

# **Отчет по лабораторной работе №8**

**Группа: НКАбд-04-23**  
**Зоригоо Номун**

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
4.1	Реализация циклов в NASM . . . . .	7
4.2	Обработка аргументов командной строки .....	10
4.3	Задания для самостоятельной работы.....	13
<b>5</b>	<b>Выводы</b>	<b>15</b>
	<b>Список литературы</b>	<b>16</b>

## Список иллюстраций

4.1	Создание файлов для лабораторной работы . . . . .	7
4.2	Ввод текста программы из листинга 8.1 . . . . .	7
4.3	Запуск программного кода . . . . .	8
4.4	Изменение текста программы . . . . .	8
4.5	Создание исполняемого файла . . . . .	8
4.6	Изменение текста программы . . . . .	9
4.7	Вывод программы . . . . .	9
4.8	Создание файла . . . . .	9
4.9	Ввод текста программы из листинга 8.3 .....	10
4.10	Проверка работы файла .....	10
4.11	Создание файла листинга .....	10
4.12	Изучение файла листинга .....	11
4.13	Выбранные строки файла .....	11
4.14	Получение файла листинга .....	12
4.15	Написание программы .....	13
4.16	Запуск файла и проверка его работы.....	13
4.17	Написание программы .....	14
4.18	Запуск файла и проверка его работы.....	14

# **1 Цель работы**

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

## 2 Задание

1. Реализация циклов в NASM.
2. Обработка аргументов командной строки.
3. Задания 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.

На рис. 8.1 показана схема организации стека в процессоре. 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).

8.2.1.1. 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.

## 4 Выполнение лабораторной работы

### 4.1 Реализация циклов в NASM

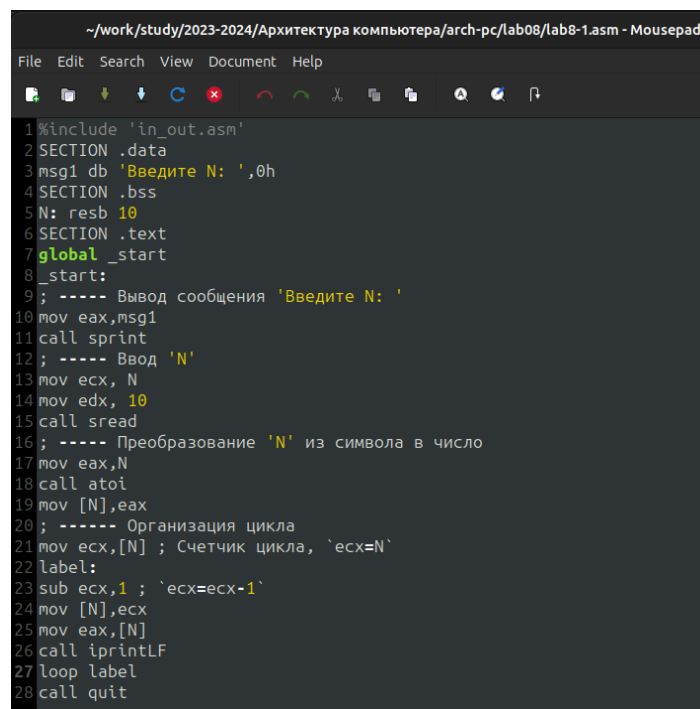
Я создаю каталог для программы лабораторной работы №8 и перехожу в него и создаю файл lab8-1.asm. (рис. 4.1).

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ mkdir lab08
```

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ cd lab08
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ touch lab8-1.asm
```

Рис. 4.1: Создание файлов для лабораторной работы

Я скопировала lab8-1.asm текст программы из листинга 8.1 в Mousepad (рис. 4.2).



```
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08/lab8-1.asm - Mousepad
File Edit Search View Document Help
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 ; `ecx=ecx-1`
24 mov [N],ecx
25 mov eax,[N]
26 call iprintLF
27 loop label
28 call quit
```

Рис. 4.2: Ввод текста программы из листинга 7.1

Создаю исполняемый файл и запускаю его. (рис. 4.3).

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -o lab8-1.o -f elf -g -l list.lst lab8-1.asm
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o main
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
```

Рис. 4.3: Запуск программного кода

Потом я изменила текст программы часть label.(рис 4.4)

```
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08/lab8-1.asm - Mousepad
File Edit Search View Document Help
[Icons]
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 ; `ecx=ecx-1`
24 mov [N],ecx
25 mov eax,[N]
26 call iprintLF
27 loop label
28 call quit
```

Рис. 4.4: Изменение текста программы



Создаю исполняемый файл и проверяю его работу. (рис. 4.5).

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -o lab8-1.o -f elf -g -l list.lst lab8-1.asm
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o main
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
Введите N: 2
1
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
```

