

Отчёт по лабораторной работе №8

Выполнил студент НКАбд-02-25

Арина Андреевна Дрекина

Содержание

1	Цель работы.	3
2	Порядок выполнения лабораторной работы.	4
3	Реализация циклов в NASM.	5
4	Обработка аргументов командной строки.	14
5	Задание для самостоятельной работы.	23
6	Вывод.	26

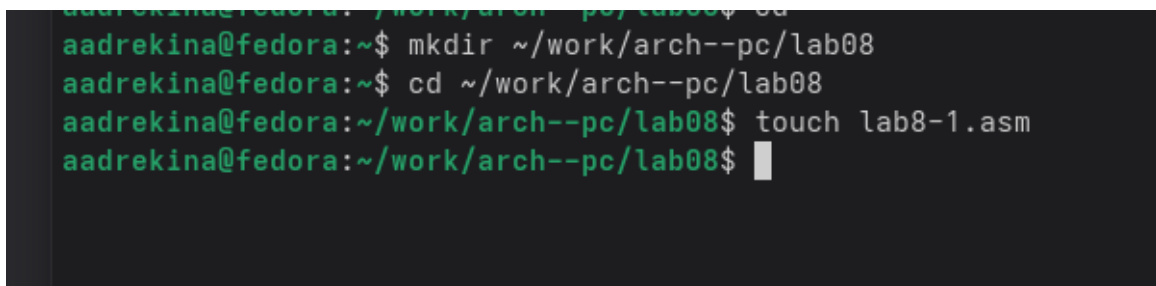
1 Цель работы.

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

2 Порядок выполнения лабораторной работы.

3 Реализация циклов в NASM.

Я создала каталог для программ, которые будут созданы во время выполнения лабораторной работы №8. (Рисунок 3.1)




```
aadrekina@fedora:~$ mkdir ~/work/arch--pc/lab08
aadrekina@fedora:~$ cd ~/work/arch--pc/lab08
aadrekina@fedora:~/work/arch--pc/lab08$ touch lab8-1.asm
aadrekina@fedora:~/work/arch--pc/lab08$
```

Рисунок 3.1: Создание каталога для лабораторной работы.

Затем я создала текстовый файл для программы и добавила в него Листинг 8.1. Я внимательно изучила программу. (Рисунок 3.2)

Вт, 25 ноября 23:32

Открыть ▾ 

• lab8-1.asm
~/work/arch--pc/lab08

сина_ | lab5-2-2.asm | lab5-1-2.asm | lab6-1.asm | ЛО7_Дрекина_

```
-----  
; 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
```

Рисунок 3.2: Текст программы.

Далее я запустила этот файл. (Рисунок 3.3)

```
aadrekina@fedora:~/work/arch--pc/lab08$ nasm -f elf lab8-1.asm
aadrekina@fedora:~/work/arch--pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aadrekina@fedora:~/work/arch--pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
aadrekina@fedora:~/work/arch--pc/lab08$
```

Рисунок 3.3: Запуск программы.

