

# **Отчёт по лабораторной работе №7**

**Дисциплина: архитектура компьютера.**

Наговицын Арсений Владимирович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Реализация переходов в NASM . . . . .	7
3.2	Изучение структуры файла листинга . . . . .	14
3.3	Выполнение заданий для самостоятельной работы . . . . .	16
<b>4</b>	<b>Выводы</b>	<b>23</b>

## Список иллюстраций

3.1	Создание каталога и файла . . . . .	7
3.2	Редактирование файла . . . . .	7
3.3	Компиляция и запуск файла . . . . .	8
3.4	Редактирование файла . . . . .	9
3.5	Компиляция и запуск файла . . . . .	9
3.6	Редактирование файла . . . . .	10
3.7	Компиляция и запуск файла . . . . .	11
3.8	Создание файла . . . . .	12
3.9	Редактирование файла . . . . .	12
3.10	Компиляция и запуск файла . . . . .	14
3.11	Создание файла . . . . .	14
3.12	Просмотр файла . . . . .	15
3.13	Редактирование файла . . . . .	16
3.14	Трансляция файла . . . . .	16
3.15	Создание файла . . . . .	16
3.16	Редактирование файла . . . . .	17
3.17	Компиляция и запуск файла . . . . .	17
3.18	Компиляция и запуск файла . . . . .	19
3.19	Редактирование файла . . . . .	19
3.20	Компиляция и запуск файла . . . . .	20

## **Список таблиц**

# 1 Цель работы

Целью данной лабораторной работы является изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием переходов и знакомство с назначением и структурой файла листинга.

## 2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга.
3. Выполнение заданий для самостоятельной работы

## 3 Выполнение лабораторной работы

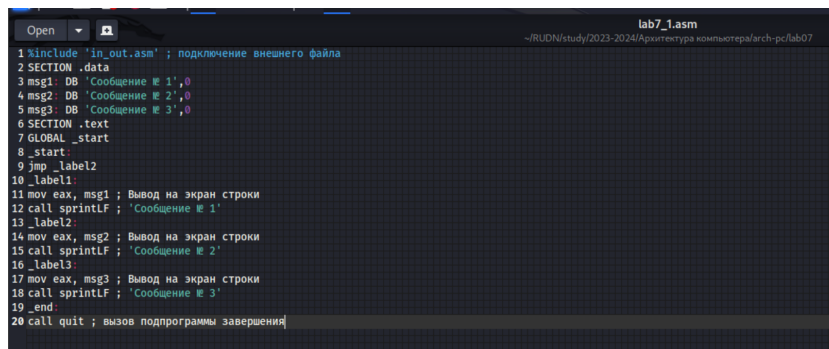
### 3.1 Реализация переходов в NASM

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

```
(avnagovicihn@avnagovicihn)-[~/../study/2023-2024/Архитектура компьютера/arch-pc]
$ mkdir lab07
(avnagovicihn@avnagovicihn)-[~/../study/2023-2024/Архитектура компьютера/arch-pc]
$ cd lab07
(avnagovicihn@avnagovicihn)-[~/../2023-2024/Архитектура компьютера/arch-pc/lab07]
$ touch lab7_1.asm
```

Рис. 3.1: Создание каталога и файла

Ввожу в файл текст программы из листинга 7.1(рис. 3.2).



```
lab7_1.asm
~/BUON/study/2023-2024/Архитектура компьютера/arch-pc/lab07

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1 DB 'Сообщение № 1',0
4 msg2 DB 'Сообщение № 2',0
5 msg3 DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintf ; 'Сообщение № 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintf ; 'Сообщение № 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис. 3.2: Редактирование файла

#### Листинг 7.1 Программа с использованием инструкции jmp

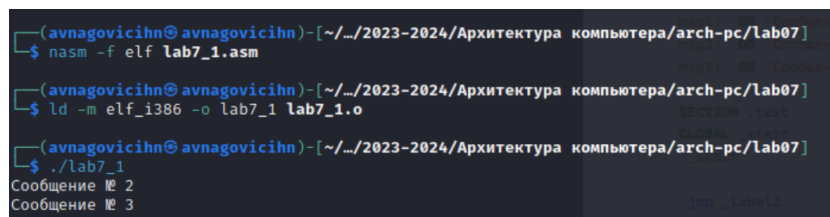
```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
```

```

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 sprintfLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

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



```

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ nasm -f elf lab7_1.asm

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ ld -m elf_i386 -o lab7_1 lab7_1.o

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ ./lab7_1
Сообщение № 2
Сообщение № 3

```

Рис. 3.3: Компиляция и запуск файла

Изменяю текст программы в соответствии с листингом 7.2 (рис. 3.4)



```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call printf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call printf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call printf ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения

