

Отчёт по лабораторной работе №8

Дисциплина: архитектура компьютеров

Терещенкова Маргарита Владимировна

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	12
4.3	Самостоятельная работа	18
5	Выводы	22
	Список литературы	23

Список иллюстраций

4.1	Создание каталога	7
4.2	Копирование файла	7
4.3	Редактирование файла	8
4.4	Запуск файла	8
4.5	Редактирование файла	9
4.6	Запуск файла	10
4.7	Редактирование файла	11
4.8	Запуск файла	12
4.9	Создание файла	12
4.10	Редактирование файла	13
4.11	Запуск исполняемого файла	13
4.12	Создание файла	14
4.13	Редактирование файла	15
4.14	Запуск исполняемого файла	16
4.15	Редактирование файла	17
4.16	Запуск исполняемого файла	18
4.17	Создание файла	18
4.18	Редактирование файла	19
4.19	Запуск исполняемого файла	20

1 Цель работы

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

2 Задание

1. Реализация циклов 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.

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

Создала каталог для программ лабораторной работы № 8, перешла в него и создала файл lab8-1.asm:

```
mvtereshenkova@margo-pc:~$ mkdir ~/work/study/2024-2025/Архитектура\ компьютера  
/arch-pc/lab08  
mvtereshenkova@margo-pc:~$ cd ~/work/study/2024-2025/Архитектура\ компьютера/arch-  
pc/lab08/  
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/l  
ab08$ touch lab8-1.asm
```

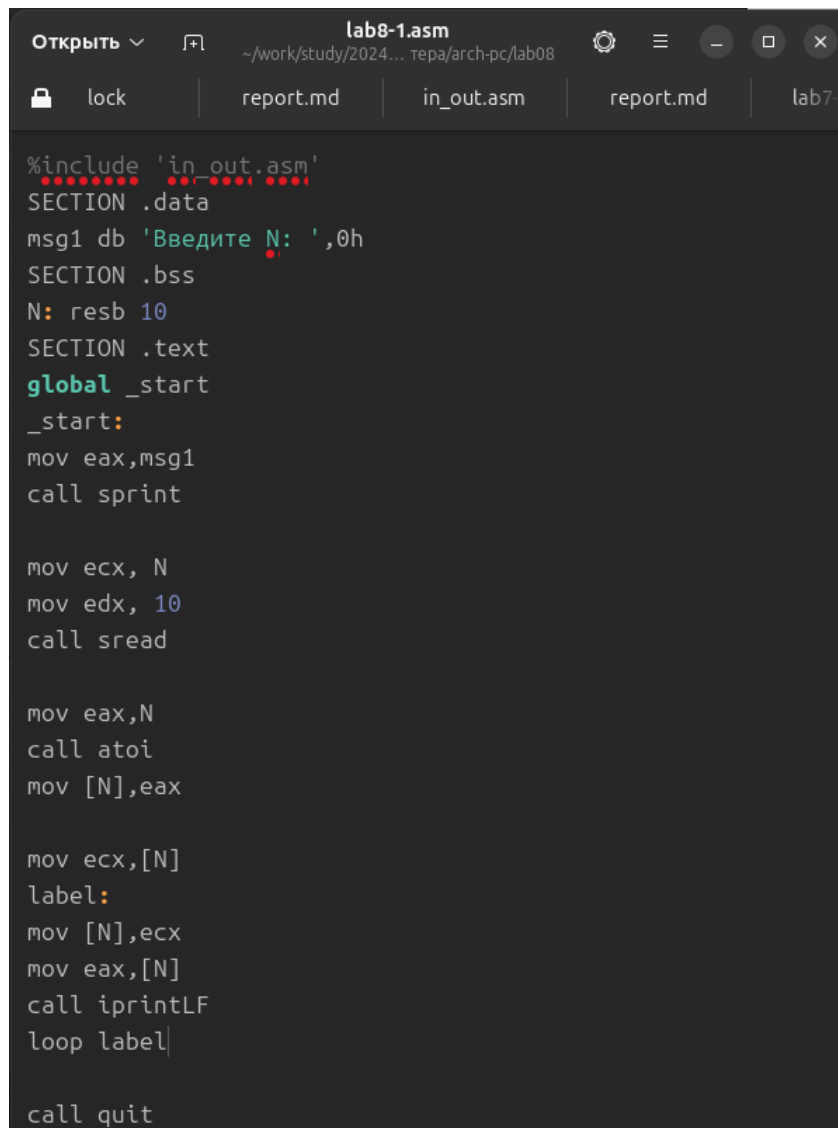
Рис. 4.1: Создание каталога

Копирую в текущий каталог файл in_out.asm с помощью утилиты cp, так как он будет использоваться в других программах. И проверяю наличие файла в данной директории с помощью команды ls.

```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ cp ~/Загрузки/in_out.asm in_out.asm  
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ls  
in_out.asm lab8-1.asm
```