Листинг 8.1:

```
;-----
; Программа вывода значений регистра 'ecx'
;-----

%include 'in_out.asm'

SECTION .data
msg1 db 'Введите N: ',0h

SECTION .bss
N: resb 10

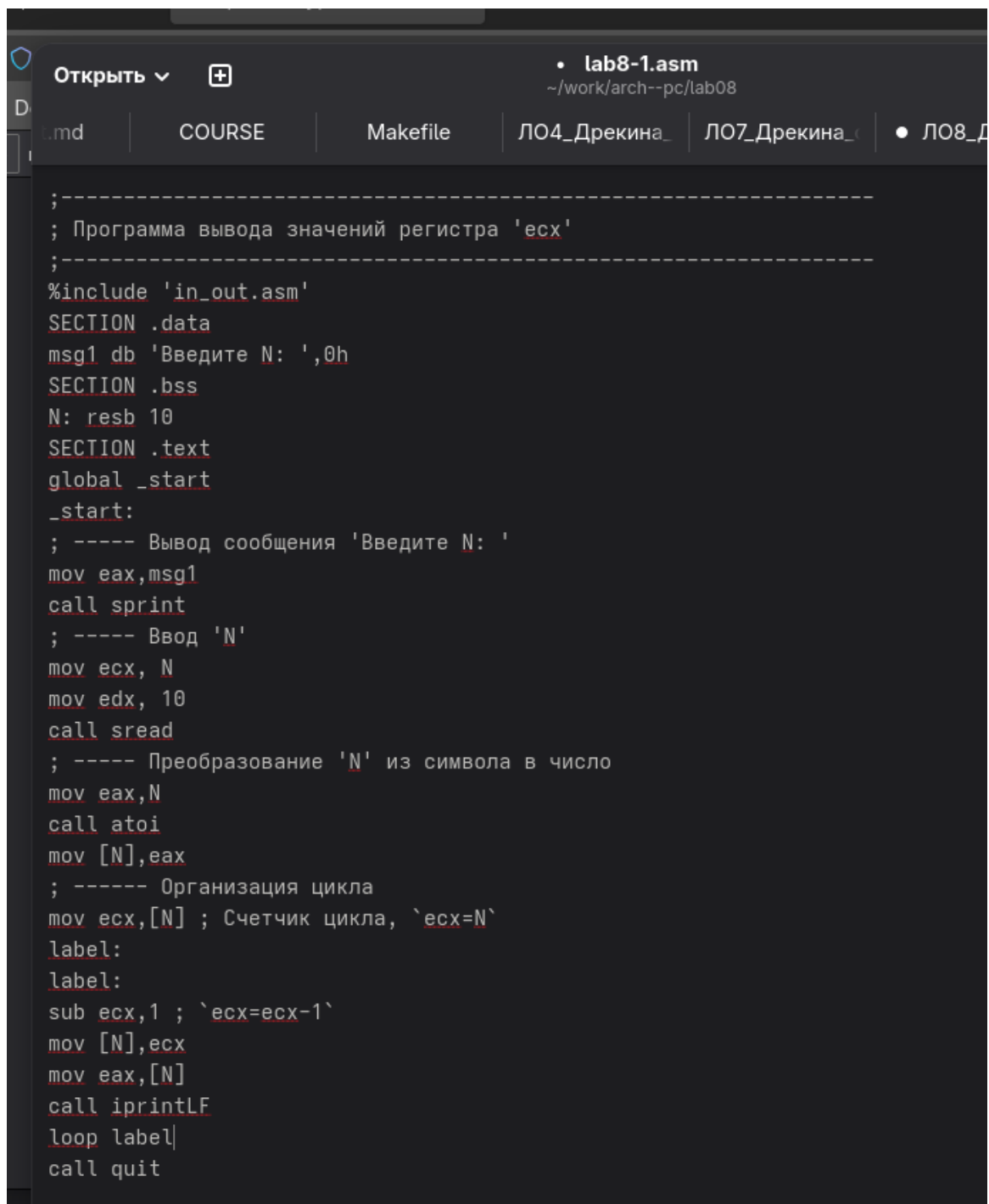
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
```

```

; ----- 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

