

Отчет по лабораторной работе №8

Группа: НКАбд-04-23

Монхжаргал

Тувшинбаяр

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	Задания для самостоятельной работы	13
5	Выводы	15
	Список литературы	16

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:

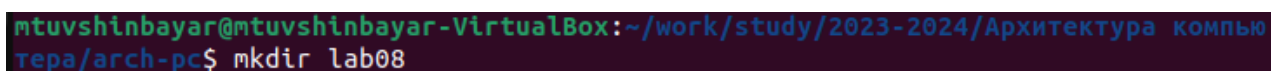
- добавление элемента в вершину стека (push);
- извлечение элемента из вершины стека (pop).

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

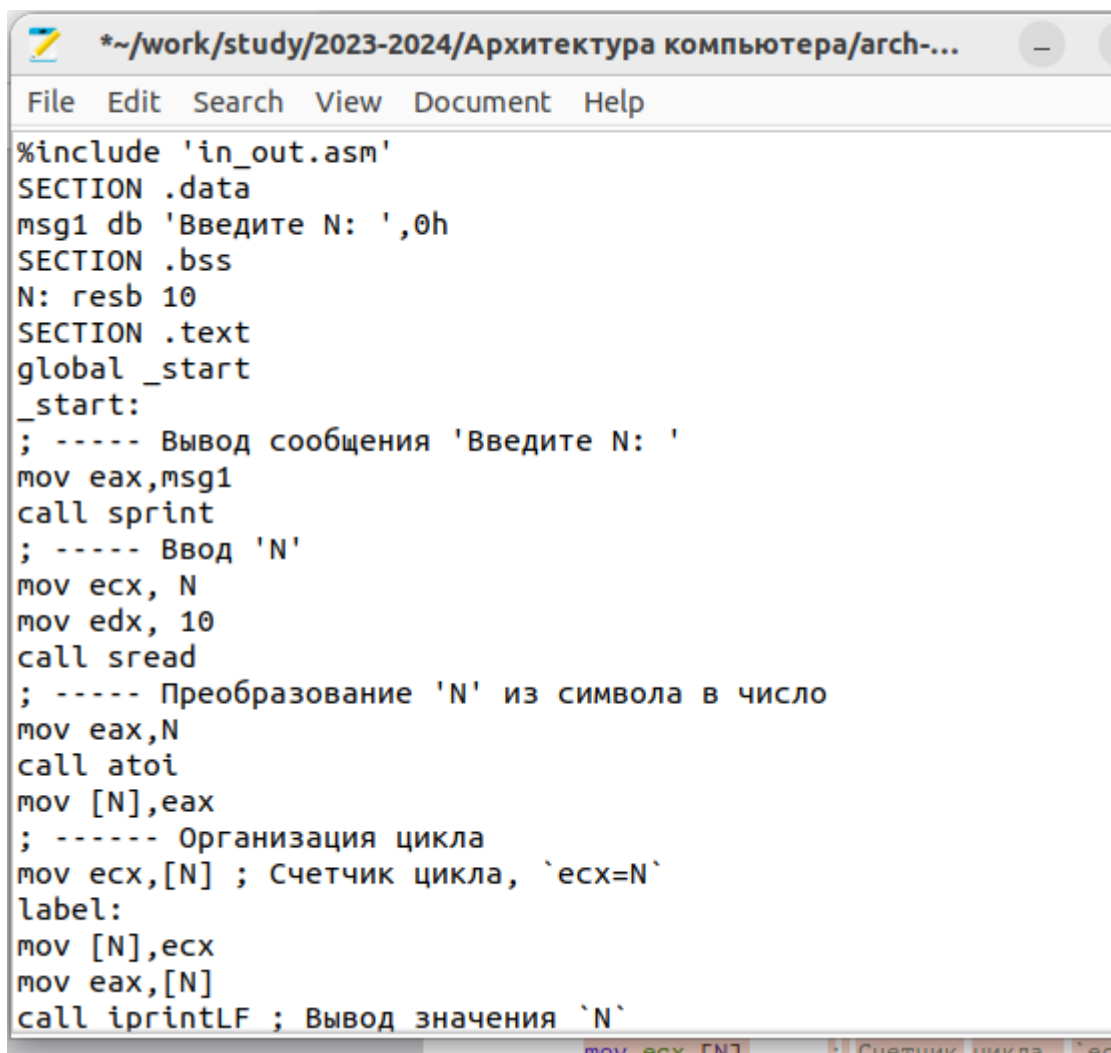
Я создаю каталог для программы лабораторной работы №8 и перехожу в него и создаю файл lab8-1.asm. (рис. 4.1).