Рис. 4.5: Создание исполняемого файла

Затем изменяю текст программы, добавив в конце программы push и pop (рис. 4.1).

```
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08/lab8-1.asm - Mousepad
File Edit Search View Document Help
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 ; добавление значения ecx в стек
24 sub ecx,1
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF
28 pop ecx ; извлечение значения ecx из стека
29 loop label
Filetype: Matlab UTF-8 Line: 22 Column: 1
```

Рис. 4.6: Изменение текста программы

Создаю исполняемый файл и проверяю его работу. Соответствует в данном случае число проходов цикла значению N введенному с клавиатуры. (рис. 4.7)

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -o lab8-1.o -f elf -g -l list.lst lab8-1.asm
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o main
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
1
```

Рис. 4.7: Создание исполняемого файла

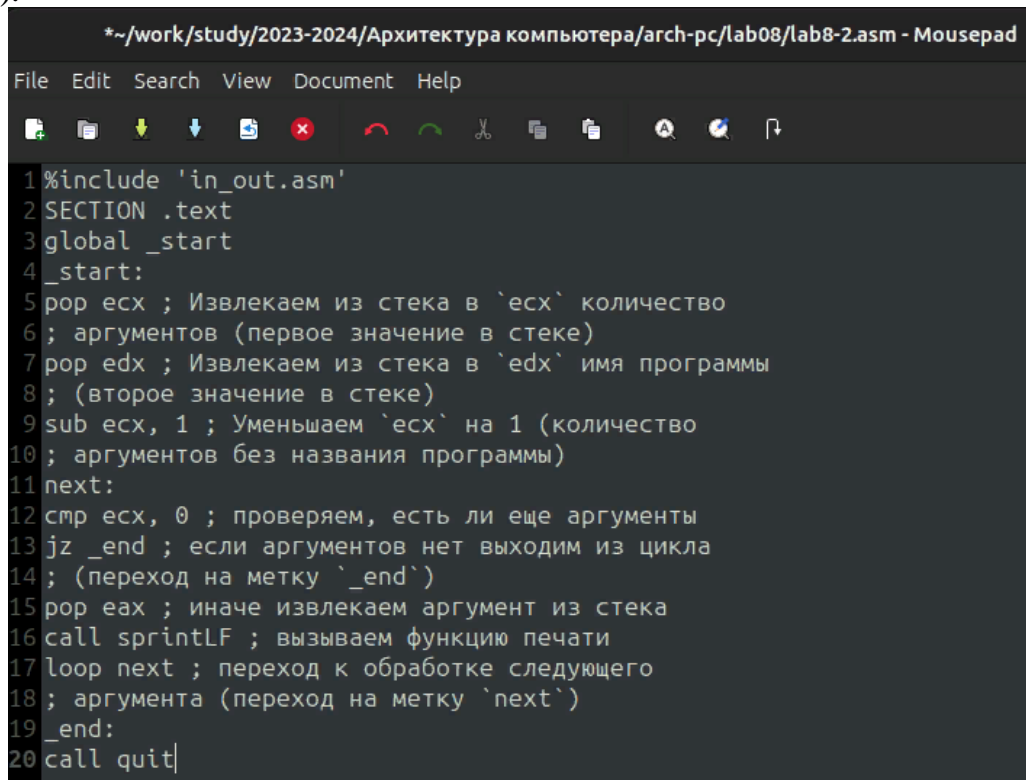
## 4.2 Обработка аргументов командной строки

Создаю файл листинга для программы из файла lab8-2.asm. (рис. 4.8).

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ touch lab8-2.asm
```

Рис. 4.8: Создание файла листинга

Я скопировала lab8-2.asm текст программы из листинга 8.2 в Mousepad (рис. 4.9).



```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08/lab8-2.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2 SECTION .text
3 global _start
4 _start:
5 pop ecx ; Извлекаем из стека в `ecx` количество
6 ; аргументов (первое значение в стеке)
7 pop edx ; Извлекаем из стека в `edx` имя программы
8 ; (второе значение в стеке)
9 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
10 ; аргументов без названия программы)
11 next:
12 cmp ecx, 0 ; проверяем, есть ли еще аргументы
13 jz _end ; если аргументов нет выходим из цикла
14 ; (переход на метку `_end`)
15 pop eax ; иначе извлекаем аргумент из стека
16 call sprintf ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18 ; аргумента (переход на метку `next`)
19 _end:
20 call quit
```

Рис. 4.8: Изучение файла листинга

Создаю исполняемый файл и проверяю его работу.(рис. 4.9)

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-2.asm
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