```

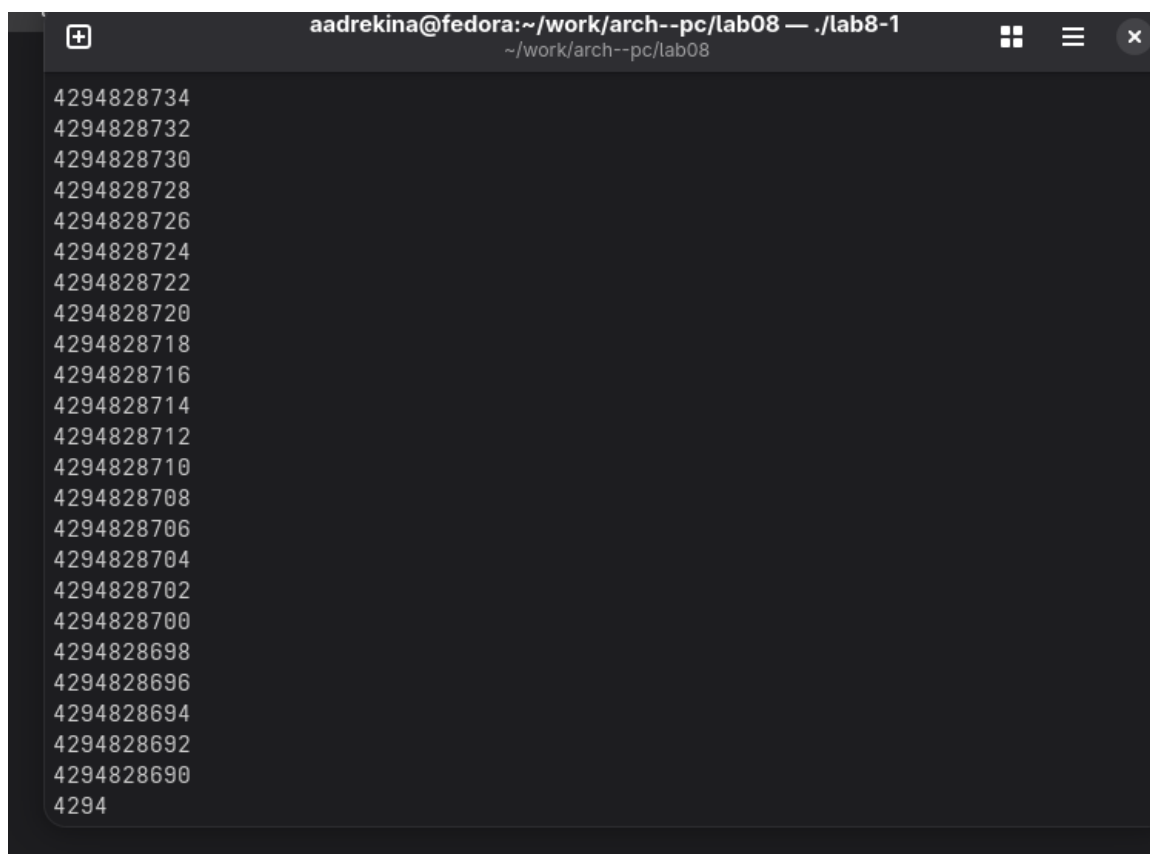
Затем я внесла изменения уже в созданный Листинг. (Рисунок 3.4)



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

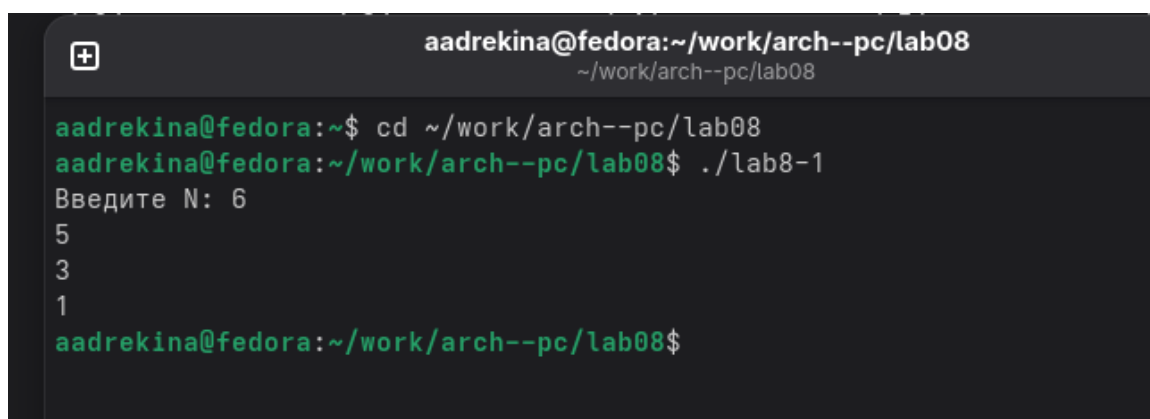
Рисунок 3.4: Изменения программы.