Рис. 4.2: Копирование файла

Ввожу в файл lab8-1.asm текст программы из листинга 8.1.



```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
mov eax,msg1
call sprint

mov ecx, N
mov edx, 10
call sread

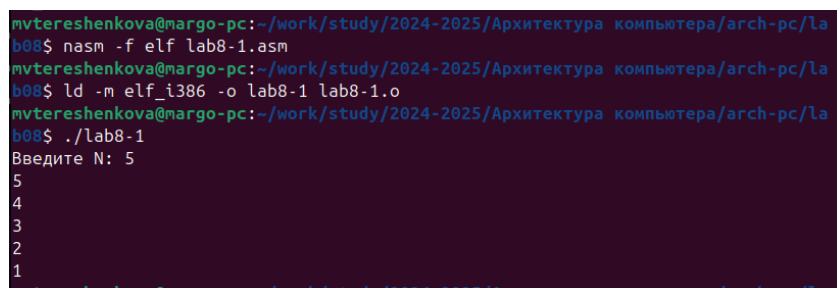
mov eax,N
call atoi
mov [N],eax

mov ecx,[N]
label:
mov [N],ecx
mov eax,[N]
call iprintLF
loop label

call quit
```

Рис. 4.3: Редактирование файла

Создаю исполняемый файл и запускаю его.

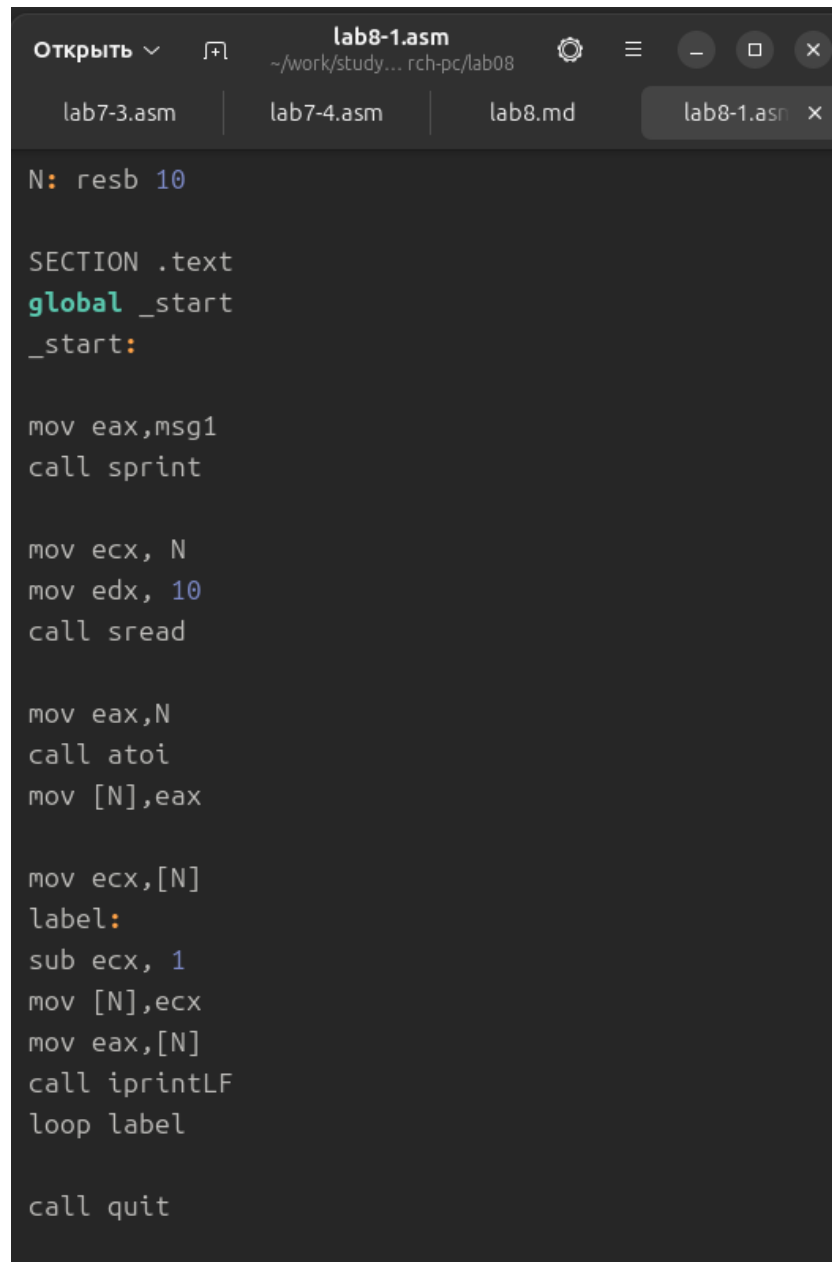


```
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ nasm -f elf lab8-1.asm
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ./lab8-1
Введите N: 5
5
4
3
2
1
```

