

Отчет по лабораторной работе №7

Костеренко Полина

Содержание

1	1. Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файлы листинга	11
2.3	Задание для самостоятельной работы	14
3	Выводы	19

Список иллюстраций

2.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code>	6
2.2	Заполняем файл	7
2.3	Запускаем файл и смотрим на его работу	7
2.4	Изменяем файл	8
2.5	Запускаем файл и смотрим на его работу	8
2.6	Редактируем файл	9
2.7	Проверяем, сошелся ли наш вывод с данным в условии выводом . .	9
2.8	Создаем файл командой <code>touch</code>	9
2.9	Заполняем файл	10
2.10	Смотрим на работу программ	11
2.11	Создаем файл листинга	11
2.12	Изучаем файл	12
2.13	Удаляем операндум из файла	13
2.14	Транслируем файл	13
2.15	Изучаем файл с ошибкой	14
2.16	Создаем файл командой <code>touch</code>	15
2.17	Пишем программу	15
2.18	Смотрим на работу программы(всё верно)	16
2.19	Создаем файл командой <code>touch</code>	16
2.20	Пишем программу	17
2.21	Проверяем работу программы	17
2.22	Проверяем работу программы	18

Список таблиц

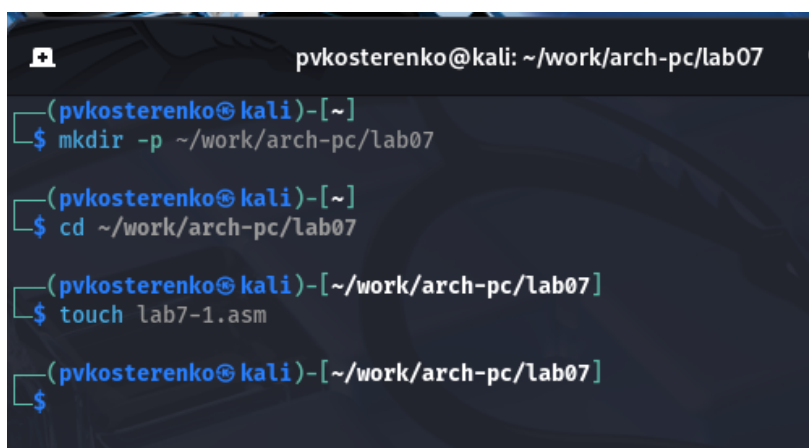
1 1. Цель работы

Освоить условного и безусловного перехода. Ознакомиться с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

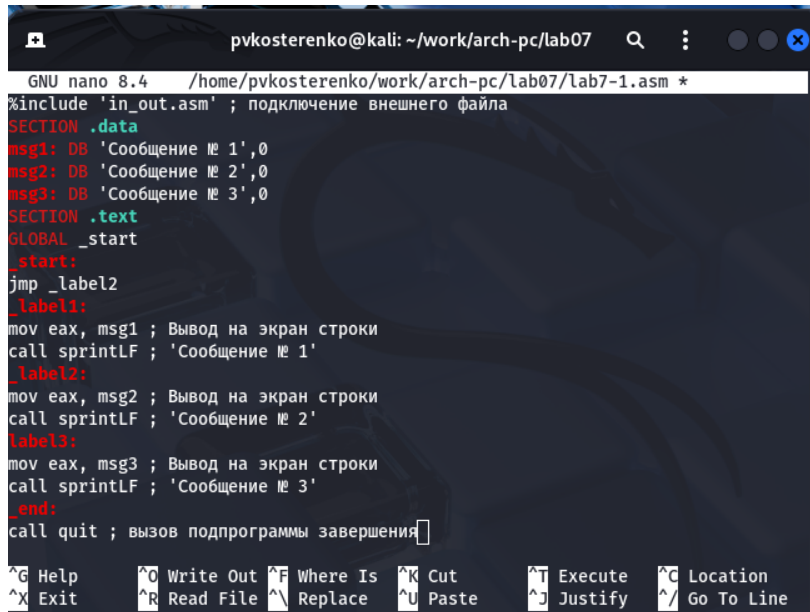
Создаем каталог для программ ЛБ7, и в нем создаем файл (рис. Рисунок 2.1).



```
pvkosterenko@kali: ~/work/arch-pc/lab07
(pvkosterenko@kali)-[~]
$ mkdir -p ~/work/arch-pc/lab07
(pvkosterenko@kali)-[~]
$ cd ~/work/arch-pc/lab07
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ touch lab7-1.asm
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$
```