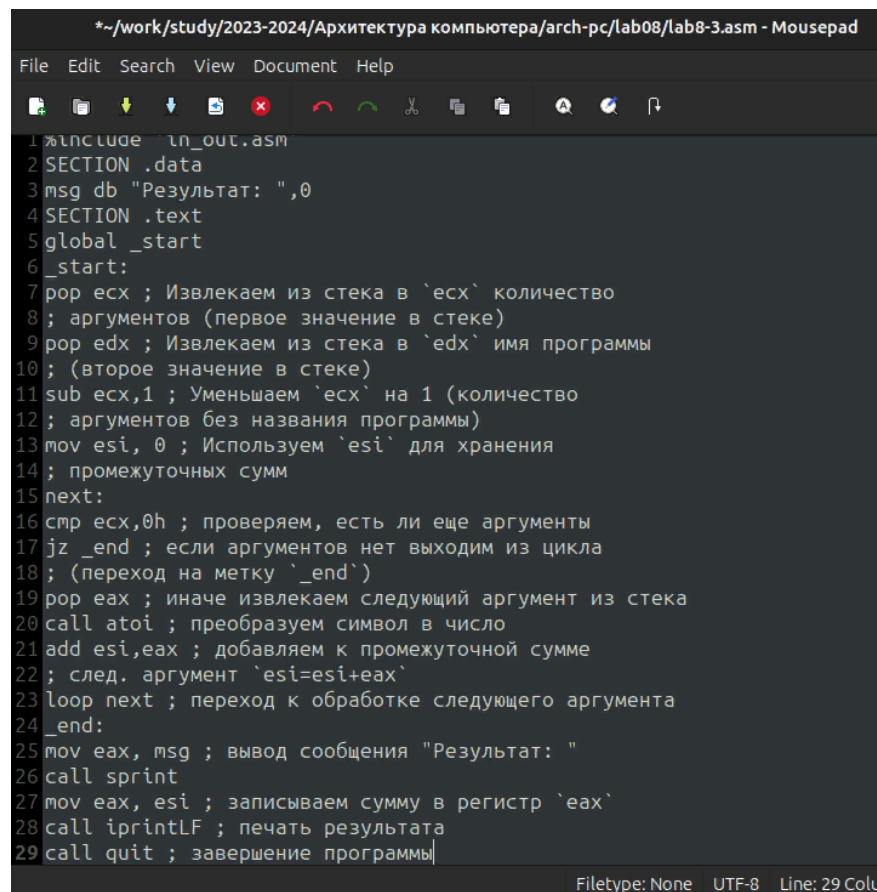
Рис. 4.9: Создание исполняемого файла

Создаю файл листинга для программы из файла lab8-3.asm. (рис. 4.10).

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ touch lab8-3.asm
```

Рис 4.10: Создание исполняемого файла

Я скопировала lab8-2.asm текст программы из листинга 8.2 в Mousepad (рис. 4.11).



```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08/lab8-3.asm - Mousepad
File Edit Search View Document Help
1 %include "in_out.asm"
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 0 ; Используем `esi` для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку `_end`)
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 add esi,eax ; добавляем к промежуточной сумме
22 ; след. аргумент `esi=esi+eax`
23 loop next ; переход к обработке следующего аргумента
24 _end:
25 mov eax, msg ; вывод сообщения "Результат: "
26 call sprint
27 mov eax, esi ; записываем сумму в регистр `eax`
28 call iprintLF ; печать результата
29 call quit ; завершение программы
```

Рис. 4.11: Изучение файла листинга

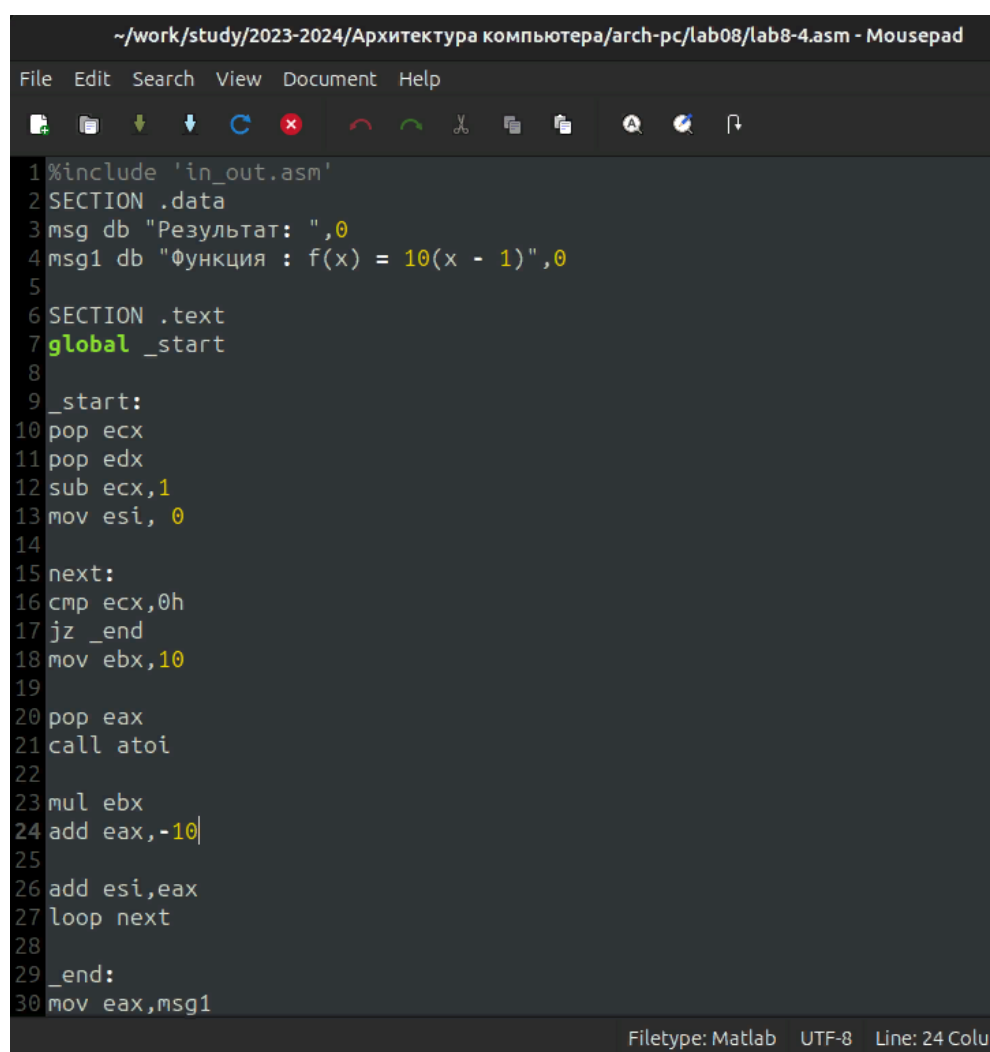
Создаю файл листинга для программы из файла lab8-3.asm. Программ работала без проблема. (рис. 4.12).

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-  
ch-pc/lab08$ nasm -f elf lab8-3.asm  
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-  
ch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o  
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-  
ch-pc/lab08$ ./lab8-3 12 13 7 10 5  
Результат: 47
```

