

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера.

Наговицын Арсений Владимирович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация циклов в NASM	7
3.2	Обработка аргументов командной строки	12
3.3	Выполнение заданий для самостоятельной работы	17
4	Выводы	20

Список иллюстраций

3.1	Создание каталога и файла	7
3.2	Редактирование файла	7
3.3	Компиляция и запуск файла	9
3.4	Редактирование файла	9
3.5	Компиляция и запуск файла	10
3.6	Компиляция и запуск файла	10
3.7	Редактирование файла	11
3.8	Компиляция и запуск файла	11
3.9	Создание файла	12
3.10	Редактирование файла	12
3.11	Компиляция и запуск файла	13
3.12	Создание файла	13
3.13	Редактирование файла	14
3.14	Компиляция и запуск файла	15
3.15	Редактирование файла	16
3.16	Компиляция и запуск файла	17
3.17	Создание файла	17
3.18	Редактирование файла	18
3.19	Компиляция и запуск файла	19

Список таблиц

1 Цель работы

Целью данной лабораторной работы является 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. Выполнение заданий для самостоятельной работы

3 Выполнение лабораторной работы

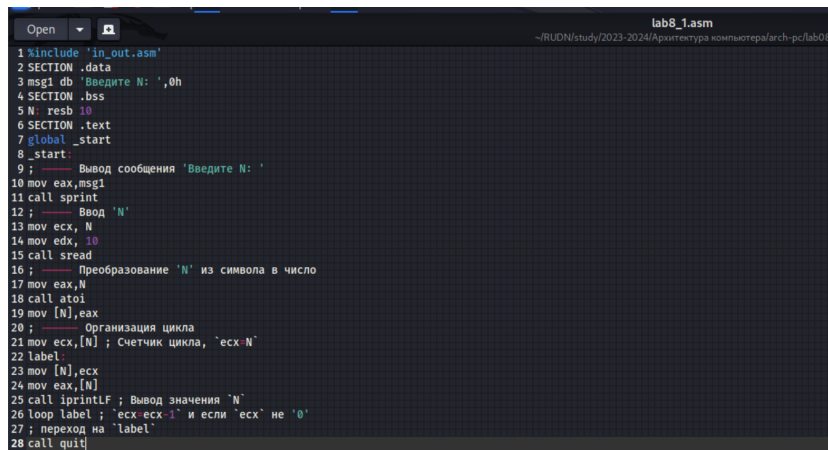
3.1 Реализация циклов в NASM

Создаю каталог для программ и перехожу в него. Создаю файл (рис. 3.1).

```
(avnagovicihn@avnagovicihn)-[~/../study/2023-2024/Архитектура компьютера/arch-pc]
$ mkdir lab08
(avnagovicihn@avnagovicihn)-[~/../study/2023-2024/Архитектура компьютера/arch-pc]
$ cd lab08
(avnagovicihn@avnagovicihn)-[~/../2023-2024/Архитектура компьютера/arch-pc/lab08]
$ touch lab8_1.asm
```

Рис. 3.1: Создание каталога и файла

Ввожу в файл текст программы из листинга 8.1(рис. 3.2).



```
lab8_1.asm
~/RUDN/study/2023-2024/Архитектура компьютера/arch-pc/lab08
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 iprintf ; Вывод значения 'N'
26 loop label ; 'ecx-ecx-1' и если 'ecx' не '0'
27 ; переход на 'label'
28 call quit
```

Рис. 3.2: Редактирование файла

Листинг 8.1. Программа вывода значений 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:
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

```

Создаю исполняемый файл и запускаю его (рис. 3.3).


```

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ nasm -f elf lab8_1.asm

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ld -m elf_i386 -o lab8_1 lab8_1.o

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ./lab8_1
Введите N: 14
14
13
12
11
10
9
8
7
6
5
4
3
2
1

```

Рис. 3.3: Компиляция и запуск файла

Данная программа выводит все числа от N до 1 включительно. Изменяю текст программы (рис. 3.4).

```

label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label

```

```

lab8_1.asm
~/RUDN/study/2023-2024/Архитектура компьютера/arch-pc/lab08

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 Вывод значения 'N'
27 loop label
28 ; переход на 'label'
29 call quit

```

Рис. 3.4: Редактирование файла

Создаю исполняемый файл и запускаю его. Ввожу в него четное значение (рис. 3.5).