Рис. 4.4: Запуск файла

Исполняемый файл работает корректно.

Меняю текст программы добавив изменение значение регистра ecx в цикле.



```
Открыть ▾ [icon] lab8-1.asm
~/work/study... rch-pc/lab08

lab7-3.asm | lab7-4.asm | lab8.md | lab8-1.asm x

N: resb 10

SECTION .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx, N
mov edx, 10
call sread

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

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

call quit
```

Рис. 4.5: Редактирование файла

Создаю исполняемый файл и запускаю его.

```

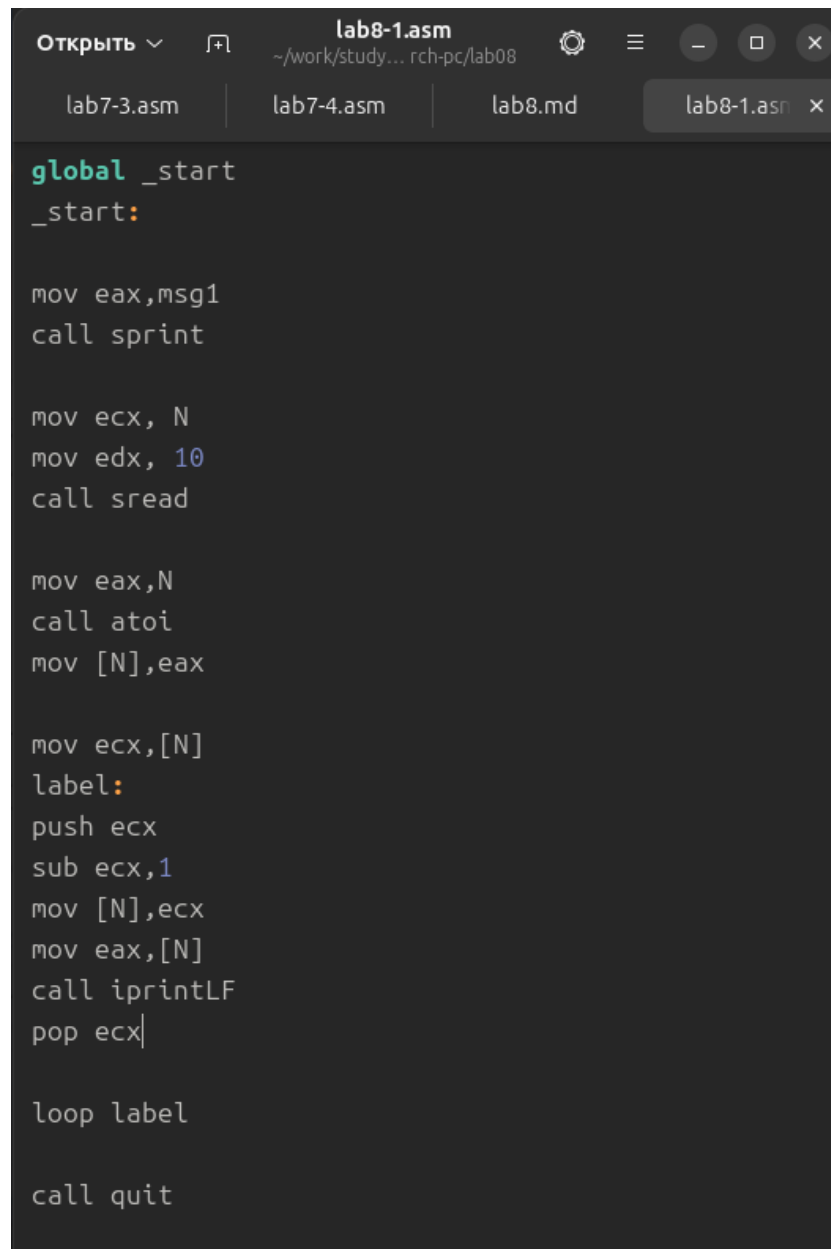
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ nasm -f elf lab8-1.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ./lab8-1
Введите N: 4
3
1