A screenshot of a terminal window with a dark background. The prompt is 'mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компью' and the command entered is 'тера/arch-pc\$ mkdir lab08'.

```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компью  
тера/arch-pc$ mkdir lab08
```

Рис. 4.1: Создание файлов для лабораторной работы

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



The screenshot shows a text editor window with the title bar: `*~/work/study/2023-2024/Архитектура компьютера/arch-...`. The menu bar includes `File`, `Edit`, `Search`, `View`, `Document`, and `Help`. The code is written in assembly and includes comments in Russian. It defines a data section with a message, a bss section for a variable `N`, and a text section for the main logic. The logic involves printing the message, reading a string, converting it to an integer, and then printing the integer value in a loop.

```
%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:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
mov ecx,[N] ; Счетчик цикла, `ecx=N`
```


Рис. 4.2: Ввод текста программы из листинга 7.1

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

```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компь  
тера/arch-pc/lab08$ nasm -o lab8-1.o -f elf -g -l list.lst lab8-1.asm  
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компь  
тера/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1  
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компь  
тера/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o main  
ld: unrecognized emulation mode: elf  
Supported emulations: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu elf_l10m elf_k10m  
i386pep i386pe  
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компь  
тера/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)

```
%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]
```

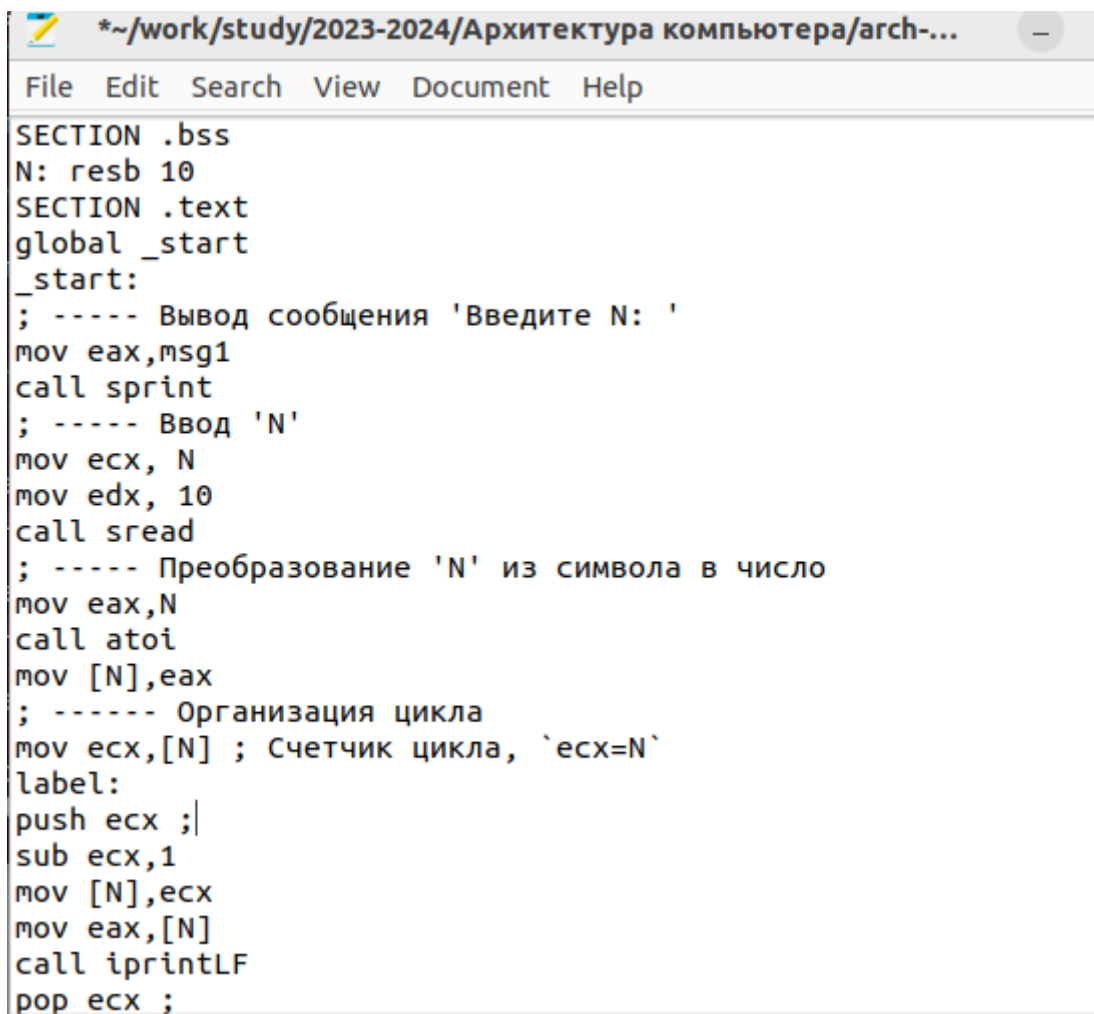
Рис. 4.4: Изменение текста программы

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