Потом я запустила программу. (Рисунок 3.5, Рисунок 3.6)



```
aadrekina@fedora:~/work/arch--pc/lab08 — ./lab8-1
~/work/arch--pc/lab08
4294828734
4294828732
4294828730
4294828728
4294828726
4294828724
4294828722
4294828720
4294828718
4294828716
4294828714
4294828712
4294828710
4294828708
4294828706
4294828704
4294828702
4294828700
4294828698
4294828696
4294828694
4294828692
4294828690
4294
```

Рисунок 3.5: Запуск программы. Значение 5.



```
aadrekina@fedora:~/work/arch--pc/lab08
~/work/arch--pc/lab08
aadrekina@fedora:~$ cd ~/work/arch--pc/lab08
aadrekina@fedora:~/work/arch--pc/lab08$ ./lab8-1
Введите N: 6
5
3
1
aadrekina@fedora:~/work/arch--pc/lab08$
```

Рисунок 3.6: Запуск программы. Значение 6.

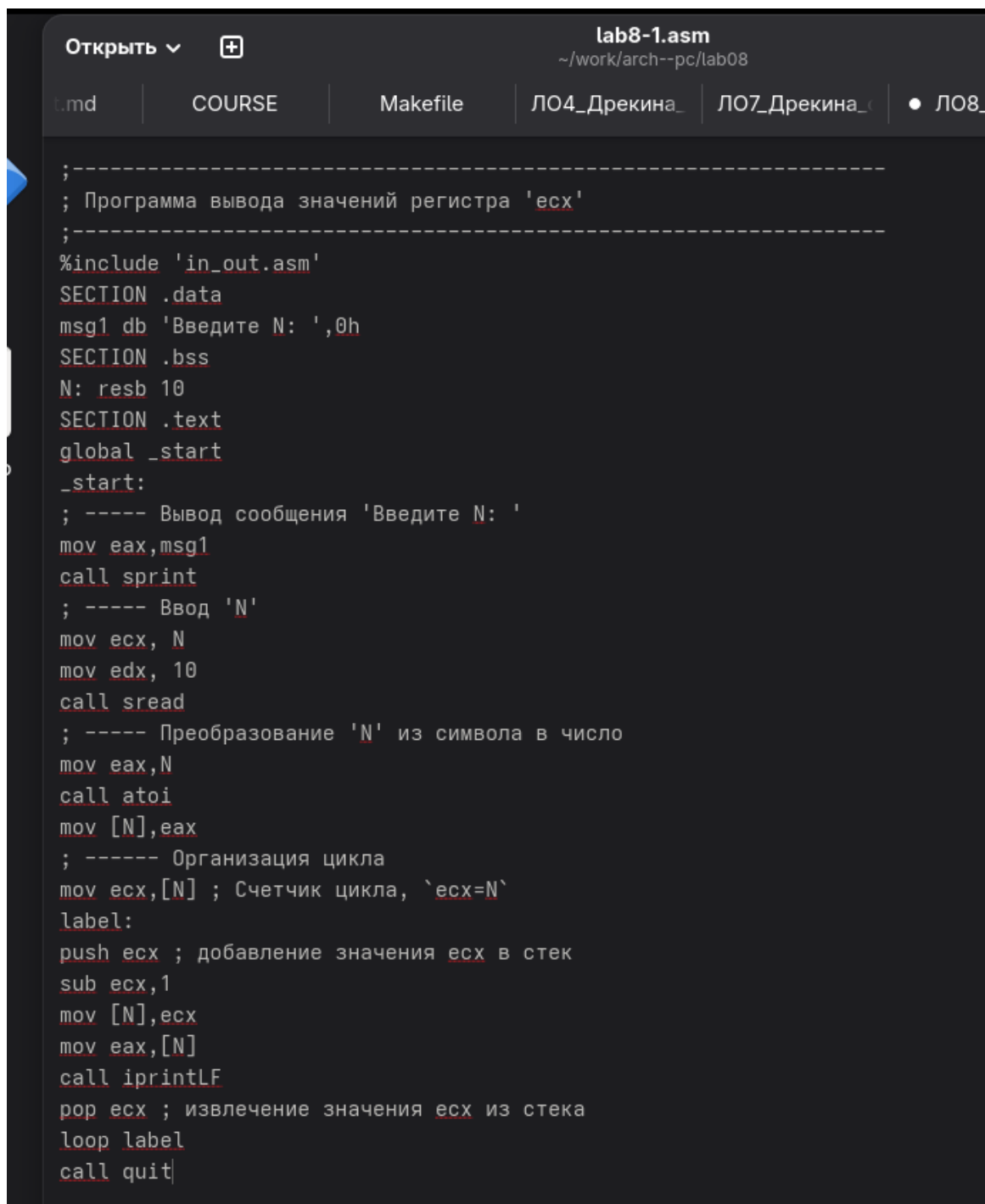
Я ввела два значения, 5 и 6. Увидев разницу в выводе я решила ввести еще пару значений и заметила, что для четных чисел ответ выводится, а для нечетных выводится