```

Рис. 4.6: Запуск файла

Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Число проходов не соответствует значению `N`.

Вношу изменения в текст программы, добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`.



```
global _start
_start:

mov eax,msg1
call sprint

mov ecx, N
mov edx, 10
call sread

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

mov ecx,[N]
label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx

loop label

call quit
```

Рис. 4.7: Редактирование файла

Создаю исполняемый файл и запускаю его.

```

mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ nasm -f elf lab8-1.asm
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ld -m elf_i386 -o lab8-1 lab8-1.o
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0

```

Рис. 4.8: Запуск файла

В данном случае число проходов цикла соответствует значению N.

4.2 Обработка аргументов командной строки

Создаю файл с названием lab8-2.asm.

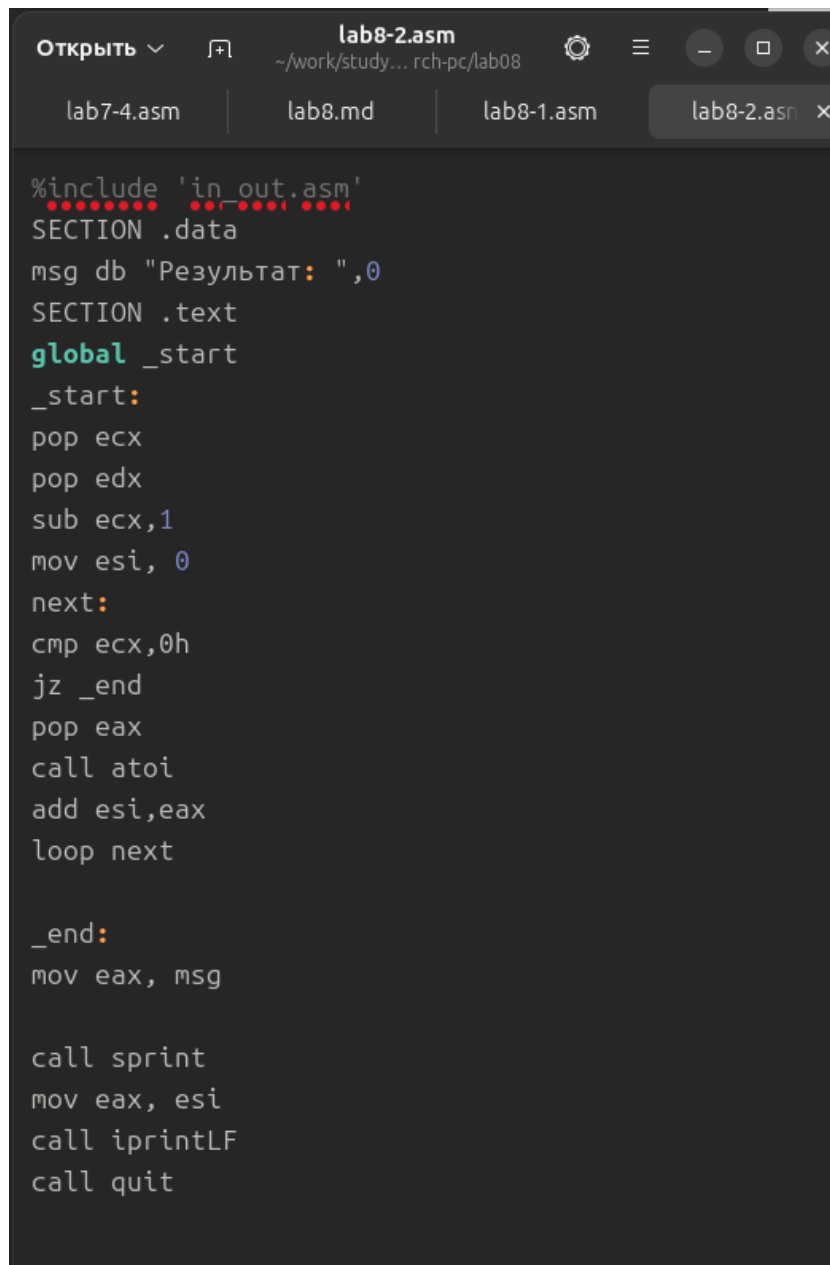
```

mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ touch lab8-2.asm
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ls
in_out.asm lab8-1 lab8-1.asm lab8-1.o lab8-2.asm

