

Отчет по лабораторной работе №8

Дисциплина: Архитектура компьютера

Юсуфов Джабар Артикович

Содержание

1	Цель работы	1
2	Задание	1
3	Выполнение лабораторной работы	1
3.1	Реализация циклов в NASM.	1
3.2	Обработка аргументов командной строки	6
3.3	Задание для самостоятельной работы.....	9
4	Выводы	10

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Задание

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

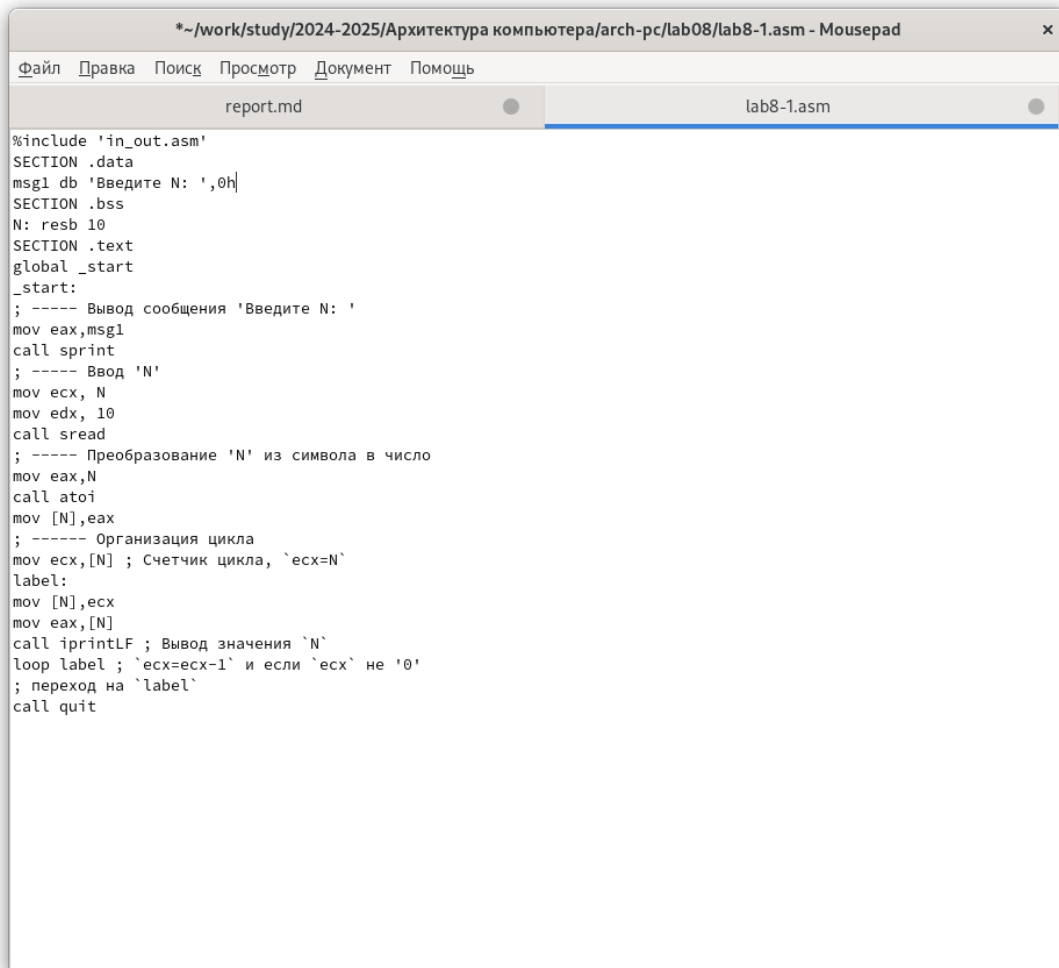
3 Выполнение лабораторной работы

3.1 Реализация циклов в NASM.

Создаю каталог для программ лабораторной работы №8.

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

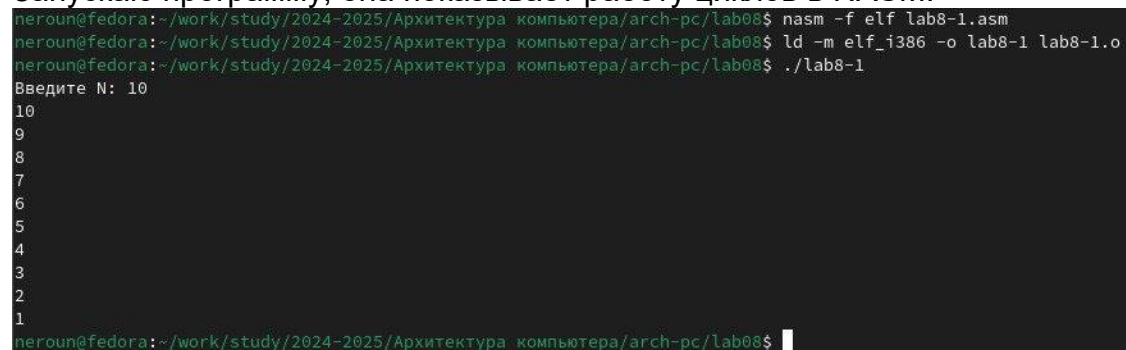
Копирую в созданный файл программу из листинга.



```
*~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08/lab8-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
report.md  lab8-1.asm

%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`
loop label ; `ecx=ecx-1` и если `ecx` не `0`
; переход на `label`
call quit
```

