

Отчёта по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

ДЖАЛЛОХ ИШМАИЛ

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Реализация циклов в NASM	6
3.2	Обработка аргументов командной строки.	10
3.3	Задание для самостоятельной работы	14
4	Выводы	17

Список иллюстраций

3.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code>	6
3.2	Заполняем файл	7
3.3	Запускаем файл и проверяем его работу	7
3.4	Изменяем файл	8
3.5	Запускаем файл и смотрим на его работу	8
3.6	Редактируем файл	9
3.7	Проверяем, сошелся ли наш вывод с данным в условии выводом .	10
3.8	Создаем файл командой <code>touch</code>	10
3.9	Заполняем файл	11
3.10	Смотрим на работу программ	11
3.11	Создаем файл командой <code>touch</code>	12
3.12	Заполняем файл	12
3.13	Смотрим на работу программы	12
3.14	Изменяем файл	13
3.15	Проверяем работу файла(работает правильно)	13
3.16	Создаем файл командой <code>touch</code>	14
3.17	Пишем программу	15
3.18	Смотрим на работу программы при $x1=5$ $x2=3$ $x1=4$ (всё верно) . .	15
3.19	Смотрим на работу программы при $x1=1$ $x2=3$ $x1=7$ (всё верно) . .	16

1 Цель работы

Изучить работу циклов и обработкой аргументов командной строки.

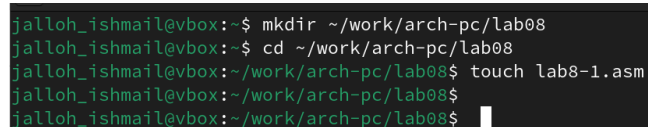
2 Задание

Написать программы с использованием циклов и обработкой аргументов командной строки.

3 Выполнение лабораторной работы

3.1 Реализация циклов в NASM

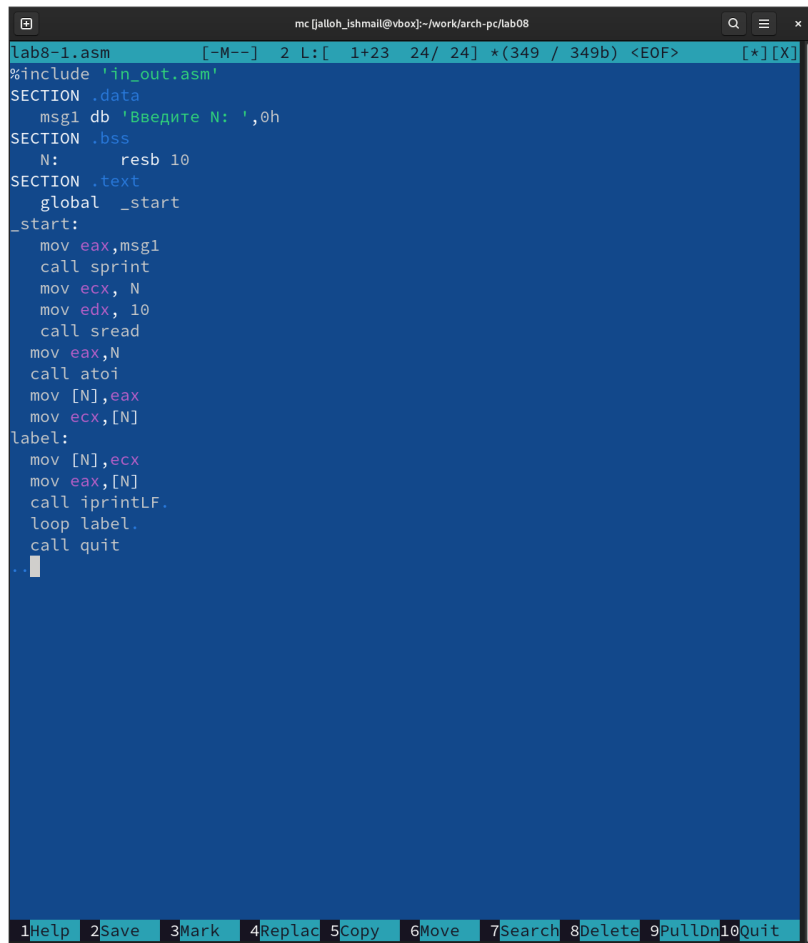
Создаем каталог для программ ЛБ8, и в нем создаем файл (рис. fig. 3.1).