```

Рис. 4.9: Создание файла

Ввожу в него текст программы из листинга 8.2



```
lab8-2.asm
~/work/study...rch-pc/lab08

lab7-4.asm | lab8.md | lab8-1.asm | lab8-2.asm x

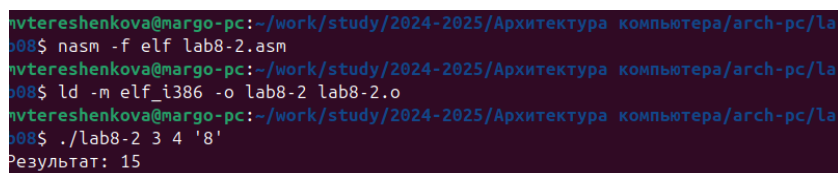
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0
next:
cmp ecx,0h
jz _end
pop eax
call atoi
add esi,eax
loop next

_end:
mov eax, msg

call sprint
mov eax, esi
call iprintLF
call quit
```

Рис. 4.10: Редактирование файла

Создаю исполняемый файл и запускаю его, указав аргументы.



```
mtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
008$ nasm -f elf lab8-2.asm
mtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
008$ ld -m elf_i386 -o lab8-2 lab8-2.o
mtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
008$ ./lab8-2 3 4 '8'
Результат: 15
```

Рис. 4.11: Запуск исполняемого файла

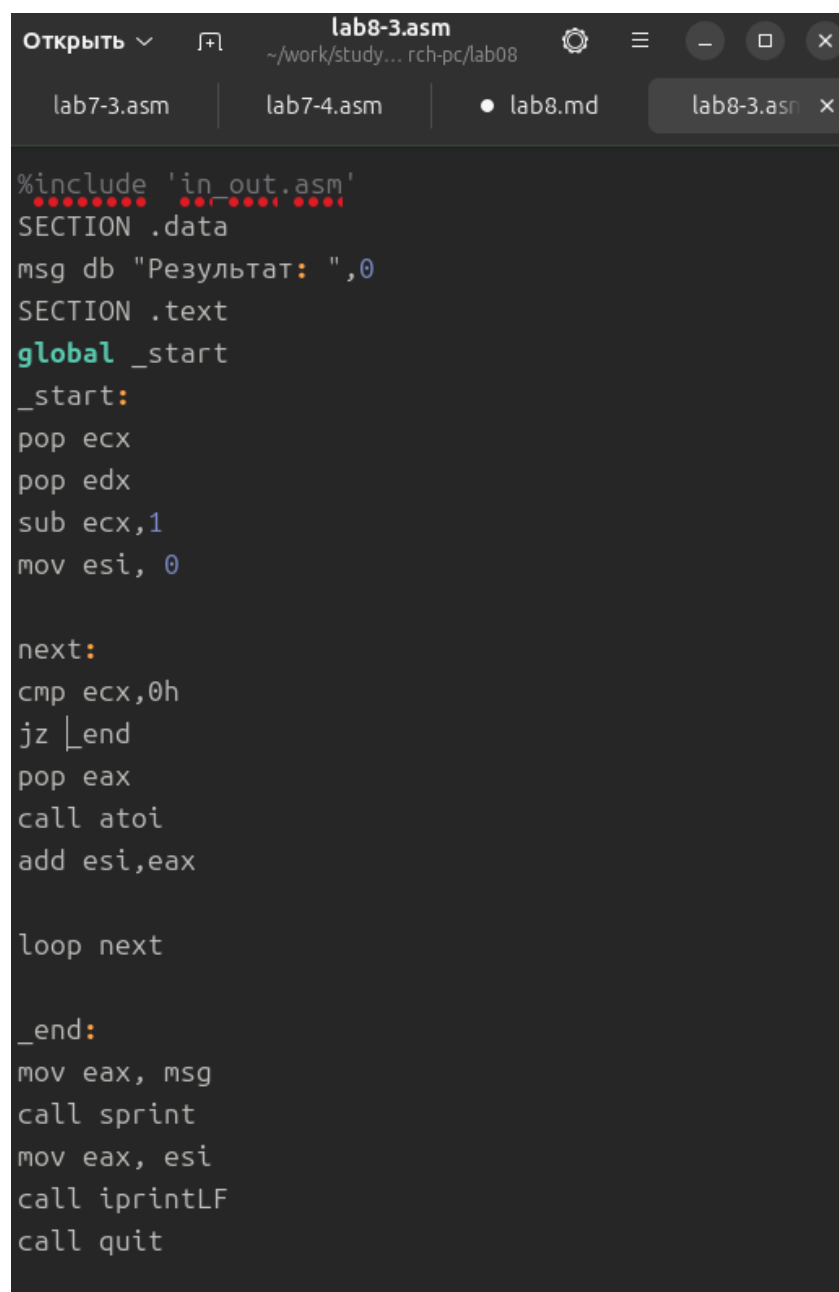
Программа обработала 3 аргумента.

