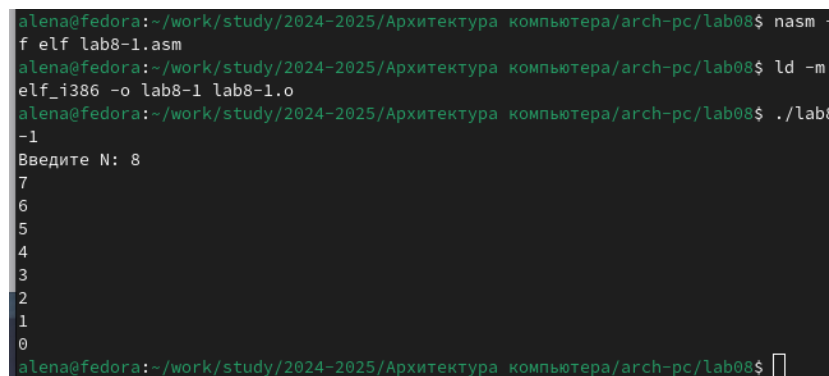
Добавляю команды push и pop в программу(рис. 4.6).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 push ecx
24 sub ecx, 1
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF ; Вывод значения `N`
28 pop ecx
29 loop label ; `ecx=ecx-1` и если `ecx` не `0`
30 ; переход на `label`
31 call quit
```

Рис. 4.6: Редактирование файла

Создаю исполняемый файл и запускаю его, количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1(рис. 4.7).



```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-1.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
Введите N: 8
7
6
5
4
3
2
1
0
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

Рис. 4.7: Запуск исполняемого файла

4.2 Обработка аргументов командной строки

Создаю новый файл lab8-2.asm и копирую в него код из листинга №2(рис. 4.8).

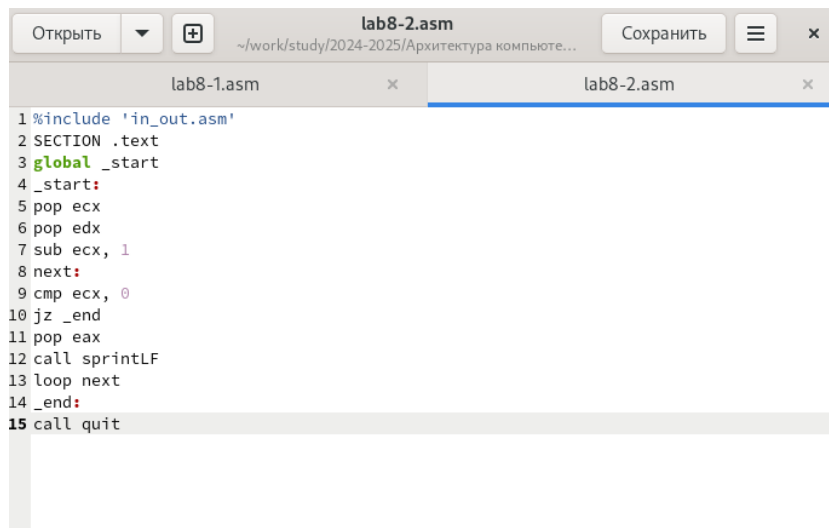


Рис. 4.8: Создание файла

Создаю исполняемый файл и запускаю его, указав аргументы, программа обработала тоже количество аргументов, что и было введено(рис. 4.9).