```
mtuvshinbayar@mtuvshinbayar-VirtualBox:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/lab08$ nasm -o lab8-1.o -f elf -g -l list.lst lab8-1.asm
mtuvshinbayar@mtuvshinbayar-VirtualBox:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o lab8-1
mtuvshinbayar@mtuvshinbayar-VirtualBox:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o main
ld: unrecognized emulation mode: elf
Supported emulations: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu elf_l1om elf_k1om
i386pep i386pe
mtuvshinbayar@mtuvshinbayar-VirtualBox:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/lab08$ ./lab8-1
Введите N: 2
1
mtuvshinbayar@mtuvshinbayar-VirtualBox:~/work/study/2023-2024/Архитектура компьютер
a/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).



The image shows a screenshot of a text editor window. The title bar at the top reads: `*~/work/study/2023-2024/Архитектура компьютера/arch-...`. Below the title bar is a menu bar with the following items: `File`, `Edit`, `Search`, `View`, `Document`, and `Help`. The main area of the window contains assembly code. The code starts with `SECTION .bss` and `N: resb 10`. It then moves to `SECTION .text` and declares `global _start`. The `_start:` label is followed by several instructions: `; ----- 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``, and a `label:` block containing `push ecx ;|`, `sub ecx,1`, `mov [N],ecx`, `mov eax,[N]`, `call iprintLF`, and `pop ecx ;`.

```
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 ;|
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ;
```

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)

```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ nasm -o lab8-1.o -f elf -g -l list.lst lab8-1.asm
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 lab8-1.o -o main
ld: unrecognized emulation mode: elf
Supported emulations: elf_x86_64 elf32_x86_64 elf_i386 elf_iarmcu elf_l10m elf_k10m i386pep i386pe
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
2
1
0
```