Создаю файл с названием lab8-3.asm.

```
mvtreshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ touch lab8-3.asm
mvtreshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ ls
in_out.asm  lab8-1.asm  lab8-2      lab8-2.o
lab8-1      lab8-1.o    lab8-2.asm  lab8-3.asm
```

Рис. 4.12: Создание файла

Ввожу в него текст программы из листинга 8.3



The image shows a text editor window with a dark theme. The title bar at the top reads "lab8-3.asm" and the path is "~/work/study... rch-pc/lab08". Below the title bar, there are tabs for "lab7-3.asm", "lab7-4.asm", "lab8.md", and "lab8-3.asm". The main editing area contains the following assembly code:

```
%include 'in_out.asm'
SECTION .data
msg 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
add esi,eax

loop next

_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

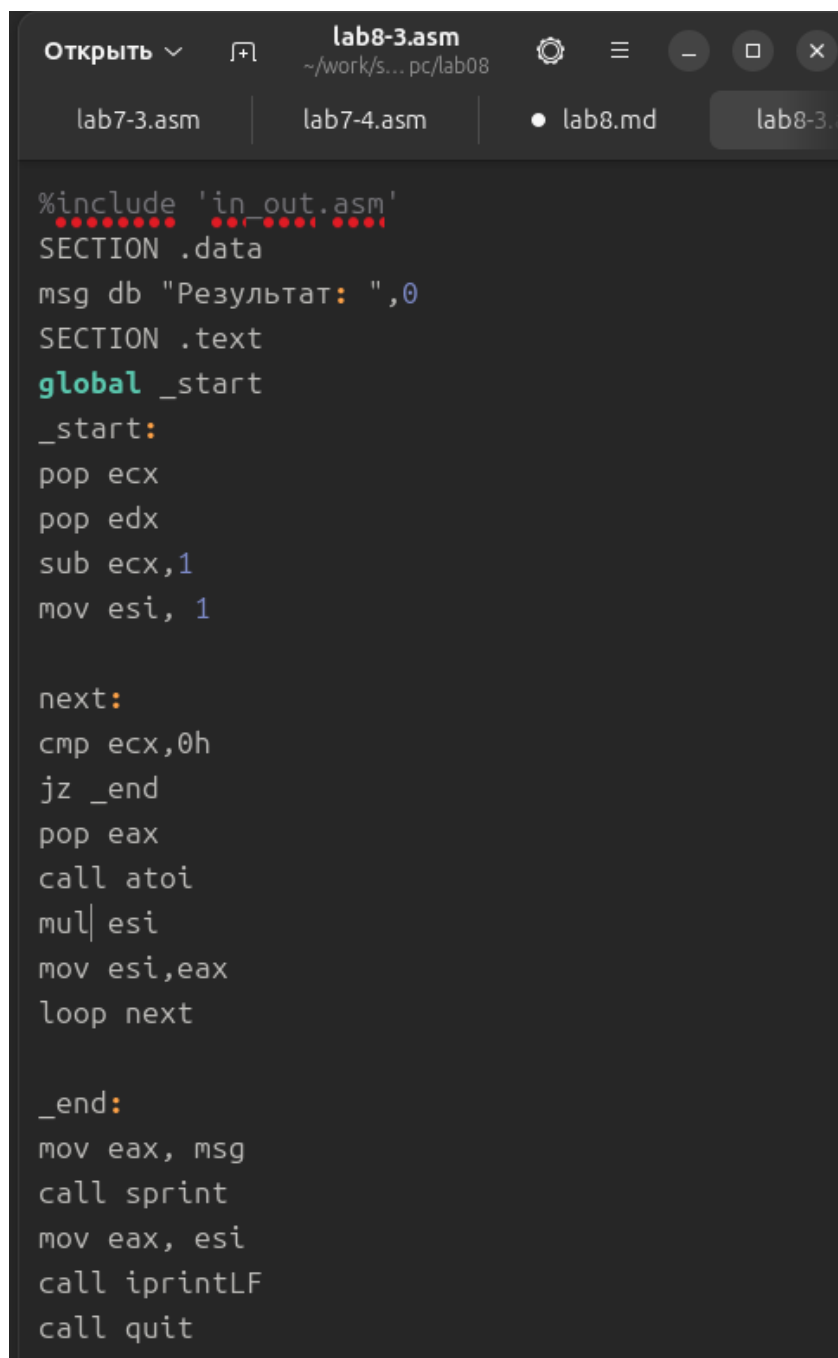
Рис. 4.13: Редактирование файла

Создаю исполняемый файл и запускаю его, указав аргументы.