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

Рис. 3.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.1 (рис. fig. 3.2).



```
lab8-1.asm [-M--] 2 L: [ 1+23 24/ 24] *(349 / 349b) <EOF> [*][X]
#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. 3.2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.3).

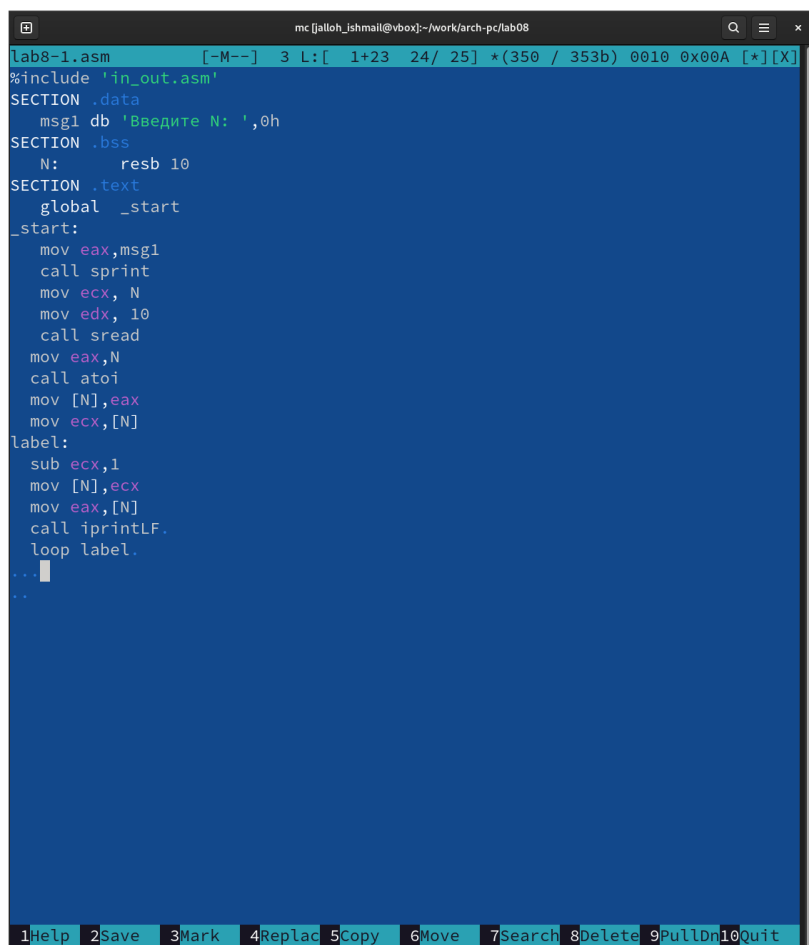


```
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
jalloh_ishmail@vbox:~/work/arch-pc/lab08$
```

Рис. 3.3: Запускаем файл и проверяем его работу

Снова открываем файл для редактирования и изменяем его, добавив измене-

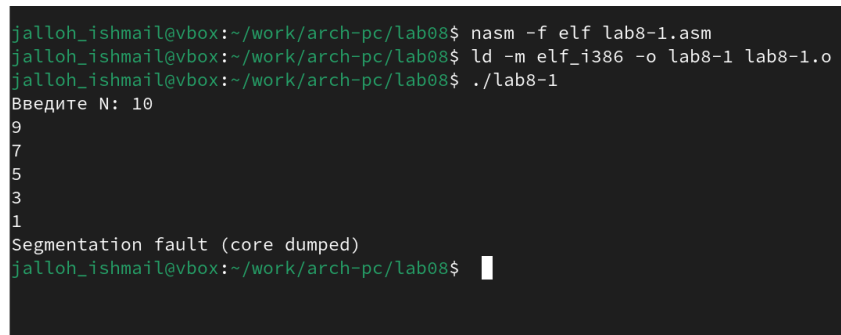
ние значения регистра в цикле (рис. fig. 3.4).