Рисунок 2.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

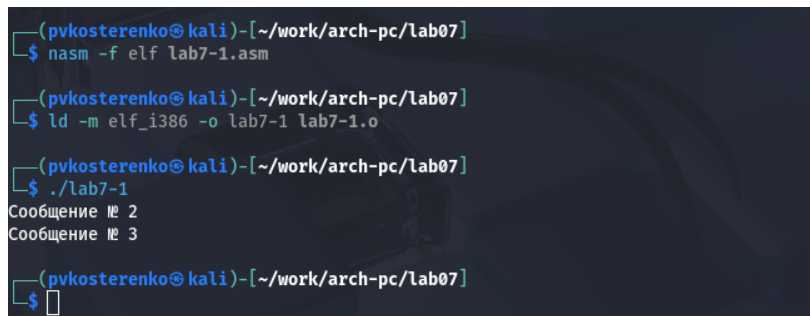
Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.1 (рис. Рисунок 2.2).



```
GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab07/lab7-1.asm *
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рисунок 2.2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. Рисунок 2.3).



```
(pvkosterenko@kali)~[~/work/arch-pc/lab07]
$ nasm -f elf lab7-1.asm

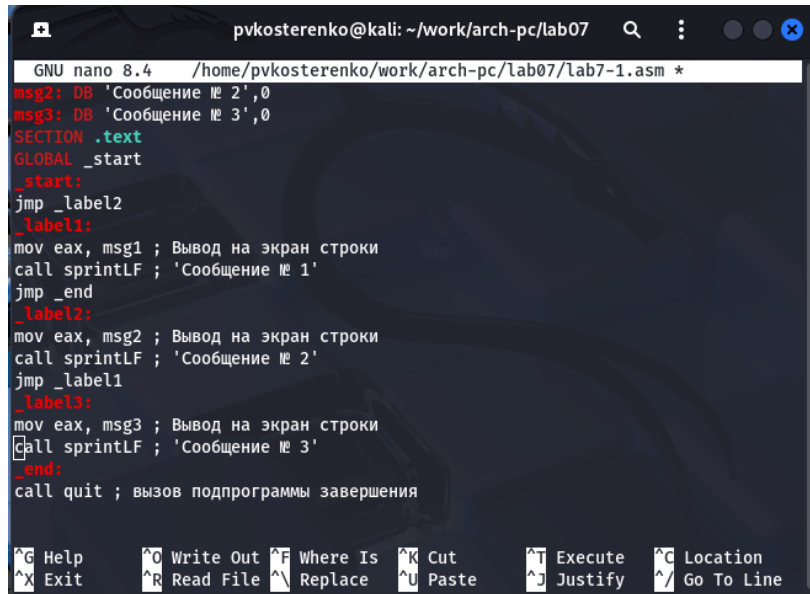
(pvkosterenko@kali)~[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o lab7-1 lab7-1.o

(pvkosterenko@kali)~[~/work/arch-pc/lab07]
$ ./lab7-1
Сообщение № 2
Сообщение № 3

(pvkosterenko@kali)~[~/work/arch-pc/lab07]
$
```

Рисунок 2.3: Запускаем файл и смотрим на его работу

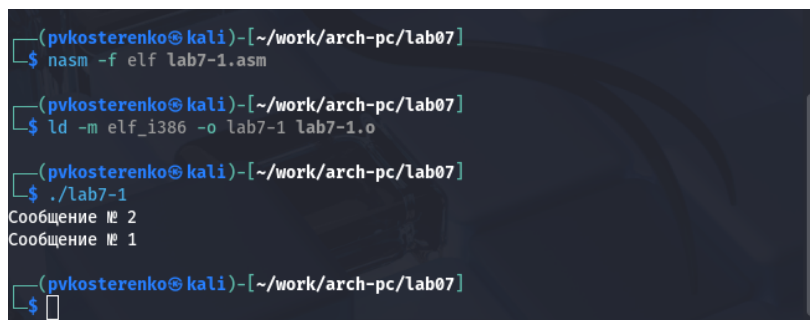
Снова открываем файл для редактирования и изменяем его в соответствии с листингом 7.2 (рис. Рисунок 2.4).