```
(avnagovicihn@ avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ nasm -f elf lab8_1.asm

(avnagovicihn@ avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ld -m elf_i386 -o lab8_1 lab8_1.o

(avnagovicihn@ avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ./lab8_1
Введите N: 10
9
7
5
3
1
```

Рис. 3.5: Компиляция и запуск файла

Программа выводит все числа от N-1 до 1 с интервалом в 2 - то есть все нечётные числа. Ввожу нечетное число. Данная программа выводит бесконечную последовательность значений, так как значение регистра `ecx` переваливает за 0 (рис. 3.6).

```
(avnagovicihn@ avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ./lab8_1
Введите N: 7
6
4
2
0
4294967294
4294967292
4294967290
4294967288
4294967286
4294967284
4294967282
4294967280
4294967278
4294967276
4294967274
4294967272
4294967270
4294967268
4294967266
4294967264
4294967262
4294967260
4294967258
4294967256
```

Рис. 3.6: Компиляция и запуск файла

Изменяю текст программы: (рис. 3.7).

`label:`

`push ecx ; добавление значения ecx в стек`

```

sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop labe

```

```

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]
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
30 ; переход на 'label'
31 call quit

```

Рис. 3.7: Редактирование файла

Создаю исполняемый файл и запускаю его. Ввожу в него нечетное значение (рис. 3.8).

```

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ nasm -f elf lab8_1.asm

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ld -m elf_i386 -o lab8_1 lab8_1.o

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ./lab8_1
Введите N: 5
4
3
2
1
0

```

Рис. 3.8: Компиляция и запуск файла

3.2 Обработка аргументов командной строки

Создаю новый файл lab8_2.asm (рис. 3.9).

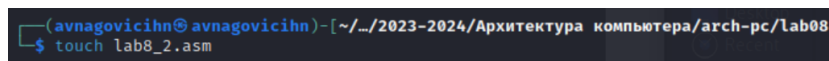


Рис. 3.9: Создание файла

Ввожу в файл текст программы из листинга 8.2(рис. 3.10).

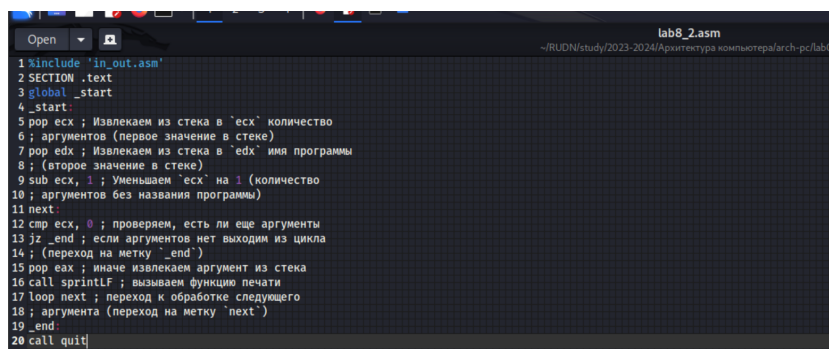


Рис. 3.10: Редактирование файла

Листинг 8.2. Программа выводящая на экран аргументы командной строки

```
%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`)
```

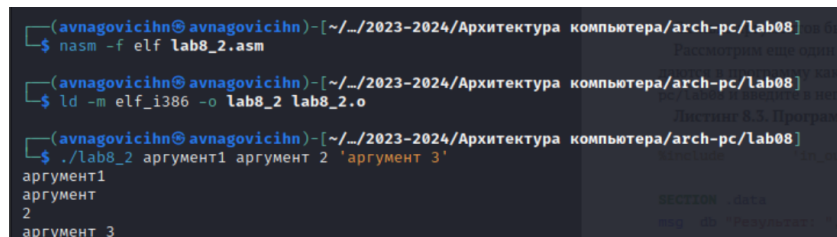
pop eax ; иначе извлекаем аргумент из стека

```
call sprintLF ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
```

_end:

```
call quit
```

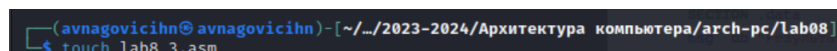
Создаю исполняемый файл и запускаю его (рис. 3.11).