```
lab8-1.asm [-M--] 3 L: [ 1+23 24/ 25] *(350 / 353b) 0010 0x00A [*][X]
#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:
    sub ecx,1
    mov [N],ecx
    mov eax,[N]
    call iprintLF.
    loop label.
..
..
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit
```

Рис. 3.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.5).



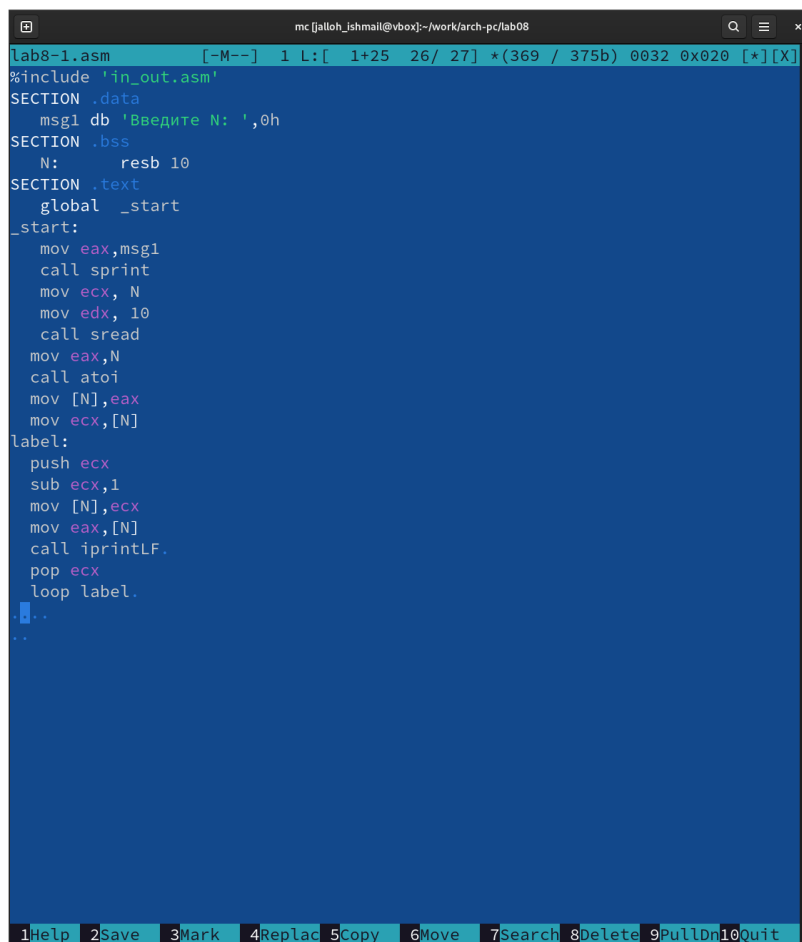
```
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
Segmentation fault (core dumped)
jalloh_ishmail@vbox:~/work/arch-pc/lab08$
```

Рис. 3.5: Запускаем файл и смотрим на его работу

Регистр `ecx` принимает значения 9,7,5,3,1(на вход подается число 10, в цикле `label` данный регистр уменьшается на 2 командой `sub` и `loop`).

Число проходов цикла не соответствует числу `N`, так как уменьшается на 2.

Снова открываем файл для редактирования и изменяем его, чтобы все корректно работало (рис. fig. 3.6).



```
lab8-1.asm [-M--] 1 L: [ 1+25 26/ 27] *(369 / 375b) 0032 0x020 [*][X]
#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:
    push ecx
    sub ecx,1
    mov [N],ecx
    mov eax,[N]
    call iprintLF
    pop ecx
    loop label
    ..
    ..
```

Рис. 3.6: Редактируем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.7).