```
GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab07/lab7-1.asm *
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рисунок 2.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. Рисунок 2.5).



```
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ nasm -f elf lab7-1.asm

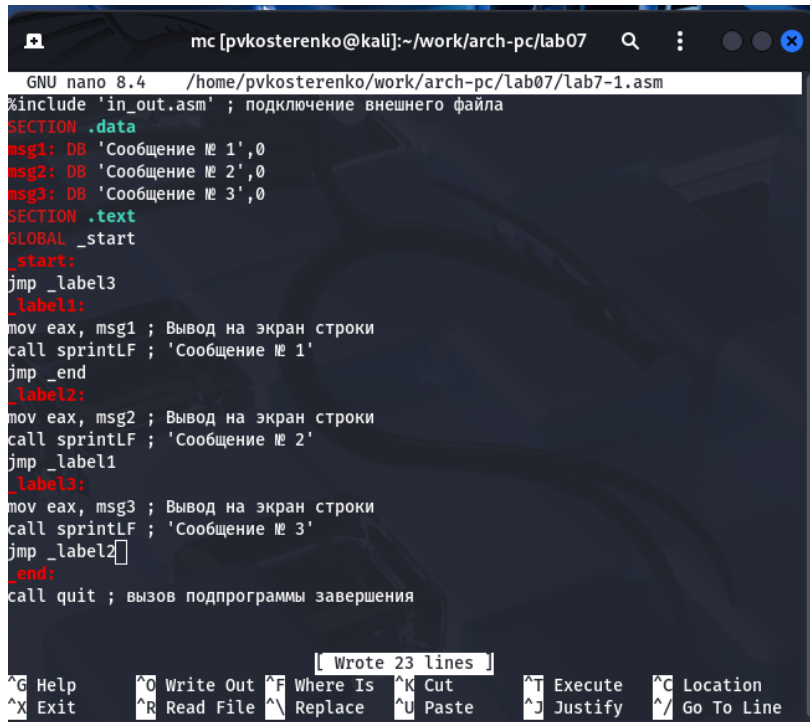
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o lab7-1 lab7-1.o

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ./lab7-1
Сообщение № 2
Сообщение № 1

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$
```

Рисунок 2.5: Запускаем файл и смотрим на его работу

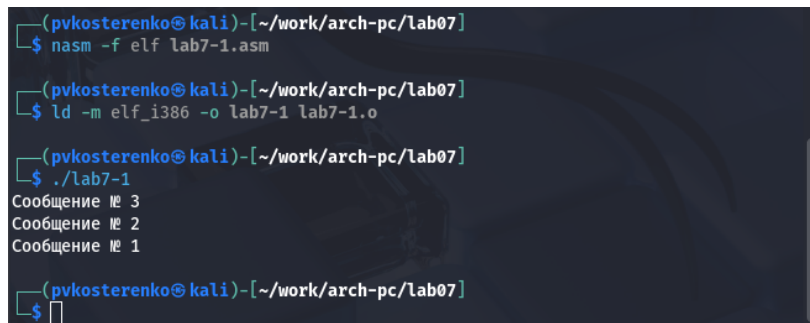
Снова открываем файл для редактирования и изменяем его, чтобы произошел данный вывод (рис. Рисунок 2.6).



```
mc [pvkosterenko@kali]:~/work/arch-pc/lab07
GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab07/lab7-1.asm
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
[Wrote 23 lines]
^G Help      ^O Write Out  ^F Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^N Replace    ^U Paste       ^J Justify    ^_ Go To Line
```

Рисунок 2.6: Редактируем файл

Создаем исполняемый файл и запускаем его (рис. Рисунок 2.7).



```
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ nasm -f elf lab7-1.asm

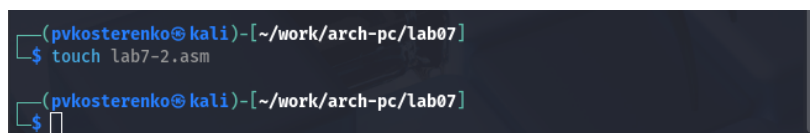
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o lab7-1 lab7-1.o

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$
```

Рисунок 2.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

Создаем новый файл (рис. Рисунок 2.8).