```
mvtereshenkova@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ nasm -f elf lab8-3.asm
mvtereshenkova@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ld -m elf_i386 -o lab8-3 lab8-3.o
mvtereshenkova@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ./lab8-3 2 3 4 5
Результат: 14
```

Рис. 4.14: Запуск исполняемого файла

Редактирую текст программы для вычисления произведения аргументов командной строки.



```
Открыть ▾  lab8-3.asm  ~\work\s...pc\lab08  [Icons]
lab7-3.asm | lab7-4.asm | ● lab8.md | lab8-3.asm

%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 1

next:
cmp ecx,0h
jz _end
pop eax
call atoi
mul esi
mov esi,eax
loop next

_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

Рис. 4.15: Редактирование файла

Создаю исполняемый файл и запускаю его, указав аргументы.

```

mvtereshenkova@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ nasm -f elf lab8-3.asm
mvtereshenkova@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ld -m elf_i386 -o lab8-3 lab8-3.o
mvtereshenkova@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ./lab8-3 2 3 4 5
Результат: 120

```

Рис. 4.16: Запуск исполняемого файла

4.3 Самостоятельная работа

Создаю файл lab8-4.asm.

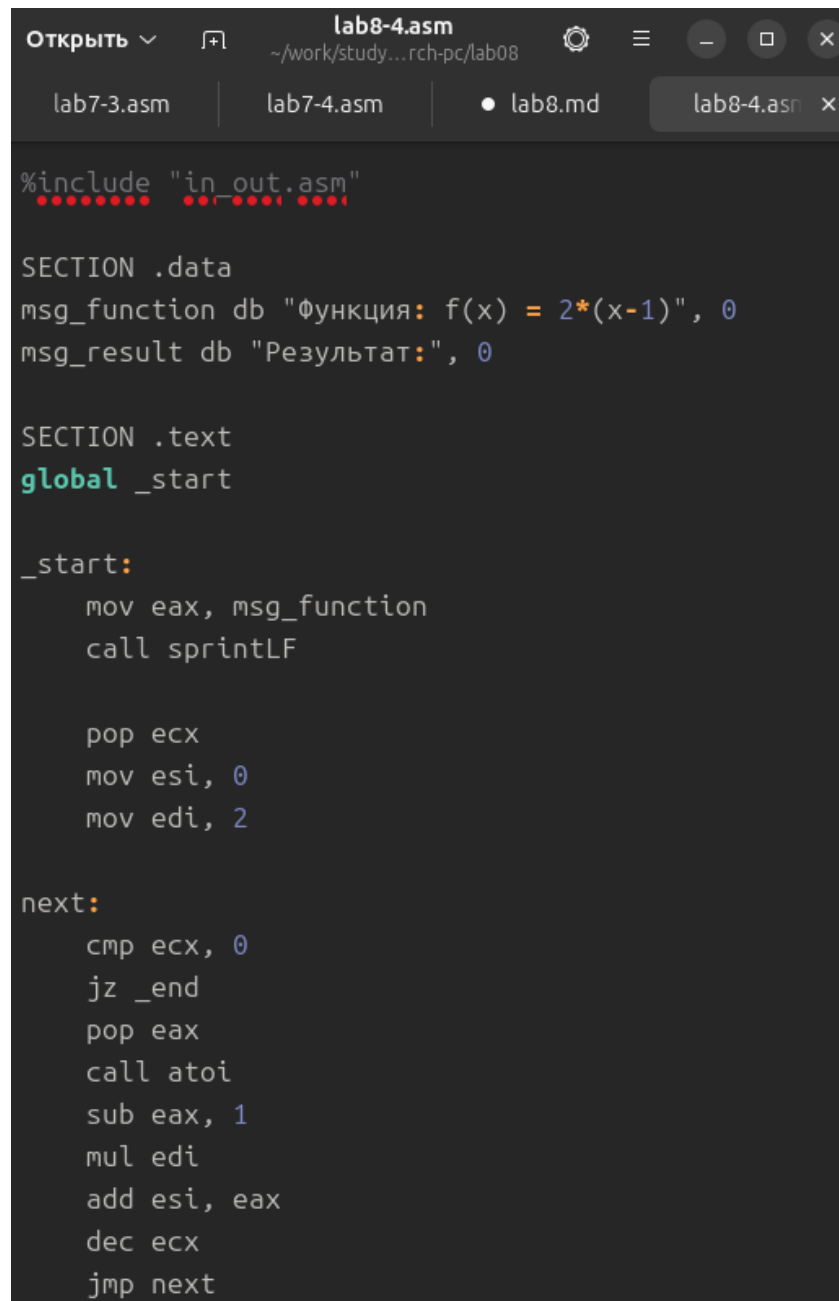
```

mvtereshenkova@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ touch lab8-4.asm
mvtereshenkova@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ls
in_out.asm  lab8-1.asm  lab8-2      lab8-2.o  lab8-3.asm  lab8-4.asm
lab8-1      lab8-1.o   lab8-2.asm  lab8-3    lab8-3.o

```

Рис. 4.17: Создание файла

Начинаю написание программы, которая будет вычислять сумму значений $f(x)=2(x-1)$. (вариант 4)