```

jalloh_ishmail@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
Segmentation fault (core dumped)
jalloh_ishmail@vbox:~/work/arch-pc/lab08$

```

Рис. 3.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

В данном случае число проходов цикла равна числу N.

3.2 Обработка аргументов командной строки.

Создаем новый файл (рис. fig. 3.8).

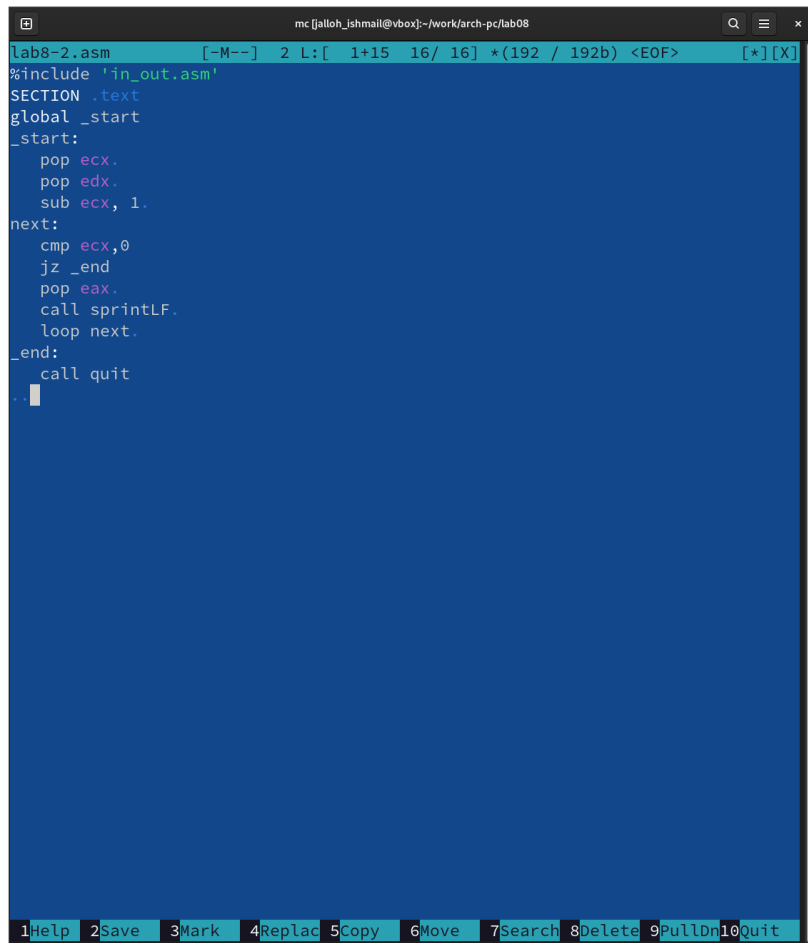
```

jalloh_ishmail@vbox:~/work/arch-pc/lab08$ touch lab8-2.asm
jalloh_ishmail@vbox:~/work/arch-pc/lab08$
jalloh_ishmail@vbox:~/work/arch-pc/lab08$

```

Рис. 3.8: Создаем файл командой touch

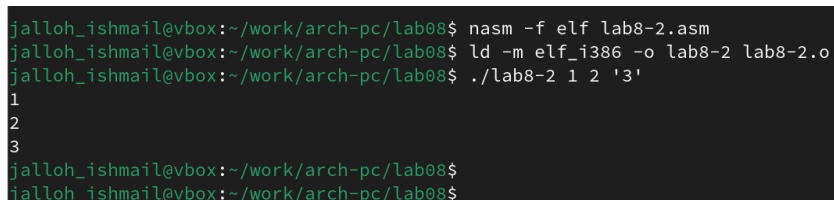
Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.2 (рис. fig. 3.9).

A screenshot of a text editor window titled 'mc [jalloh_ishmail@vbox]~/work/arch-pc/lab08'. The editor displays assembly code for 'lab8-2.asm'. The code includes a header line with metadata, followed by an include directive for 'in_out.asm'. It then defines a text section and a global symbol '_start'. The main logic starts at '_start:', where it pops values from the stack into 'ecx' and 'edx', subtracts 1 from 'ecx', and enters a loop labeled 'next:'. The loop compares 'ecx' with 0, jumps to '_end' if zero, pops 'eax', calls 'sprintf', and loops back. The loop ends at '_end:' with a 'call quit' instruction. The editor has a dark blue background and a status bar at the bottom with menu items like Help, Save, Mark, etc.

```
lab8-2.asm [-M--] 2 L:[ 1+15 16/ 16] *(192 / 192b) <EOF> [*][X]
#include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx.
    pop edx.
    sub ecx, 1.
next:
    cmp ecx,0
    jz _end
    pop eax.
    call sprintf.
    loop next.
_end:
    call quit
..
```

Рис. 3.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, указав аргументы (рис. fig. 3.10).

A screenshot of a terminal window showing the compilation and execution of the assembly program. The user runs 'nasm -f elf lab8-2.asm' to create an object file, then 'ld -m elf_i386 -o lab8-2 lab8-2.o' to create an executable. Finally, they run './lab8-2 1 2 '3'' which prints the numbers 1, 2, and 3 on separate lines. The terminal has a dark background and green text.

```
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ./lab8-2 1 2 '3'
1
2
3
jalloh_ishmail@vbox:~/work/arch-pc/lab08$
jalloh_ishmail@vbox:~/work/arch-pc/lab08$
```

Рис. 3.10: Смотрим на работу программ

Программой было обработано 3 аргумента.

Создаем новый файл lab8-3.asm (рис. fig. 3.11).