Запускаю программу, она показывает работу циклов в NASM.



```
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-1.asm
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра ecx.



```
%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
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не `0`
; переход на `label`
call quit
```

Из-за того, что теперь регистр `ecx` на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое.

```
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-1.asm
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

Добавляю команды push и pop в программу.



```
Открыть + lab8-1.asm ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08

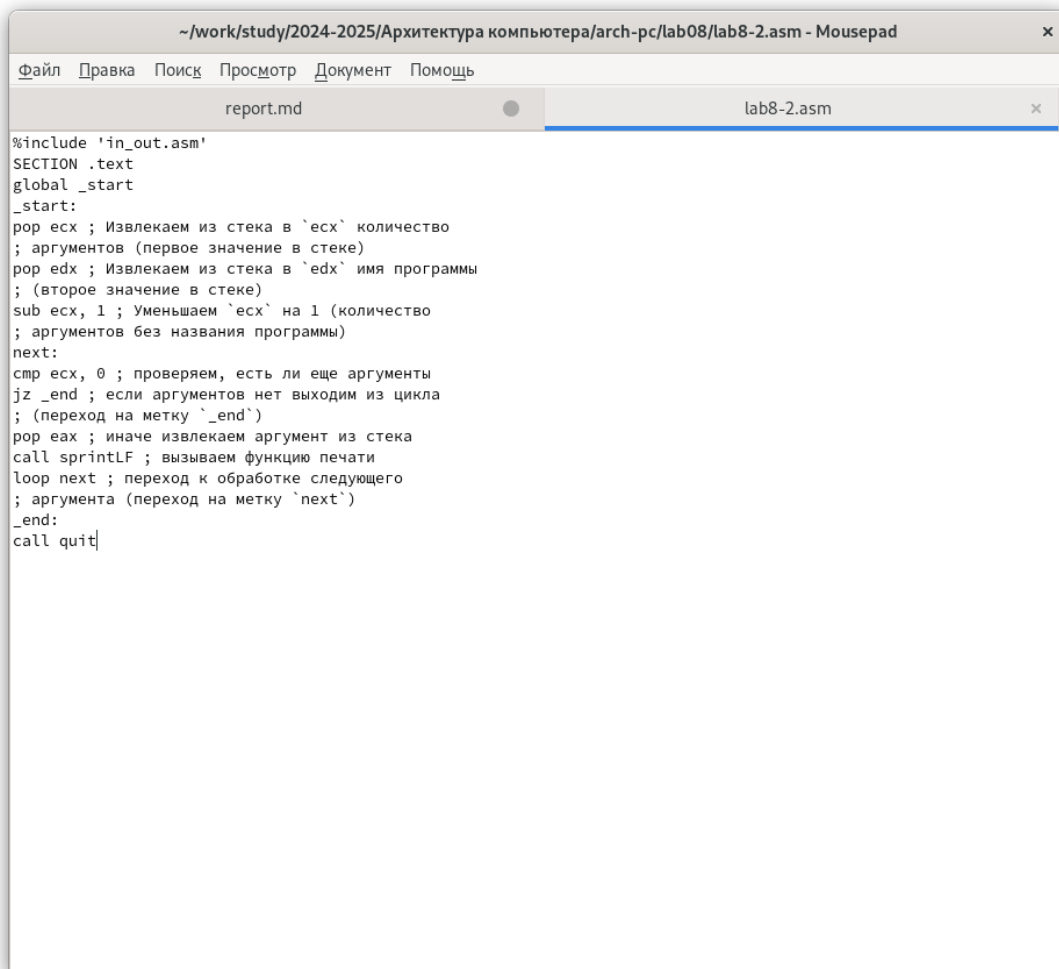
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
pop ecx
loop label ; `ecx=ecx-1` и если `ecx` не `0`
; переход на `label`
call quit
```

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

```
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-1.asm
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

3.2 Обработка аргументов командной строки

Создаю новый файл для программы и копирую в него код из следующего листинга.