бесконечное количество чисел.

Первые изменения в программе:

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

Затем я внесла еще одно изменение в программу. (Рисунок 3.7)



```
lab8-1.asm
~/work/arch--pc/lab08

t.md | COURSE | Makefile | ЛО4_Дрекина_ | ЛО7_Дрекина_ | ● ЛО8_

;-----
; 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:
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 label
call quit
```

Рисунок 3.7: Изменение в программе.

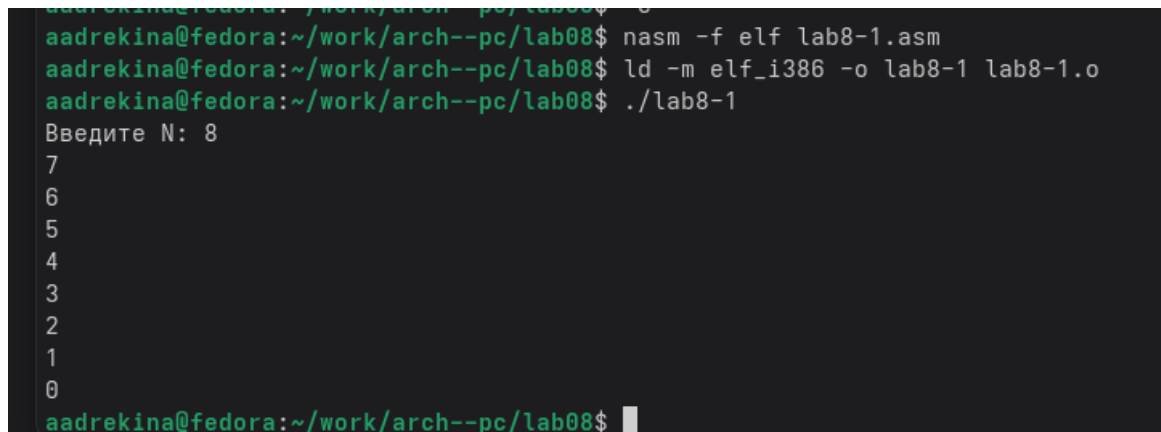
Вторые изменения в программме:

```

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