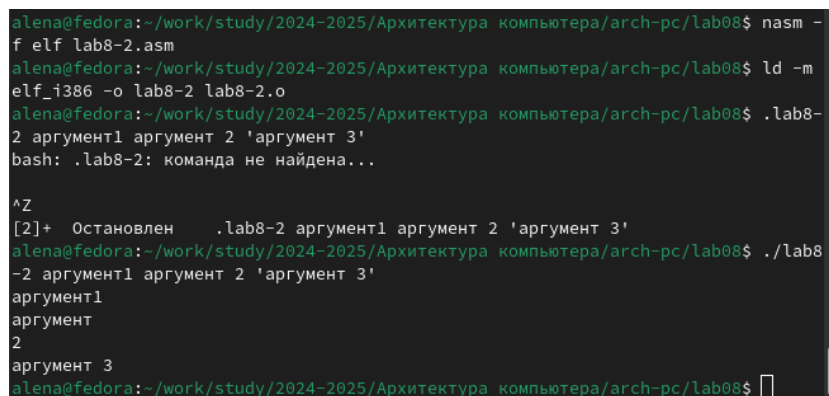
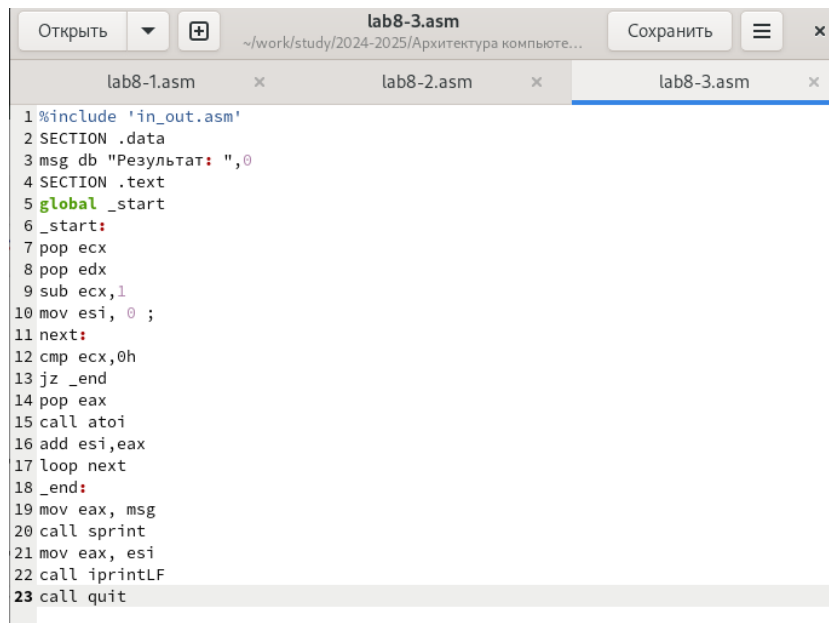


Рис. 4.9: Запуск исполняемого файла

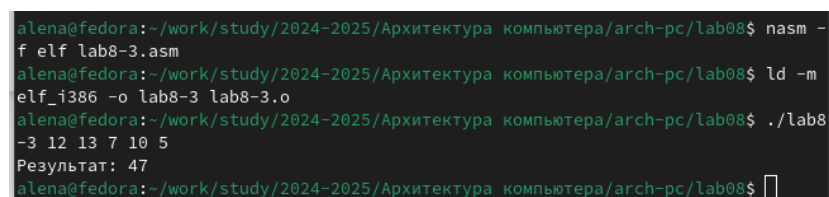
Создаю новый файл lab8-3.asm и ввожу в него текст программы из листинга №3(рис. 4.10).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx
8 pop edx
9 sub ecx,1
10 mov esi, 0 ;
11 next:
12 cmp ecx,0h
13 jz _end
14 pop eax
15 call atoi
16 add esi,eax
17 loop next
18 _end:
19 mov eax, msg
20 call sprint
21 mov eax, esi
22 call iprintLF
23 call quit
```

Рис. 4.10: Создание файла

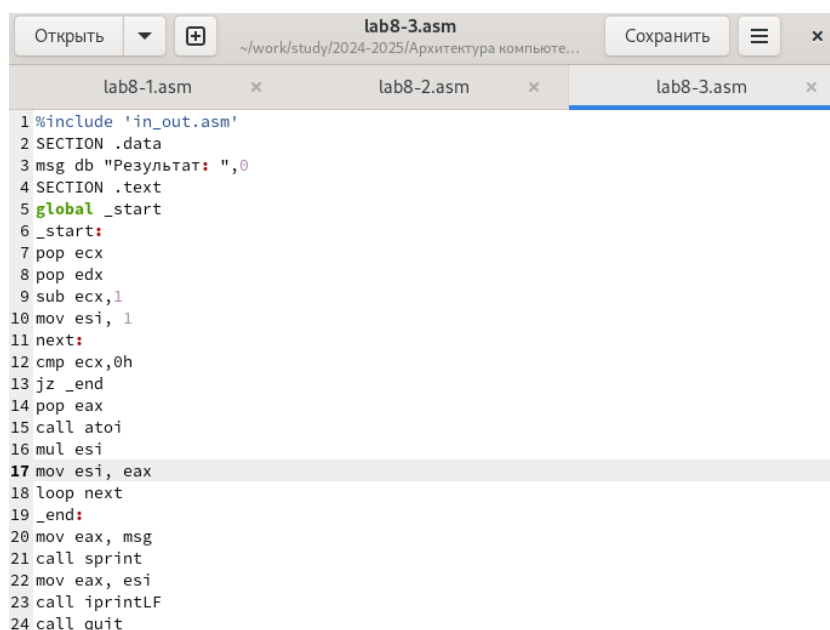
Создаю исполняемый файл и запускаю его, указав аргументы, программа складывает числа(рис. 4.11).