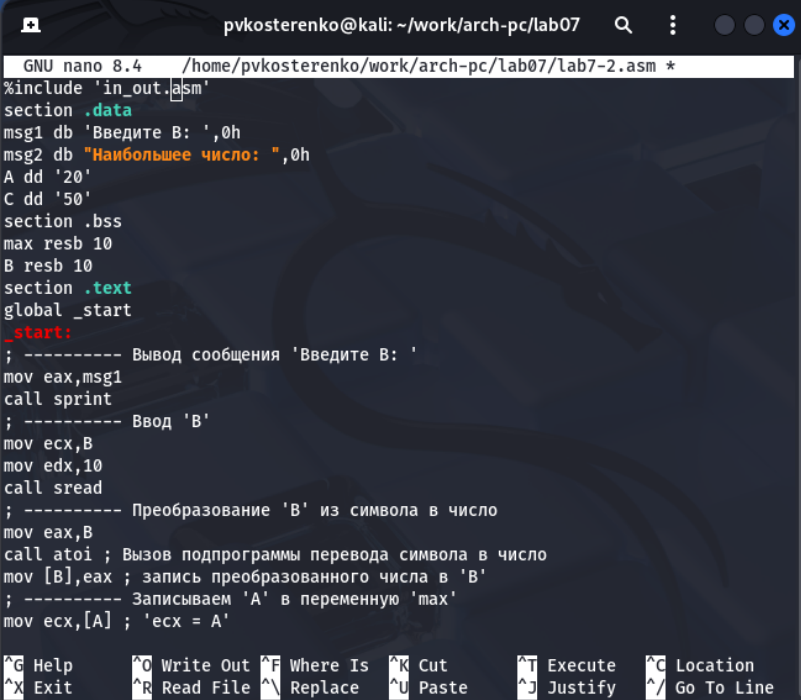


```
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ touch lab7-2.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$
```

Рисунок 2.8: Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.3 (рис. Рисунок 2.9).



```
GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab07/lab7-2.asm *
#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
```

Рисунок 2.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, вводя разные значения B (рис. Рисунок 2.10).

```
pvkosterenko@kali: ~/work/arch-pc/lab07
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ./lab7-2
Введите В: 1
Наибольшее число: 50

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ nasm -f elf lab7-2.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o lab7-2 lab7-2.o

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ./lab7-2
Введите В: 3
Наибольшее число: 50

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ nasm -f elf lab7-2.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o lab7-2 lab7-2.o

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ./lab7-2
Введите В: 5
Наибольшее число: 50

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$
```

Рисунок 2.10: Смотрим на работу программ

2.2 Изучение структуры файлы листинга

Создаем файл листинга для программы lab7-2.asm (рис. Рисунок 2.11).

```
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ nasm -f elf -l lab7-2.lst lab7-2.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$
```

Рисунок 2.11: Создаем файл листинга

Открываем файл листинга с помощью команды `mcedit` и изучаем его (рис. Рисунок 2.12).

Рисунок 2.12: Изучаем файл

Строка 33: 0000001D-адрес в сегменте кода, BB01000000-машинный код, mov ebx,1-присвоение переменной ebx значения 1.

Строка 34: 00000022-адрес в сегменте кода, B804000000-машинный код, mov eax,4-присвоение переменной eax значения 4.

Строка 35 00000027-адрес в сегменте кода, CD80-машинный код, int 80h-вызов ядра.

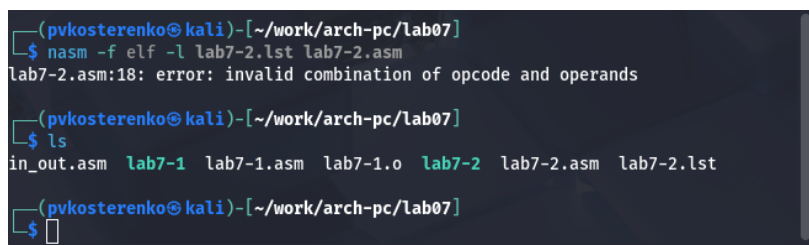
Открываем файл и удаляем один операндум (рис. Рисунок 2.13).



```
GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab07/lab7-2.asm
#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
[ Read 40 lines ]
^G Help      ^O Write Out ^F Where Is   ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace    ^U Paste     ^J Justify   ^_ Go To Line
```

Рисунок 2.13: Удаляем операндум из файла

Транслируем с получением файла листинга (рис. Рисунок 2.14).



```
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
└─$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:18: error: invalid combination of opcode and operands

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
└─$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
└─$
```

Рисунок 2.14: Транслируем файл

При трансляции файла, выдается ошибка, но создаются исполнительный файл lab7-2 и lab7-2.lst

Снова открываем файл листинга и изучаем его (рис. Рисунок 2.15).

