

# **Отчёт по лабораторной работе №7**

**Дисциплина: Архитектура компьютера**

Пономарева Татьяна Александровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</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 файлы листинга . . . . .	11
<b>4</b>	<b>Задания для самостоятельной работы</b>	<b>14</b>
<b>5</b>	<b>Выводы</b>	<b>19</b>
	<b>Список литературы</b>	<b>20</b>

# Список иллюстраций

3.1	Терминал. Создание каталога lab07. Переход в каталог lab07. Создание файла lab7-1.asm . . . . .	7
3.2	Окно Text Editor. Содержание файла lab7-1.asm . . . . .	7
3.3	Терминал. Создание исполняемого файла lab7-1 и его запуск . . .	8
3.4	Окно Text Editor. Содержание измененного файла lab7-1.asm . . . .	8
3.5	Терминал. Создание исполняемого файла lab7-1 и его запуск . . .	8
3.6	Окно Text Editor. Содержание измененного файла lab7-1.asm . . . .	9
3.7	Терминал. Создание исполняемого файла lab7-1 и его запуск . . .	9
3.8	Терминал. Создание файла lab7-2.asm . . . . .	9
3.9	Окно Text Editor. Содержание файла lab7-2.asm . . . . .	10
3.10	Терминал. Создание исполняемого файла lab7-2 и проверка его работы на корректность . . . . .	10
3.11	Терминал. Создание файла листинга lab7-2.lst . . . . .	11
3.12	Содержание листинга lab7-2.lst . . . . .	11
3.13	Окно Text Editor. Содержание файла lab7-2.asm без операнда . . . .	12
3.14	Окно Midnight Commander. Содержание листинга lab7-2.lst . . . .	13
4.1	Терминал. Создание исполняемого файла lab7-4-1 и его запуск . .	16
4.2	Терминал. Создание исполняемого файла lab7-4-2 и его запуск . .	18

## **Список таблиц**

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

## 2 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: • условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия. • безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

## 3 Выполнение лабораторной работы

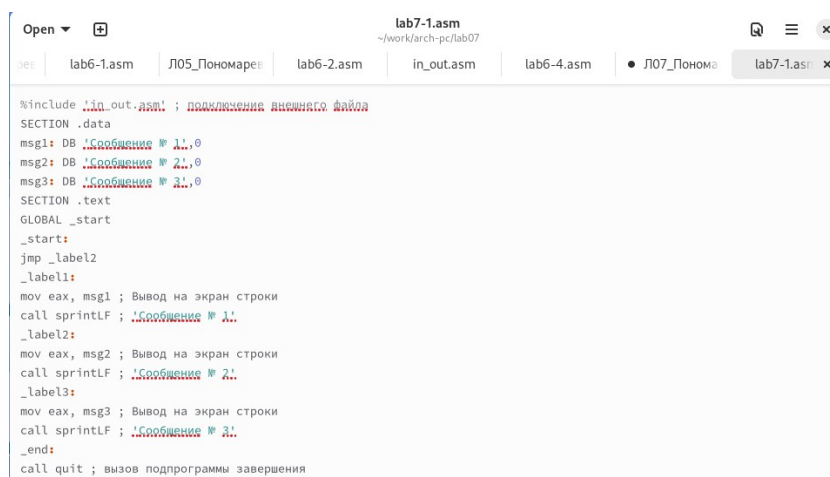
### 3.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm (рис. 3.1).

```
taponomareva@2c7fe9w:~$ mkdir ~/work/arch-pc/lab07
taponomareva@2c7fe9w:~$ cd ~/work/arch-pc/lab07
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 3.1: Терминал. Создание каталога lab07. Переход в каталог lab07. Создание файла lab7-1.asm

Ввожу в файл lab7-1.asm текст программы из листинга 7.1 (рис. 3.2).



```
Open ▾ + lab7-1.asm
~/work/arch-pc/lab07
lab6-1.asm Л05_Пономарева lab6-2.asm in_out.asm lab6-4.asm Л07_Пономарева lab7-1.asm x