```
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-3.asm
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

Рис. 4.11: Запуск исполняемого файла

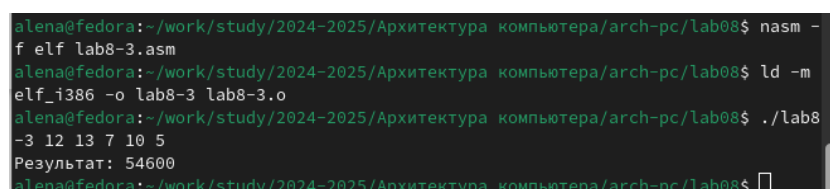
Далее изменяю текст программы для вычисления произведения аргументов(рис. 4.12).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx
8 pop edx
9 sub ecx,1
10 mov esi, 1
11 next:
12 cmp ecx,0h
13 jz _end
14 pop eax
15 call atoi
16 mul esi
17 mov esi, eax
18 loop next
19 _end:
20 mov eax, msg
21 call sprint
22 mov eax, esi
23 call iprintLF
24 call quit
```

Рис. 4.12: Редактирование файла

Создаю исполняемый файл и запускаю его, указав аргументы, программа умножает числа(рис. 4.13).



```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-3.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

Рис. 4.13: Запуск исполняемого файла

4.3 Задание для самостоятельной работы

9 вариант: Создаю файл lab8-4.asm и пишу программу, которая находит сумму значений функции $10^x - 4$ (рис. 4.14).

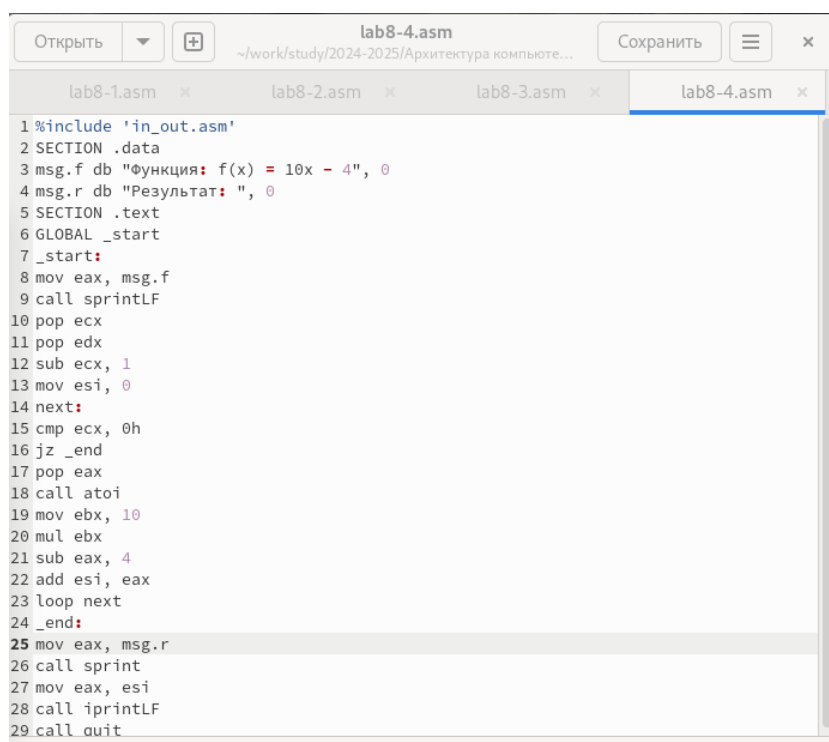


Рис. 4.14: Создание файла

Создаю исполняемый файл и проверяю его работу на нескольких наборах(рис. 4.15).

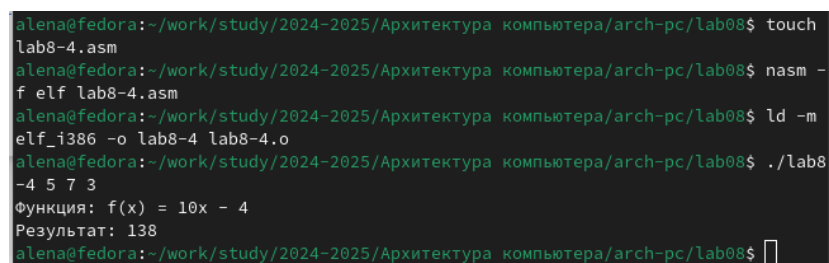


Рис. 4.15: Запуск исполняемого файла

Код программы:

```

#include 'in_out.asm'

SECTION .data
msg.f db "Функция: f(x) = 10x - 4", 0
msg.r db "Результат: ", 0

```

```

SECTION .text
GLOBAL _start
_start:
mov eax, msg.f
call sprintLF
pop ecx
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
mov ebx, 10
mul ebx
sub eax, 4
add esi, eax
loop next
_end:
mov eax, msg.r
call sprint
mov eax, esi
call iprintLF
call quit

```

5 Выводы

При выполнении данной лабораторной работы я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.

Список литературы

1. Архитектура ЭВМ