```
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ touch lab8-3.asm
jalloh_ishmail@vbox:~/work/arch-pc/lab08$
```

Рис. 3.11: Создаем файл командой touch

Открываем файл и заполняем его в соответствии с листингом 8.3 (рис. fig. 3.12).

```
lab8-3.asm  [-M--]  1 L: [ 1+23 24/ 24] *(333 / 333b) <EOF>  [*] [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
```

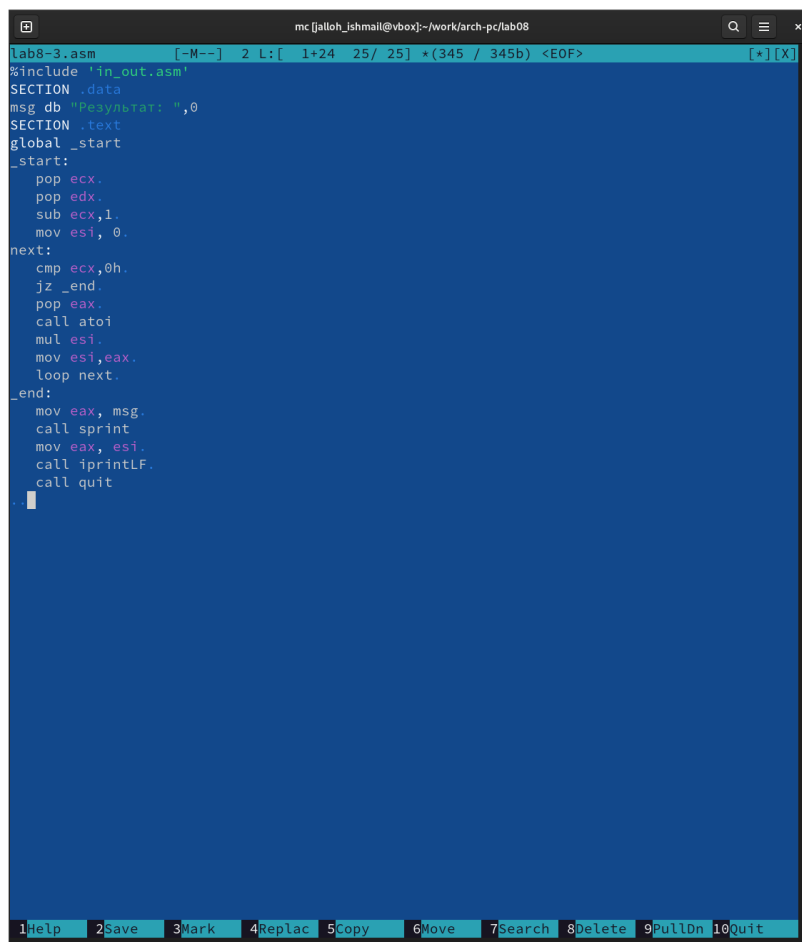
Рис. 3.12: Заполняем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. fig. 3.13).

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

Рис. 3.13: Смотрим на работу программы

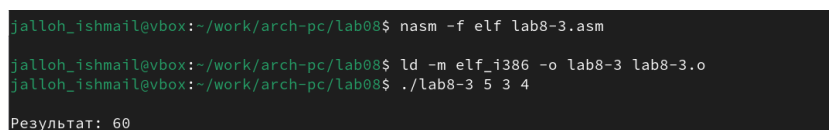
Снова открываем файл для редактирования и изменяем его, чтобы вычислялось произведение вводимых значений (рис. fig. 3.14).



```
lab8-3.asm [-M--] 2 L:[ 1+24 25/ 25] *(345 / 345b) <EOF> [*][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
    mul esi.
    mov esi,eax.
    loop next.
_end:
    mov eax, msg.
    call sprint
    mov eax, esi.
    call iprintLF.
    call quit
```

Рис. 3.14: Изменяем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. fig. 3.15).