```
~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08/lab8-2.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
report.md  lab8-2.asm
%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 sprintf ; вызываем функцию печати
    loop next ; переход к обработке следующего
             ; аргумента (переход на метку `next`)
_end:
    call quit
```

Компилирую программу и запускаю, указав аргументы. Программой было обработано то же количество аргументов, что и было введено.

```
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-2.asm
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab8-2 arg1 arg2 "arg3"
arg1
arg2
arg3
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

Создаю новый файл для программы и копирую в него код из третьего листинга.



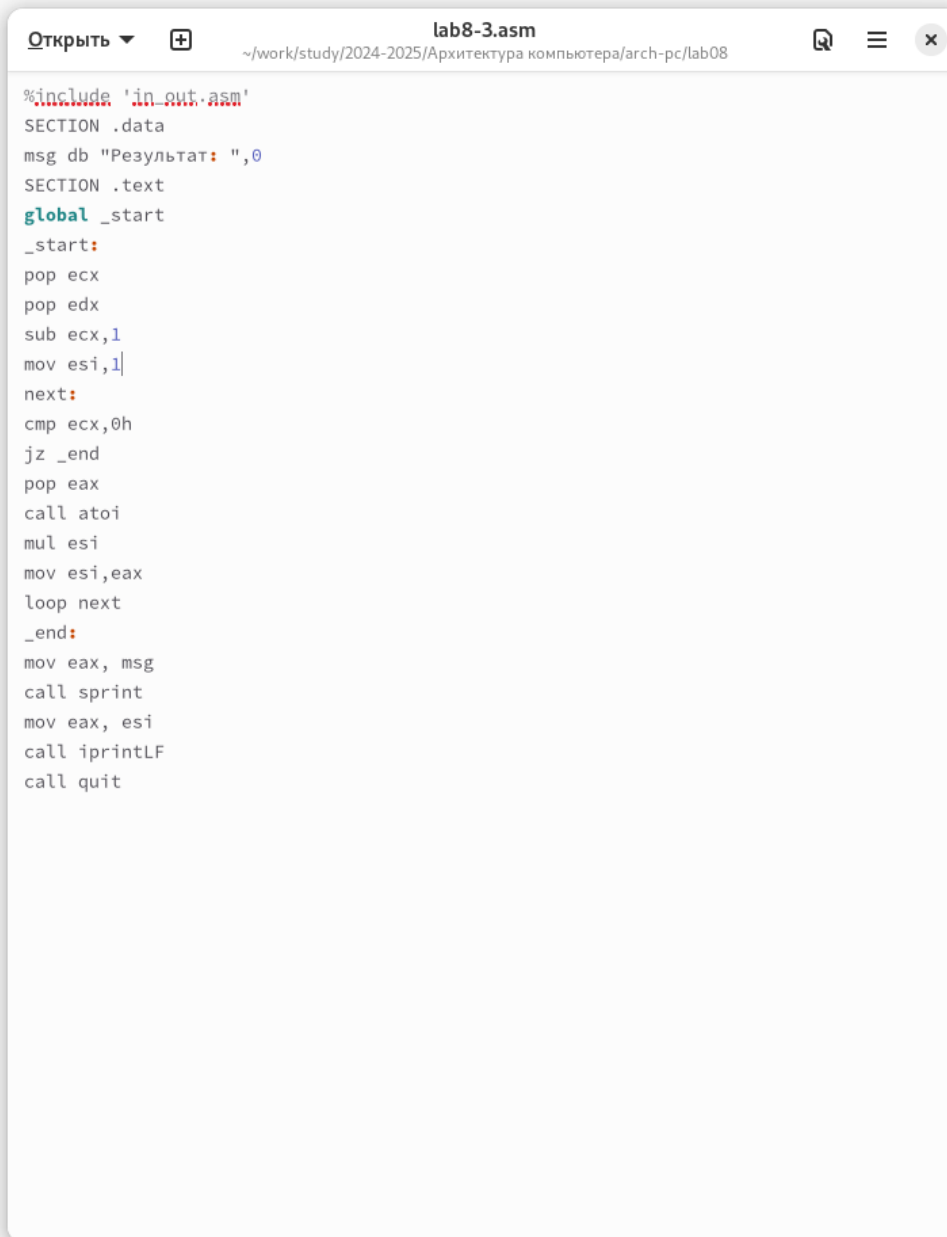
```
Открыть + lab8-3.asm ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08

%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: "
    call sprint
    mov eax, esi ; записываем сумму в регистр `eax`
    call iprintLF ; печать результата
    call quit ; завершение программы
```

Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает.