```
Открыть ▾  lab8-4.asm  ~\work\study...rch-pc\lab08  lab7-3.asm  lab7-4.asm  ● lab8.md  lab8-4.asm x

%include "in_out.asm"

SECTION .data
msg_function db "Функция: f(x) = 2*(x-1)", 0
msg_result db "Результат:", 0

SECTION .text
global _start

_start:
    mov eax, msg_function
    call sprintf

    pop ecx
    mov esi, 0
    mov edi, 2

next:
    cmp ecx, 0
    jz _end
    pop eax
    call atoi
    sub eax, 1
    mul edi
    add esi, eax
    dec ecx
    jmp next
```

Рис. 4.18: Редактирование файла

Создаю исполняемый файл и запускаю его.

```

mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ nasm -f elf lab8-4.asm
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ld -m elf_i386 -o lab8-4 lab8-4.o
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ./lab8-4 1 2 3 4
Функция:  $f(x) = 2*(x-1)$ 
Результат:10
mytereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/la
b08$ ./lab8-4 2 5 6 7 8
Функция:  $f(x) = 2*(x-1)$ 
Результат:44

```

Рис. 4.19: Запуск исполняемого файла

Произведя несложные математические вычисления, делаю вывод, что программа работает верно.

Текст программы:

```

#include "in_out.asm"

SECTION .data
msg_function db "Функция:  $f(x) = 2*(x-1)$ ", 0
msg_result db "Результат:", 0

SECTION .text
global _start
_start:
mov eax, msg_function
call sprintLF
pop ecx
mov esi, 0
mov edi, 2
next:
cmp ecx, 0
jz _end
pop eax
call atoi
sub eax, 1
mul edi

```

```
add esi, eax
dec ecx
jmp next
_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit
```

5 Выводы

Благодаря данной лабораторной работе, приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.

Список литературы

1. Архитектура компьютеров