Рис. 4.7: Создание исполняемого файла

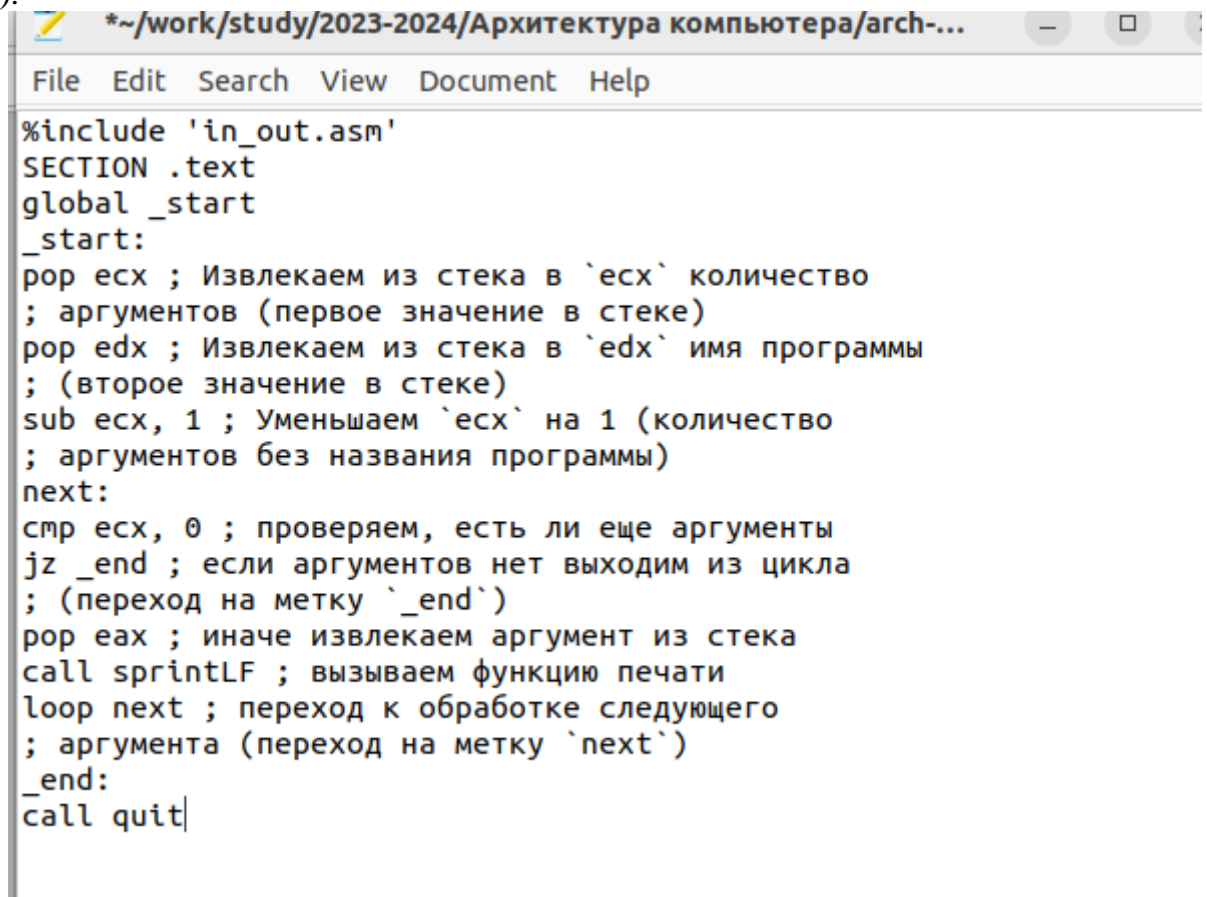
4.2 Обработка аргументов командной строки

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

```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/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-...
File Edit Search View Document Help
%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 sprintLF ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

Рис. 4.8: Изучение файла листинга

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

```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/lab08$ nasm -f elf lab8-2.asm
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