```

Далее я запустила измененный файл. (Рисунок 3.8)



```

aadrekina@fedora:~/work/arch--pc/lab08$ nasm -f elf lab8-1.asm
aadrekina@fedora:~/work/arch--pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aadrekina@fedora:~/work/arch--pc/lab08$ ./lab8-1
Введите N: 8
7
6
5
4
3
2
1
0
aadrekina@fedora:~/work/arch--pc/lab08$

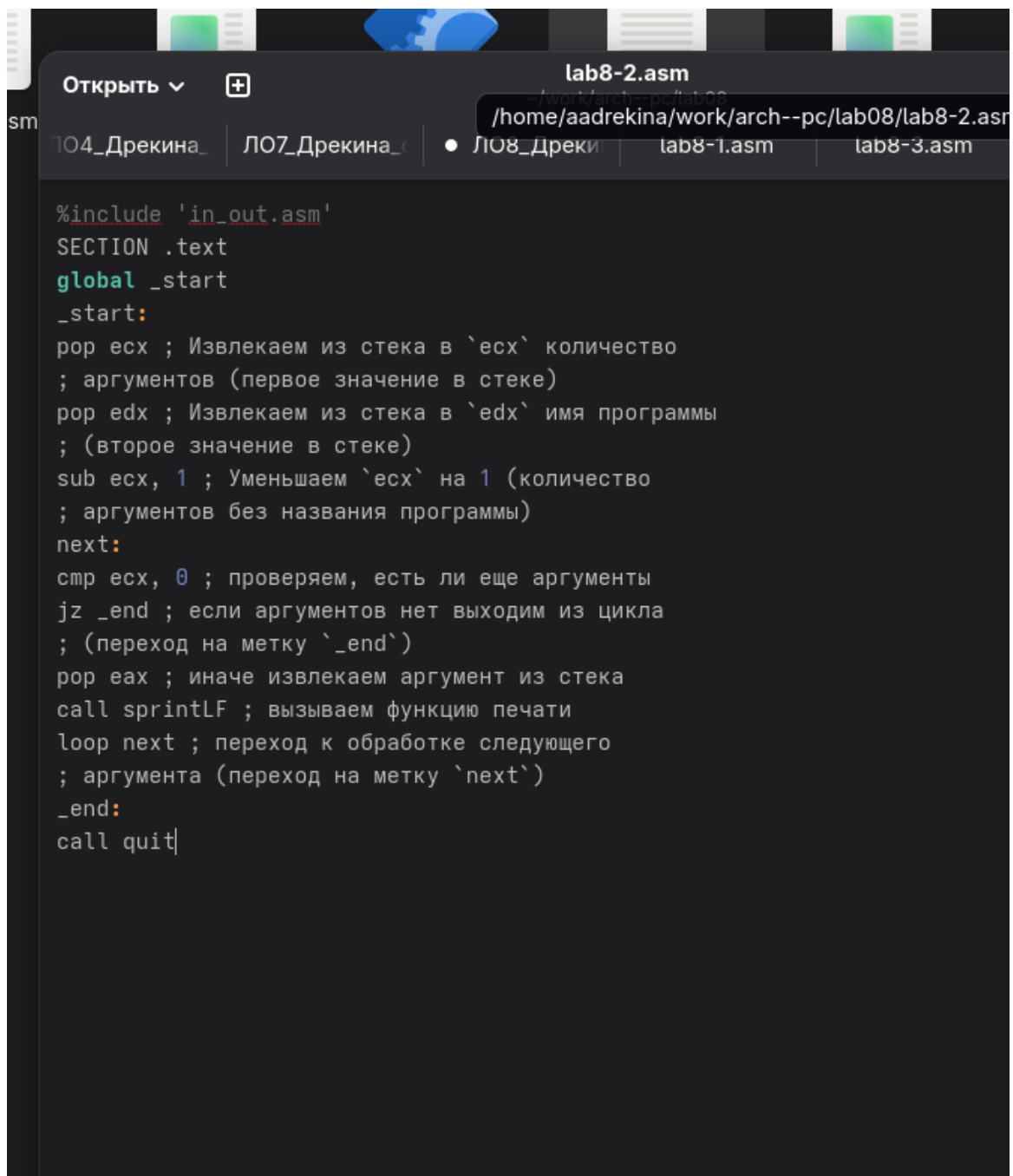
```

Рисунок 3.8: Запуск программы.

Теперь программа выводит правильный ответ, она уменьшает полученное на вход число на 1, причем вывод начинается уже с уменьшенного числа, а не с введенного как было в первой программе.

4 Обработка аргументов командной строки.

Я создала еще один текстовый файл. И ввела туда текст программы из Листинга 8.2.(Рисунок 4.1)



```
%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 printf ; вызываем функцию печати
    loop next ; переход к обработке следующего
              ; аргумента (переход на метку `next`)
_end:
    call quit
```

Рисунок 4.1: Текст программы.