```

Рис. 3.4: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 3.5).

```

(avnagovicihn@avnagovicihn)-[~/../2023-2024/Архитектура компьютера/arch-pc/lab07]
$ nasm -f elf lab7_1.asm

(avnagovicihn@avnagovicihn)-[~/../2023-2024/Архитектура компьютера/arch-pc/lab07]
$ ld -m elf_i386 -o lab7_1 lab7_1.o

(avnagovicihn@avnagovicihn)-[~/../2023-2024/Архитектура компьютера/arch-pc/lab07]
$ ./lab7_1
Сообщение № 2
Сообщение № 1

```

Рис. 3.5: Компиляция и запуск файла

## Листинг 7.2 Программа с использованием инструкции jmp

```

#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

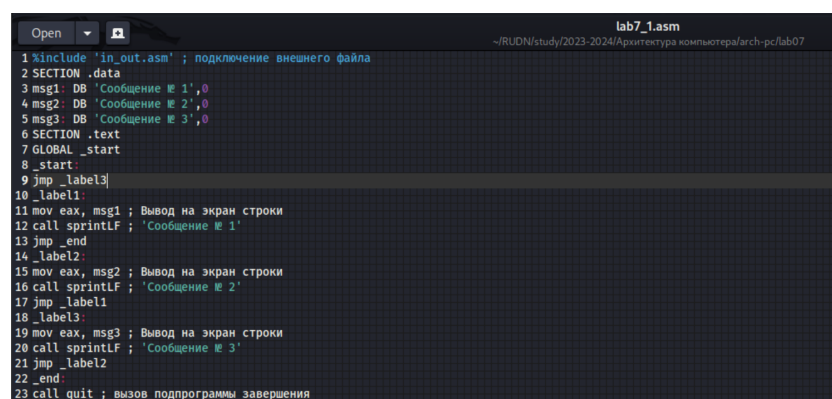
```

```

call sprintfLF ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Изменяю текст программы так, чтобы программа сначала выводила ‘Сообщение № 3’, затем ‘Сообщение № 2’, а затем ‘Сообщение № 1’ (рис. 3.6)



```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1 DB 'Сообщение № 1',0
4 msg2 DB 'Сообщение № 2',0
5 msg3 DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintfLF ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintfLF ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintfLF ; 'Сообщение № 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения

```

Рис. 3.6: Редактирование файла

### Листинг 7.3 Программа с использованием инструкции jmp

```

#include 'in_out.asm' ; подключение внешнего файла

SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0

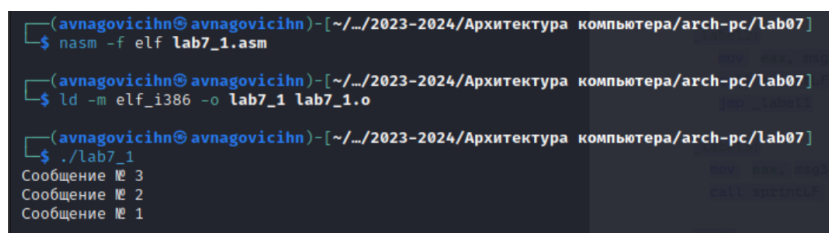
```

```

msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения

```

Создаю исполняемый файл и запускаю его (рис. 3.7).



```

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ nasm -f elf lab7_1.asm

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ ld -m elf_i386 -o lab7_1 lab7_1.o

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ ./lab7_1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 3.7: Компиляция и запуск файла

Создаю новый файл lab7\_2.asm (рис. 3.8).

```
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ touch lab7_2.asm

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ gedit lab7_2.asm
```

Рис. 3.8: Создание файла

Изменяю текст программы в соответствии с листингом 7.4 (рис. 3.9)



```
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
```

Рис. 3.9: Редактирование файла

### Листинг 7.4 Программа с использованием инструкции jmp

```
%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'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
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: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```

Создаю исполняемый файл, запускаю его и проверяю на нескольких значениях (рис. 3.10).

```

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ nasm -f elf lab7_2.asm

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ ld -m elf_i386 -o lab7_2 lab7_2.o

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ ./lab7_2
Введите B: 3
Наибольшее число: 50

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ ./lab7_2
Введите B: 55
Наибольшее число: 55

```

Рис. 3.10: Компиляция и запуск файла

## 3.2 Изучение структуры файла листинга

Создаю файл листинга для программы из файла lab7-2.asm (рис. 3.11).