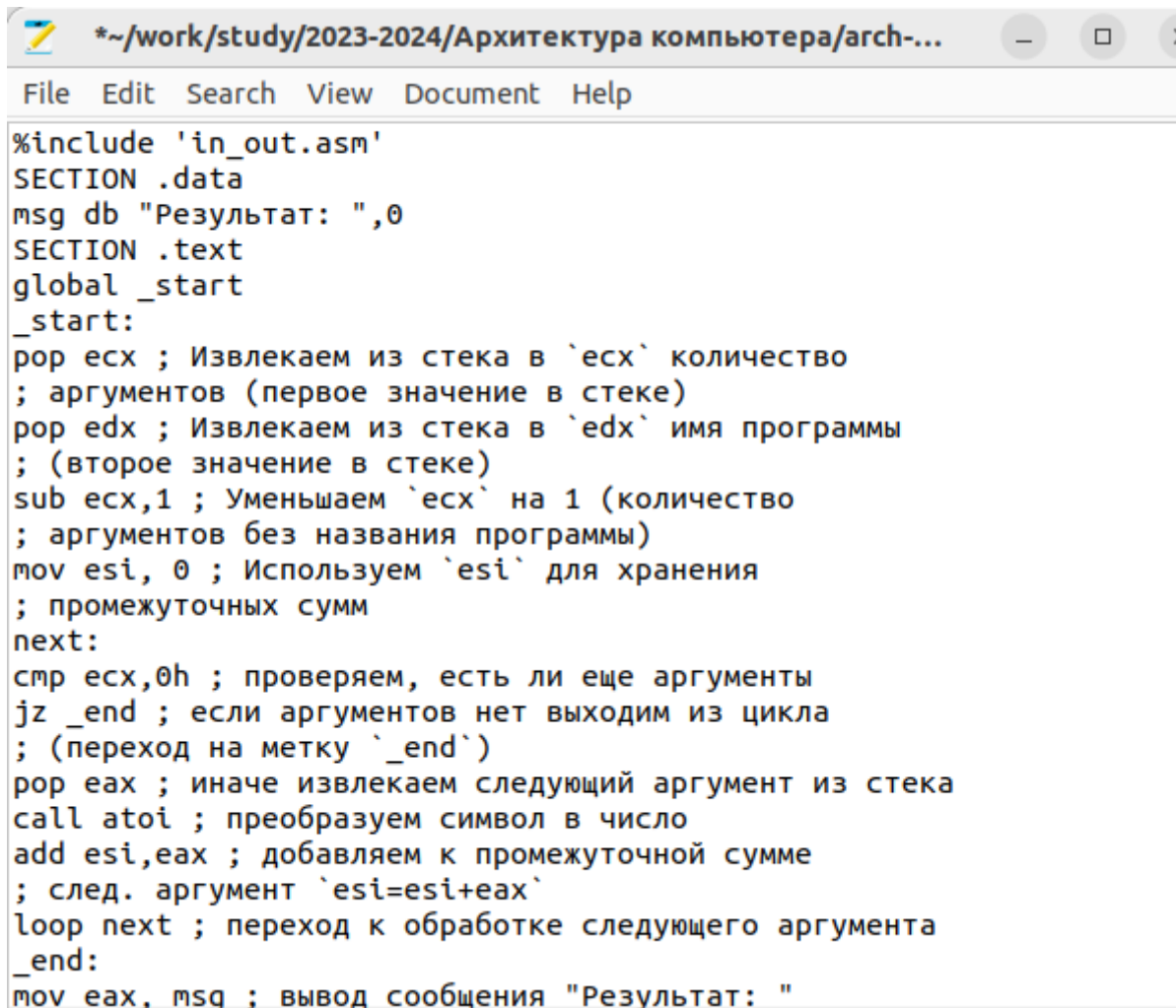
Рис. 4.9: Создание исполняемого файла

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

```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/lab08$ touch lab8-3.asm
```

Рис 4.10: Создание исполняемого файла

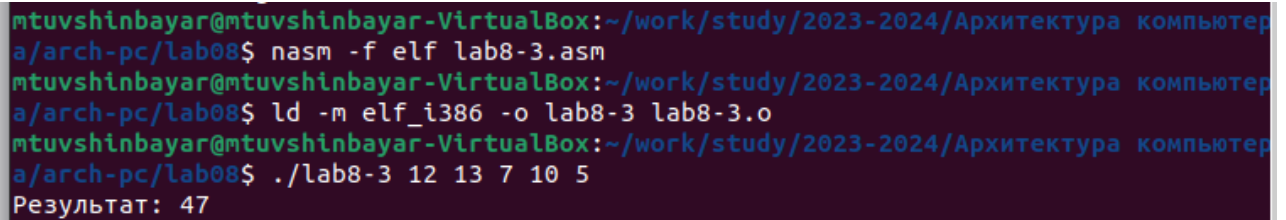
Я скопировала lab8-2.asm текст программы из листинга 8.2 в Mousepad (рис. 4.11).



```
%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: "
```