Рисунок 2.15: Изучаем файл с ошибкой

2.3 Задание для самостоятельной работы

ВАРИАНТ-17

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.

Создаем новый файл (рис. Рисунок 2.16).

```
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ touch lab7-3.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$
```

Рисунок 2.16: Создаем файл командой touch

Открываем его и пишем программу, которая выберет наименьшее число из трех(2 числа уже в программе, 3-е вводится из консоли) (рис. Рисунок 2.17).

```
GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab07/lab7-3.asm
#include 'in_out.asm'
section .data
    msg1 db 'Введите B: ',0h
    msg2 db 'Наименьшее число: ',0h
    A dd 28
    C dd 68
section .bss
    min resd 1
    B resd 1
section .text
    global _start
_start:
    ; Ввод значения B
    mov eax, msg1
    call sprint
    mov ecx, B
    mov edx, 10
    call sread

    ; Преобразование B в число
    mov eax, B
    call atoi
    mov [B], eax

    ; Сравнение A и C, находим минимум
    mov ecx, [A]
    mov [min], ecx
    cmp ecx, [C]
    jl check_B
    mov ecx, [C]
    mov [min], ecx

check_B:
    ; Сравниваем текущий минимум с B
    mov ecx, [min]
    cmp ecx, [B]
    jl fin
    mov ecx, [B]
    mov [min], ecx
```

Рисунок 2.17: Пишем программу

Транслируем файл и смотрим на работу программы (рис. Рисунок 2.18).

```
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ nasm -f elf lab7-3.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o lab7-3 lab7-3.o

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ./lab7-3
Введите B: 12
```

Рисунок 2.18: Смотрим на работу программы(всё верно)

2. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.

Создаем новый файл (рис. Рисунок 2.19).

```
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ touch lab7-4.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$
```

Рисунок 2.19: Создаем файл командой touch

Открываем его и пишем программу, которая решит систему уравнений, при данных, введенных в консоль (рис. Рисунок 2.20).


```
pvkosterenko@kali: ~/work/arch-pc/lab07
GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab07/lab7-4.asm *
%include 'in_out.asm'
SECTION .data
    msg1: DB 'Введите x: ',0
    msg2: DB 'Введите a: ',0
    otv: DB 'F(a) = ',0
SECTION .bss
    x: RESB 80
    a: RESB 80
    res: RESD 1
SECTION .text
    GLOBAL _start
_start:
    mov eax, msg1
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    mov [x], eax

    mov eax, msg2
    call sprint
    mov ecx, a
    mov edx, 80
    call sread
    mov eax, a
    call atoi
    mov [a], eax

    mov eax, [a]
    cmp eax, 8
    jge greater_equal
less:
    add eax, 8
    mov [res], eax
    jmp fin
greater_equal:
    mov eax, [a]
    mov ebx, [x]
    imul eax, ebx
```

Рисунок 2.20: Пишем программу

Транслируем файл и проверяем его работу при $x=3$ и $a=4$ (рис. Рисунок 2.21).

```
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ nasm -f elf lab7-4.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o lab7-4 lab7-4.o

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ./lab7-4
Введите x: 3
Введите a: 4
F(a) = 12
```

Рисунок 2.21: Проверяем работу программы

Транслируем файл и проверяем его работу при $x=2$ и $a=9$ (рис. Рисунок 2.22).

```
(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ nasm -f elf lab7-4.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ld -m elf_i386 -o lab7-4 lab7-4.o

(pvkosterenko@kali)-[~/work/arch-pc/lab07]
$ ./lab7-4
Введите x: 2
Введите a: 9
F(a) = 18
```

Рисунок 2.22: Проверяем работу программы

3 Выводы

Мы познакомились с структурой файла листинга, изучили команды условного и безусловного перехода.