```

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ nasm -f elf -l lab7_2.lst lab7_2.asm

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ mcedit lab7_2.lst

```

Рис. 3.11: Создание файла

Открываю файл листинга (рис. 3.12).

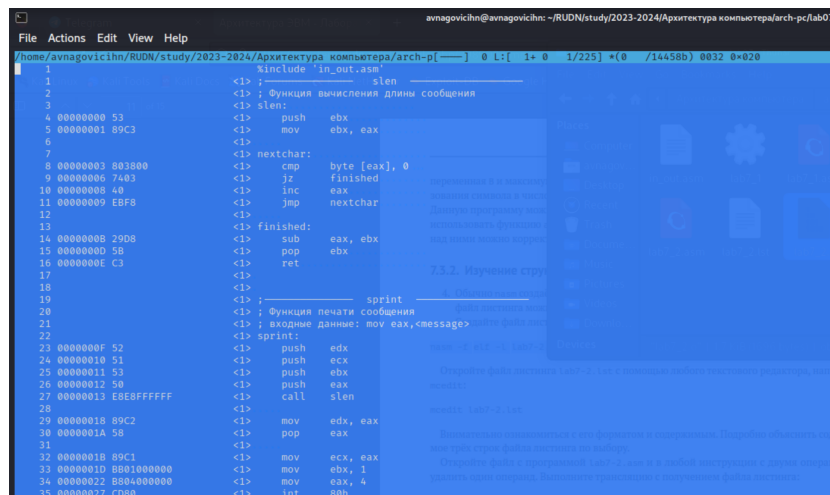


Рис. 3.12: Просмотр файла

### Объясняю содержимое первой выбранной строки:

4 00000000 53 **push ebx**

4 - номер строки; 00000000 - адрес строки (смещение машинного кода от начала текущего сегмента); “53” - машинный код; “push ebx” - исходный текст программы. Инструкция “push” помещает операнд “ebx” в стек.

### Объясняю содержимое второй выбранной строки:

29 00000018 89C2 **mov edx, eax**

29 - номер строки; 00000018 - адрес строки (смещение машинного кода от начала текущего сегмента); “89C2” - машинный код; mov edx,eax - исходный текст программы. Инструкция на машинном языке, записывающая значение переменной eax в регистр edx

### Объясняю содержимое третьей выбранной строки:

20 **Функция печати сообщения**

20 - номер строки; Функция печати сообщения - комментарий к коду, не имеет адреса, машинного кода.

Открываю файл с программой и удаляю операнд (рис. 3.13).

```
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx, [max]
39 cmp ecx, [B]; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx, [B] ; иначе 'ecx = B'
42 mov [max], ecx
```

Рис. 3.13: Редактирование файла

Выполняю трансляцию файла (рис. 3.14).

```
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ nasm -f elf -l lab7.2.lst lab7.2.asm
lab7_2.asm:39: error: invalid combination of opcode and operands
```

Рис. 3.14: Трансляция файла

На выходе я не получаю никаких файлов, так как инструкция `cmp` подразумевает сравнение двух операндов.

### 3.3 Выполнение заданий для самостоятельной работы

1. Создаю новый файл `lab7_3.asm` (рис. 3.15).

```
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab07]
$ touch lab7_3.asm
```

Рис. 3.15: Создание файла

Открываю файл и пишу программу. Присваиваю переменным значения, указанные в 12 варианте (99,29,26) (рис. 3.16).



```

1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Наименьшее число: ",0h
4 A dd '99'
5 B dd '29'
6 C dd '26'
7
8 SECTION .bss
9 min resb 10
10
11 SECTION .text
12 global _start
13 _start:
14 ; ----- Записываем 'A'
15 mov ecx,[A] ; 'ecx = A'
16 mov [min],ecx ; 'min = A'
17 ; ----- Сравниваем 'A' и 'C' (как символы)
18 cmp ecx,[C] ; Сравниваем 'A' и 'C'
19 jl check_B ; если 'A < C', то переход на метку 'check_B',
20 mov ecx,[C] ; иначе 'ecx = C'
21 mov [min],ecx ; 'min = C'
22 ; ----- Преобразование 'min(A,C)' из символа в число
23 check_B:
24 mov eax,min
25 call atoi ; Вызов подпрограммы перевода символа в число
26 mov [min],eax ; запись преобразованного числа в min
27 ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
28 mov ecx,[min]
29 cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
30 jl fin ; если 'min(A,C) < B', то переход на 'fin',
31 mov ecx,[B] ; иначе 'ecx = B'
32 mov [min],ecx
33 ; ----- Вывод результата
34 fin:
35 mov eax, msg

```

Рис. 3.16: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 3.17).

```

(avnagovicihn@avnagovicihn)~/2023-2024/Архитектура компьютера/arch-pc/lab07
$ nasm -f elf lab7_3.asm

(avnagovicihn@avnagovicihn)~/2023-2024/Архитектура компьютера/arch-pc/lab07
$ ld -m elf_i386 -o lab7_3 lab7_3.o

(avnagovicihn@avnagovicihn)~/2023-2024/Архитектура компьютера/arch-pc/lab07
$ ./lab7_3
Наименьшее число: 26

```

Рис. 3.17: Компиляция и запуск файла

## Листинг 7.5. Программа для нахождения наименьшего числа

```

#include 'in_out.asm'

SECTION .data
msg db "Наименьшее число: ",0h
A dd '99'
B dd '29'
C dd '26'

SECTION .bss
min resb 10

```

```

SECTION .text
global _start
_start:
; ----- Записываем 'A'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jl check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в min
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jl fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'

```

`call quit ; Выход`

2. Создаю новый файл lab7\_4.asm (рис. 3.18).