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

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала.



```
Открыть + lab8-3.asm ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08

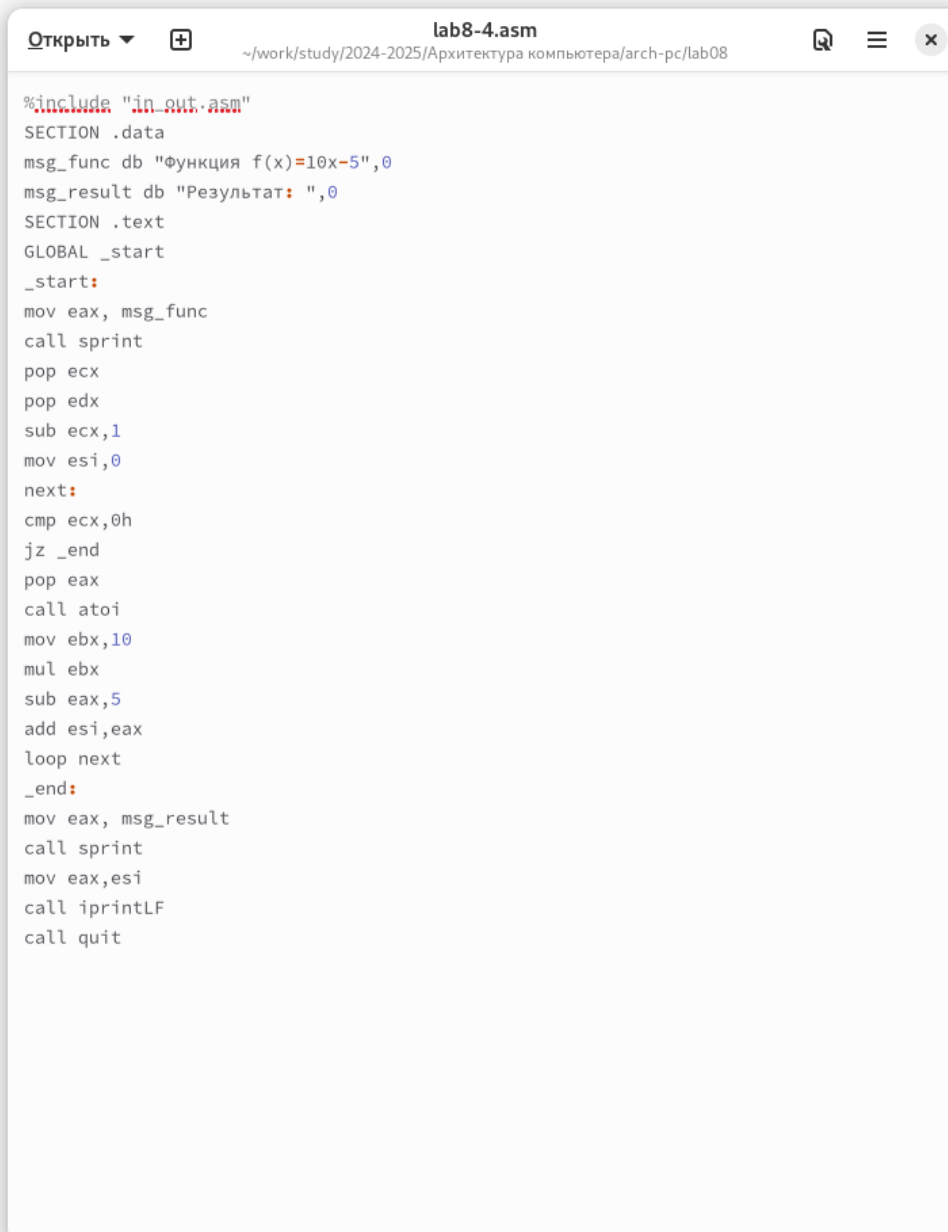
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi,1
next:
cmp ecx,0h
jz _end
pop eax
call atoi
mul esi
mov esi,eax
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```


Программа действительно теперь умножает данные на вход числа.

```
heroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-3.asm
heroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
heroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab8-3 100 2 3
Результат: 600
heroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

3.3 Задание для самостоятельной работы.

Пишу программу, которая будет находить сумма значений для функции $f(x) = 10x - 5$, которая совпадает с моим третьим вариантом.



```
lab8-4.asm
~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08

%include "in_out.asm"
SECTION .data
msg_func db "Функция f(x)=10x-5",0
msg_result db "Результат: ",0
SECTION .text
GLOBAL _start
_start:
mov eax, msg_func
call sprint
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,5
add esi,eax
loop next
_end:
mov eax, msg_result
call sprint
mov eax,esi
call iprintLF
call quit
```

Код программы:

```
%include 'in_out.asm' SECTION .data msg_func db "Функция: f(x) = 10x - 5", 0
msg_result db "Результат:", 0 SECTION .text GLOBAL _start _start: mov eax,
msg_func 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, 5 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.

```
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ nasm -f elf lab8-4.asm
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция f(x)=10x-5Результат: 80
neroun@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

4 Выводы

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием циклов а также научился обрабатывать аргументы командной строки.