Рис. 4.12: Создание исполняемого файла

## 4.3 Задания для самостоятельной работы

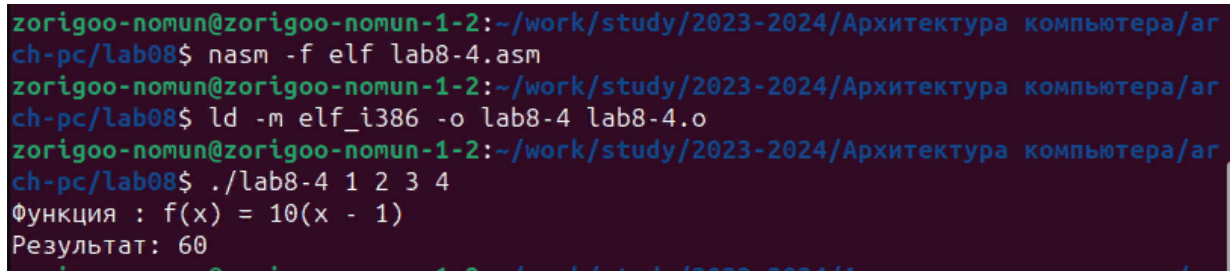
1. Пишу программу, которая находит сумму значений функции  $f(x)$  для  $x = x_1, x_2, \dots, x_n$ , т.е. программа должна выводить значение  $f(x_1) + f(x_2) + \dots + f(x_n)$ . Значения  $x_i$  передаются как аргументы. Вид функции  $f(x)$  выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Мой вариант №17. (Рис 4.13)



```
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08/lab8-4.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 msg1 db "Функция : f(x) = 10(x - 1)",0
5
6 SECTION .text
7 global _start
8
9 _start:
10 pop ecx
11 pop edx
12 sub ecx,1
13 mov esi, 0
14
15 next:
16 cmp ecx,0h
17 jz _end
18 mov ebx,10
19
20 pop eax
21 call atoi
22
23 mul ebx
24 add eax,-10|
25
26 add esi,eax
27 loop next
28
29 _end:
30 mov eax,msg1
Filetype: Matlab UTF-8 Line: 24 Colu
```

Рис. 4.13: Написание программы

Создаю исполняемый файл и проверяю его работу, подставляя необходимые значения.  
Всё хорошо работал. (рис. 4.14).



```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-4.asm
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция : f(x) = 10(x - 1)
Результат: 60
```

Рис. 4.14: Запуск файла и проверка его работы

## **5 Выводы**

Я изучала Реализацию циклов в NASM, Обработка аргументов командной строки.

# Список литературы

1. Архитектура ЭВМ Лабораторная работа №8