Затем я запустила эту программу и указала аргументы. (Рисунок 4.2)

```
aadrekina@fedora:~/work/arch--pc/lab08$ nasm -f elf lab8-2.asm
aadrekina@fedora:~/work/arch--pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
aadrekina@fedora:~/work/arch--pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
aadrekina@fedora:~/work/arch--pc/lab08$
```

Рисунок 4.2: Запуск программы.

У меня вывелись все три аргумента, однако с отличиями. Например: номер первого аргумента был написан без пробела, номер второго перешел на новую строку, а номер третьего был написан через пробел.

Листинг 8.2:

```
;-----
; Обработка аргументов 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 sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

Затем я создала еще один файл и вставила туда текст из листинга 8.3. (Рисунок 4.3)

```
%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
```

Рисунок 4.3: Текст программы.

Теперь я запустила файл. (Рисунок 4.4)

```
aadrekina@fedora:~/work/arch--pc/lab08$ nasm -f elf lab8-3.asm
aadrekina@fedora:~/work/arch--pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aadrekina@fedora:~/work/arch--pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
aadrekina@fedora:~/work/arch--pc/lab08$
```

Рисунок 4.4: Запуск программы.

Я ввела те же значения, что и в лекции, чтобы проверить корректность работы. Результат, который вывелся у меня, совпал с результатом из лекции.

Листинг 8.3:

```
%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
```

Далее я изменила программу, чтобы она не складывала, а перемножала числа. (Рисунок 4.5)

```
lab8-3.asm
~/work/arch--pc/lab08

Makefile | ЛО4_Дрекина_ | ЛО7_Дрекина_ | ЛО8_Дрекина_ | lab8

%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 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
imul 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
```

Рисунок 4.5: Изменения в тексте программы.

Теперь я создала исполняемый файл и запустила программу. (Рисунок 4.6)

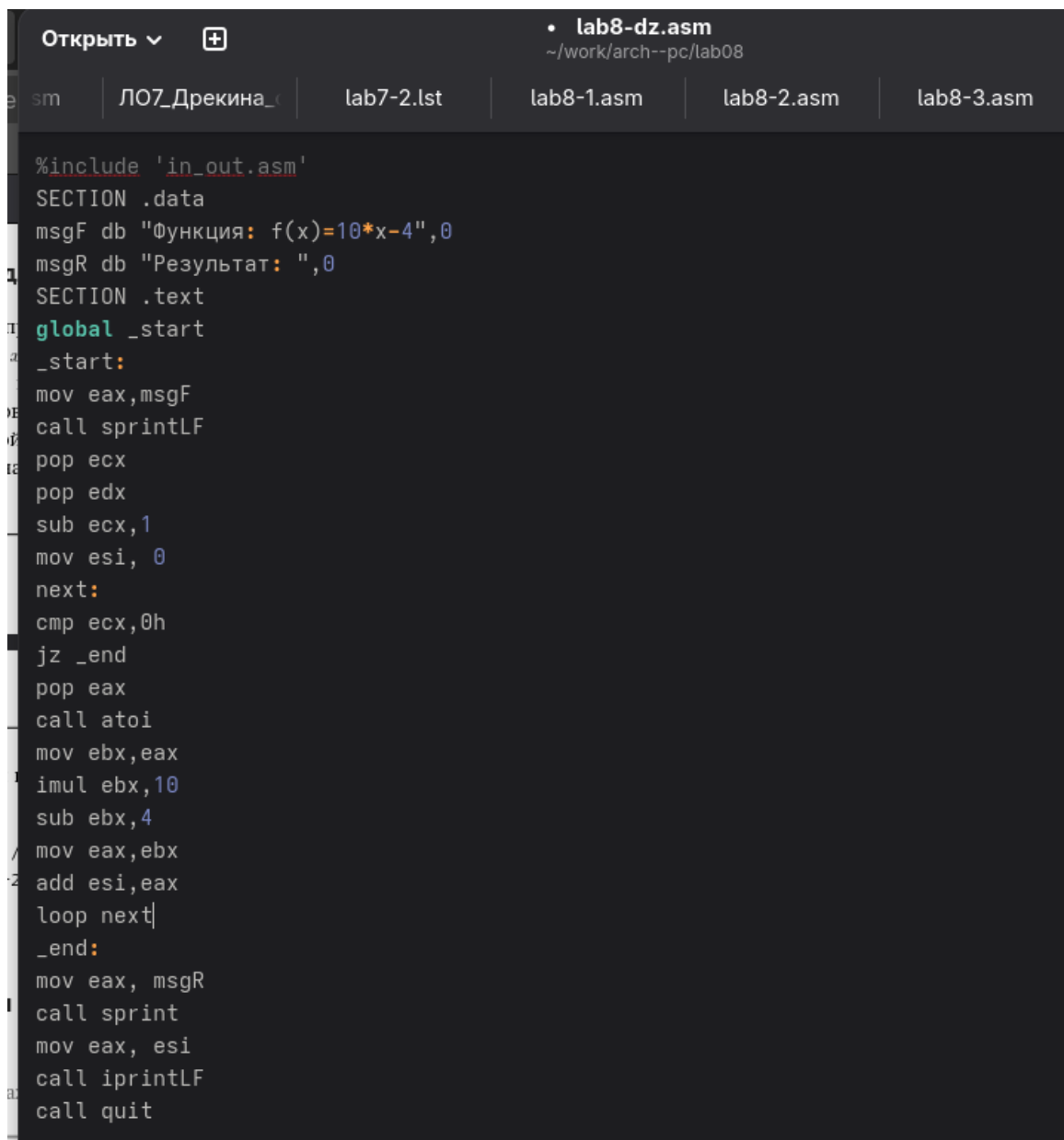
```
Результат: 0
aadrekina@fedora:~/work/arch--pc/lab08$ nasm -f elf lab8-3.asm
aadrekina@fedora:~/work/arch--pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aadrekina@fedora:~/work/arch--pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
aadrekina@fedora:~/work/arch--pc/lab08$
```

Рисунок 4.6: Запуск программы.

Я получила вывод, и затем перепроверила его вручную. Ответ который мне вывел терминал, и тот который получила я, совпали.

5 Задание для самостоятельной работы.

Я написала программу, которая будет находить сумму значений функции для x .
Функцию я взяла из таблицы вариантов из лекции. Мой вариант - 9. (Рисунок 5.1)