```
(avnagovicihn@avnagovicihn)-[~/../2023-2024/Архитектура компьютера/arch-pc/lab07]
$ touch lab7_4.asm
(avnagovicihn@avnagovicihn)-[~/../2023-2024/Архитектура компьютера/arch-pc/lab07]
$ gedit lab7_4.asm
```

Рис. 3.18: Компиляция и запуск файла

Открываю файл и пишу программу (рис. 3.19).

```
Open  lab7_4.asm
~/RUON/study/2023-2024/Архитектура компьютера/arch-pc/lab07

18 ; Ввод 'x'
19 mov ecx, x
20 mov edx, 10
21 call sread
22 ;
23 mov eax, msg2
24 call sprint
25 ; Ввод 'a'
26 mov ecx, a
27 mov edx, 10
28 call sread
29 ; Преобразование 'x' из символа в число
30 mov eax, x
31 call atoi ; Вызов подпрограммы перевода символа в число
32 mov [x], eax ; запись преобразованного числа в 'x'
33 ; Преобразование 'a' из символа в число
34 mov eax, a
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [a], eax ; запись преобразованного числа в 'a'
37 ; Сравниваем 'x' и '5' (как числа)
38 mov eax, [a]
39 mov ecx, [x]
40 cmp ecx, 5 ; Сравниваем 'x' и '5'
41 jl less_xa ; если 'x<5', то переход на метку 'less_xa'
42 add ecx, -5;
43 mov [res], ecx ; 'res = x-5'
44 jmp _res
45 ; Записываем 'a*x' в переменную 'res'
46 less_xa:
47 mov eax, [a]
48 mul ecx ;
49 mov [res], eax
50 jmp _res
51 ; Вывод результата
52 _res:
53 mov eax, msg3
54 call sprint ; Вывод сообщения 'Результат: '
55 mov eax, [res]
56 call iprintf ; Вывод
57 call quit ; Вызов подпрограммы завершения
```

Рис. 3.19: Редактирование файла

Создаю исполняемый файл и запускаю его. Ввожу значения в соответствии со своим вариантом ((3;7),(6;4)) (рис. 3.20).

```
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-пс/lab07]
$ nasm -f elf lab7_4.asm

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-пс/lab07]
$ ld -m elf_i386 -o lab7_4 lab7_4.o

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-пс/lab07]
$ ./lab7_4
Введите x: 3
Введите a: 7
Результат: 21

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-пс/lab07]
$ ./lab7_4
Введите x: 6
Введите a: 4
Результат: 1 три день (ROMAN ROMANOV)
```

Рис. 3.20: Компиляция и запуск файла

### Листинг 7.6 Программа для вычисления заданной функции

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите x: ', 0h
msg2 db 'Введите a: ', 0h
msg3 db 'Результат: ', 0h

SECTION .bss
x resb 11
a resb 11
res resb 12

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

```

mov edx, 10
call sread
; ----- Вывод сообщения 'Введите a: '
mov eax, msg2
call sprint
; ----- Ввод 'a'
mov ecx, a
mov edx, 10
call sread
; ----- Преобразование 'x' из символа в число
mov eax, x
call atoi ; Вызов подпрограммы перевода символа в число
mov [x], eax ; запись преобразованного числа в 'x'
; ----- Преобразование 'a' из символа в число
mov eax, a
call atoi ; Вызов подпрограммы перевода символа в число
mov [a], eax ; запись преобразованного числа в 'a'
; ----- Сравниваем 'x' и '5' (как числа)
mov eax, [a]
mov ecx, [x]
cmp ecx, 5 ; Сравниваем 'x' и '5'
jl less_xa ; если 'x<5', то переход на метку 'less_xa'
add ecx, -5;
mov [res], ecx ; 'res = x-5'
jmp _res
; ----- Записываем 'a*x' в переменную 'res'
less_xa:
mov eax, [a]
mul ecx ;

```

```
mov [res], eax
; jmp _res
; ----- Вывод результата
_res:
mov eax, msg3
call sprint ; Вывод сообщения 'Результат: '
mov eax, [res]
call iprintLF ; Вывод
call quit ; Вызов подпрограммы завершения
```

## 4 Выводы

В ходе выполнения данной лабораторной работы я изучил команды условного и безусловного переходов, приобрёл навыки написания программ с использованием переходов и познакомился с назначением и структурой файла листинга.