Рис. 4.11: Изучение файла листинга

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

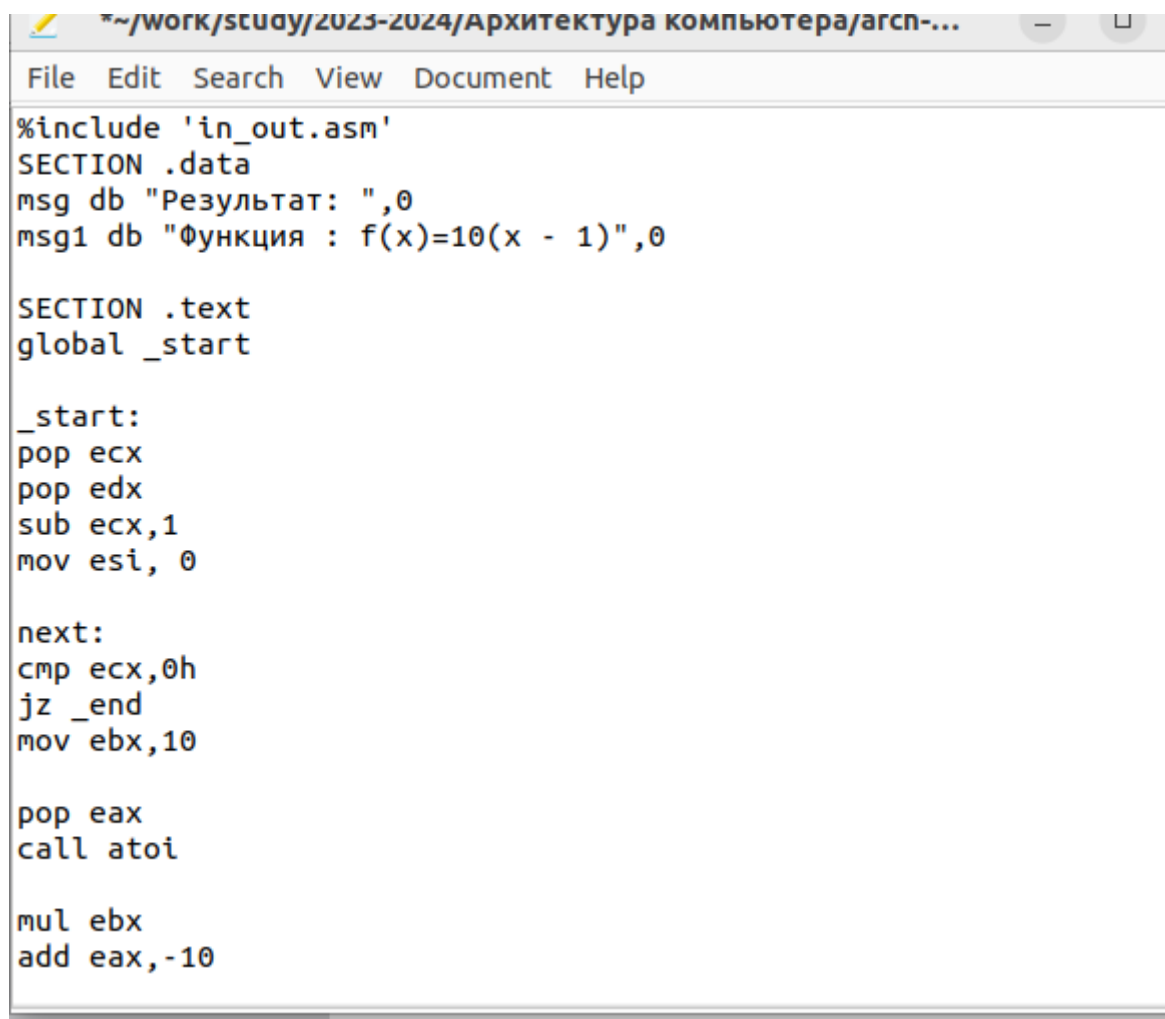


```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/lab08$ nasm -f elf lab8-3.asm
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютер
a/arch-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)



```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msg1 db "Функция : f(x)=10(x - 1)",0

SECTION .text
global _start

_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
mov ebx,10

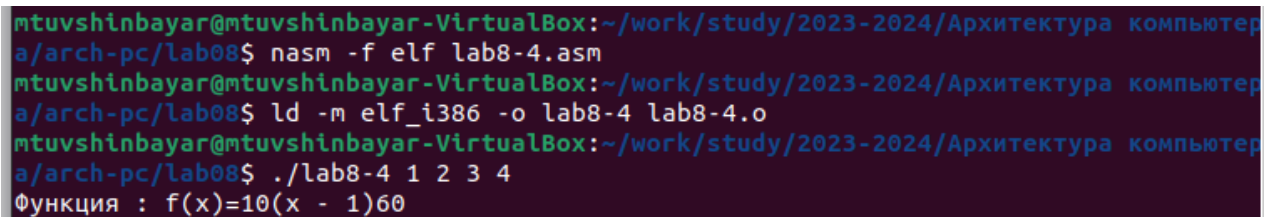
pop eax
call atoi

mul ebx
add eax,-10
```

Рис. 4.13: Написание программы

Создаю исполняемый файл и проверяю его работу, подставляя необходимые значения.

Всё хорошо работал. (рис. 4.14).



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

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