```
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ./lab8-3 5 3 4
Результат: 60
```

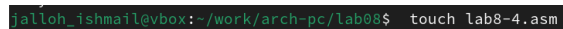
Рис. 3.15: Проверяем работу файла(работает правильно)

3.3 Задание для самостоятельной работы

ВАРИАНТ-15

1. Напишите 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. Создайте исполняемый файл и проверьте его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$.

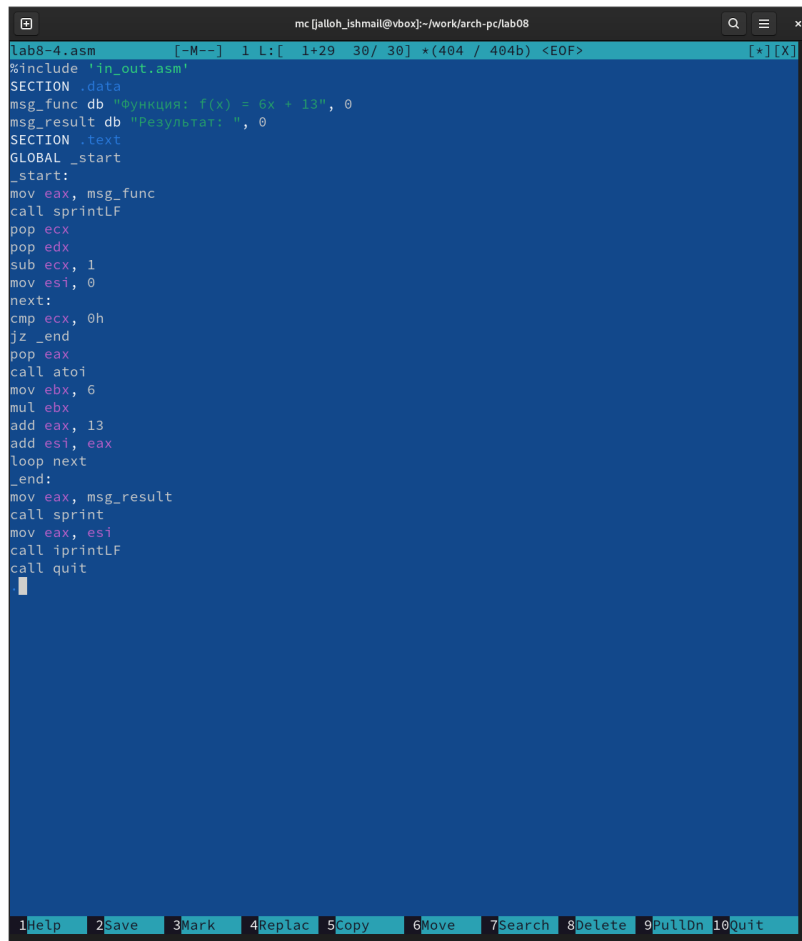
Создаем новый файл (рис. fig. 3.16).



```
jalloh_ishmail@vbox: ~/work/arch-pc/lab08$ touch lab8-4.asm
```

Рис. 3.16: Создаем файл командой touch

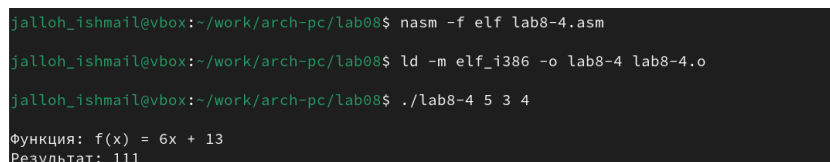
Открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения $6x + 13$ (рис. fig. 3.17).



```
lab8-4.asm [-M--] 1 L:[ 1+29 30/ 30] *(404 / 404b) <EOF> [*][X]
#include 'in_out.asm'
SECTION .data
msg_func db "Функция: f(x) = 6x + 13", 0
msg_result db "Результат: ", 0
SECTION .text
GLOBAL _start
_start:
mov eax, msg_func
call sprintf
pop ecx
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
mov ebx, 6
mul ebx
add eax, 13
add esi, eax
loop next
_end:
mov eax, msg_result
call sprintf
mov eax, esi
call iprintf
call quit
.
```

Рис. 3.17: Пишем программу

Транслируем файл и смотрим на работу программы (рис. fig. 3.18).



```
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ./lab8-4 5 3 4
Функция: f(x) = 6x + 13
Результат: 111
```

Рис. 3.18: Смотрим на работу программы при $x_1=5$ $x_2=3$ $x_3=4$ (всё верно)

Транслируем файл и смотрим на работу программы (рис. fig. 3.19).

```
jalloh_ishmail@vbox:~/work/arch-pc/lab08$ ./lab8-4 1 3 7
Функция:  $f(x) = 6x + 13$ 
Результат: 105
jalloh_ishmail@vbox:~/work/arch-pc/lab08$
```

Рис. 3.19: Смотрим на работу программы при $x_1=1$ $x_2=3$ $x_1=7$ (всё верно)

4 Выводы

Мы научились решать программы с использованием циклов и обработкой аргументов командной строки.