```
Открыть ▾  • lab8-dz.asm  
~/work/arch--pc/lab08  
asm | ЛО7_Дрекина_с | lab7-2.lst | lab8-1.asm | lab8-2.asm | lab8-3.asm  
  
%include 'in_out.asm'  
SECTION .data  
msgF db "Функция: f(x)=10*x-4",0  
msgR db "Результат: ",0  
SECTION .text  
global _start  
_start:  
mov eax,msgF  
call sprintf  
pop ecx  
pop edx  
sub ecx,1  
mov esi, 0  
next:  
cmp ecx,0h  
jz _end  
pop eax  
call atoi  
mov ebx,eax  
imul ebx,10  
sub ebx,4  
mov eax,ebx  
add esi,eax  
loop next  
_end:  
mov eax, msgR  
call sprintf  
mov eax, esi  
call iprintLF  
call quit
```

Рисунок 5.1: Текст программы.

После этого я запустила программу. (Рисунок 5.2)


```
Результат: 10
aadrekina@fedora:~/work/arch--pc/lab08$ nasm -f elf lab8-dz.asm
aadrekina@fedora:~/work/arch--pc/lab08$ ld -m elf_i386 -o lab8-dz lab8-dz.o
aadrekina@fedora:~/work/arch--pc/lab08$ ./lab8-dz 1 2 3 4
Функция:  $f(x)=10*x-4$ 
Результат: 84
aadrekina@fedora:~/work/arch--pc/lab08$
```

Рисунок 5.2: Запуск программы.

Я перепроверила вручную, и сравнила полученные ответ. Они совпали. Это означает, что программу я написала правильно и освоила материал лекции.

6 Вывод.

В ходе выполнения 8 лабораторной работы я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки. А также применила их на практике.