```
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ nasm -f elf lab8_2.asm
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ld -m elf_i386 -o lab8_2 lab8_2.o
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ./lab8_2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

Рис. 3.11: Компиляция и запуск файла

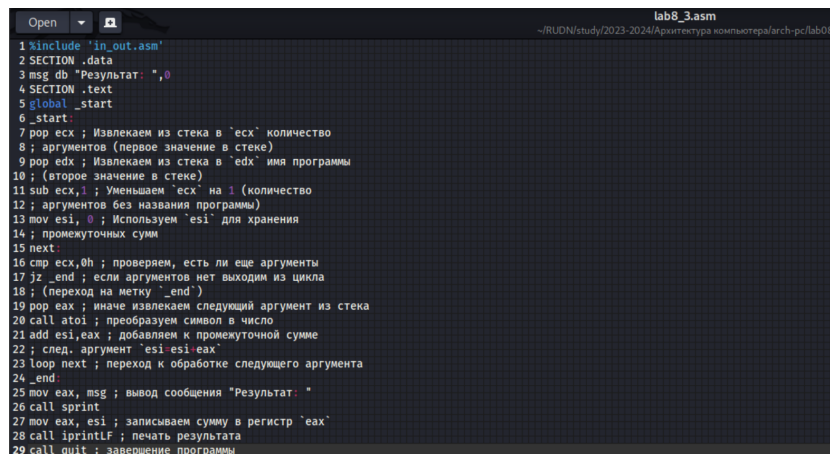
Создаю новый файл lab8_3.asm (рис. 3.12).



```
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ touch lab8_3.asm
```

Рис. 3.12: Создание файла

Ввожу в файл текст программы из листинга 8.3(рис. 3.13).



```
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. 3.13: Редактирование файла

Листинг 8.3. Программа вычисления суммы аргументов командной стро-

ки

```
%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`)

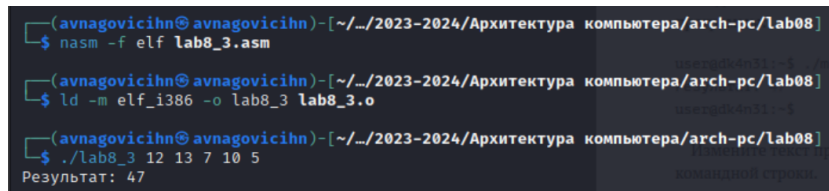
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=esi+eax`

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

```

Создаю исполняемый файл и запускаю его (рис. 3.14).



```

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ nasm -f elf lab8_3.asm

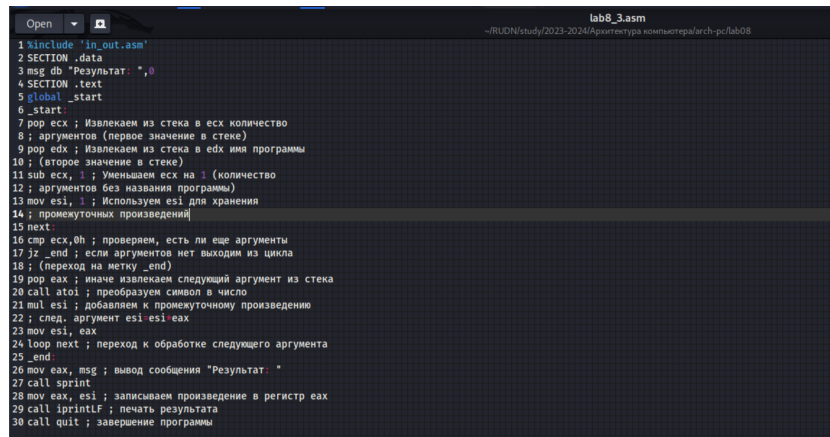
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ld -m elf_i386 -o lab8_3 lab8_3.o

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ./lab8_3 12 13 7 10 5
Результат: 47

```

Рис. 3.14: Компиляция и запуск файла

Изменяю текст программы из листинга 8.3 (рис. 3.15).



```
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, 1 ; Используем esi для хранения
14 ; промежуточных произведений
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 mul esi ; добавляем к промежуточному произведению
22 ; след. аргумент esi*eax
23 mov esi, eax
24 loop next ; переход к обработке следующего аргумента
25 _end:
26 mov eax, msg ; вывод сообщения "Результат: "
27 call printf
28 mov eax, esi ; записываем произведение в регистр eax
29 call printf ; печать результата
30 call quit ; завершение программы
```

Рис. 3.15: Редактирование файла

Листинг 8.3.1 Программа вычисления произведения аргументов командной строки

```
%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, 1 ; Используем esi для хранения
; промежуточных произведений
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
```

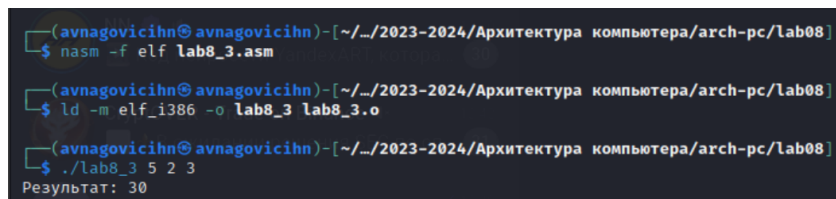


```

; (переход на метку _end)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
mul esi ; добавляем к промежуточному произведению
; след. аргумент esi=esi*eax
mov esi, eax
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем произведение в регистр eax
call iprintLF ; печать результата
call quit ; завершение программы

```

Создаю исполняемый файл и запускаю его (рис. 3.16).



```

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ nasm -f elf lab8_3.asm

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ld -m elf_i386 -o lab8_3 lab8_3.o

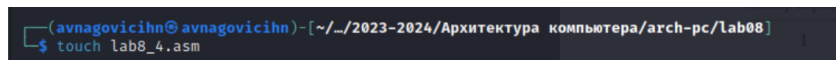
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ./lab8_3 5 2 3
Результат: 30

```

Рис. 3.16: Компиляция и запуск файла

3.3 Выполнение заданий для самостоятельной работы

Создаю новый файл lab8_4.asm (рис. 3.17).



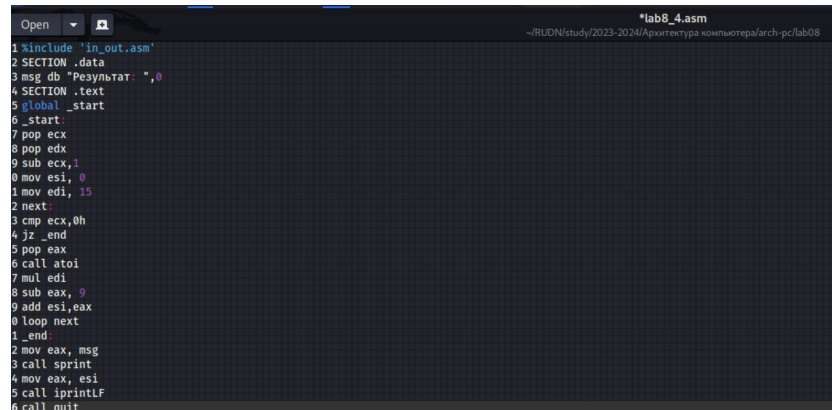
```

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ touch lab8_4.asm

```

Рис. 3.17: Создание файла

Пишу программу, которая находит сумму функции $f = 15x - 9$ (12 вариант).(рис. 3.18).



```
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 mov edi, 15
12 next:
13 cmp ecx,0h
14 jz _end
15 pop eax
16 call atoi
17 mul edi
18 sub eax, 9
19 add esi,eax
20 loop next
21 _end:
22 mov eax, msg
23 call sprint
24 mov eax, esi
25 call iprintf
26 call quit
```

Рис. 3.18: Редактирование файла

Листинг 8.4 Программа вычисления суммы функции $f = 15x - 9$.

```
%include 'in_out.asm'

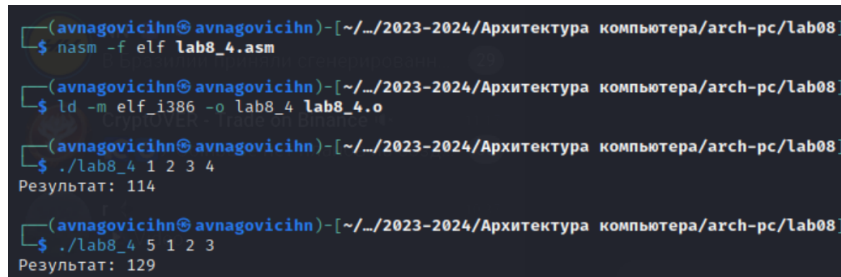
SECTION .data
msg db "Результат: ",0

SECTION .text
global _start

_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0
mov edi, 15
next:
cmp ecx,0h
jz _end
pop eax
call atoi
```

```
mul edi
sub eax, 9
add esi, eax
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

Создаю исполняемый файл и запускаю его (рис. 3.19).



```
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ nasm -f elf lab8_4.asm
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ld -m elf_i386 -o lab8_4 lab8_4.o
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ./lab8_4 1 2 3 4
Результат: 114
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab08]
$ ./lab8_4 5 1 2 3
Результат: 129
```

Рис. 3.19: Компиляция и запуск файла

4 Выводы

В ходе выполнения данной лабораторной работы я приобрёл навыки написания программ с использованием циклов и обработкой аргументов командной строки.