%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
```

Рис. 3.2: Окно Text Editor. Содержание файла lab7-1.asm

Создаю исполняемый файл и запускаю его (рис. 3.3). Программа изменяет порядок вывода сообщений при использовании безусловного перехода.

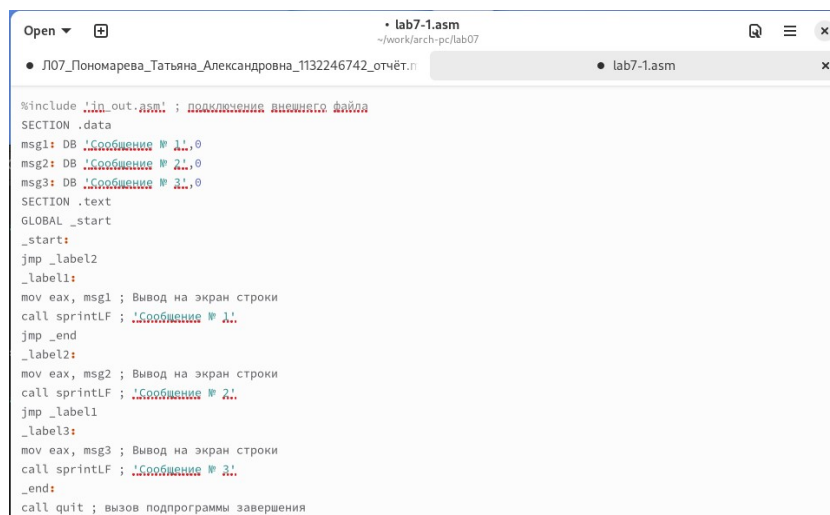
```

taponomareva@2c7fe9w:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
taponomareva@2c7fe9w:~/work/arch-pc/lab07$

```

Рис. 3.3: Терминал. Создание исполняемого файла lab7-1 и его запуск

Изменяю файл lab7-1.asm в соответствии с Листингом 7.2 (рис. 3.4).



```

%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.'
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

```

Рис. 3.4: Окно Text Editor. Содержание измененного файла lab7-1.asm

Создаю исполняемый файл, запускаю его и проверяю его работу (рис. 3.5).

Программа выводит следующее: Сообщение №2, Сообщение №1.

```

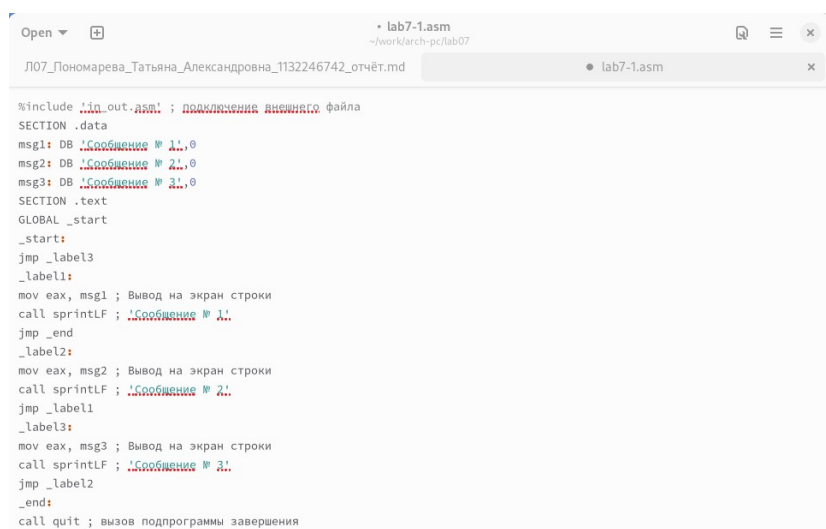
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1

```

Рис. 3.5: Терминал. Создание исполняемого файла lab7-1 и его запуск

Изменяю lab7-1.asm (рис. 3.6).

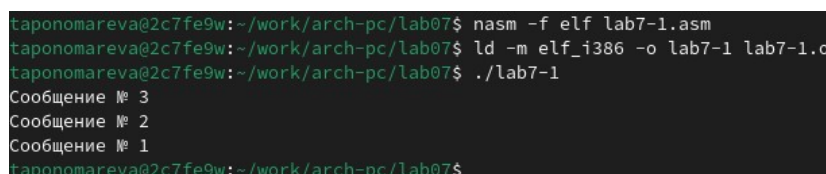




```
%include 'lab_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, 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
```

Рис. 3.6: Окно Text Editor. Содержание измененного файла lab7-1.asm

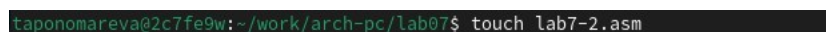
Создаю исполняемый файл, запускаю его и проверяю корректность его работы (рис. 3.7). При помощи безусловного перехода получаем такой порядок вывода: Сообщение №3, Сообщение №2, Сообщение №1.



```
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
taponomareva@2c7fe9w:~/work/arch-pc/lab07$
```

Рис. 3.7: Терминал. Создание исполняемого файла lab7-1 и его запуск

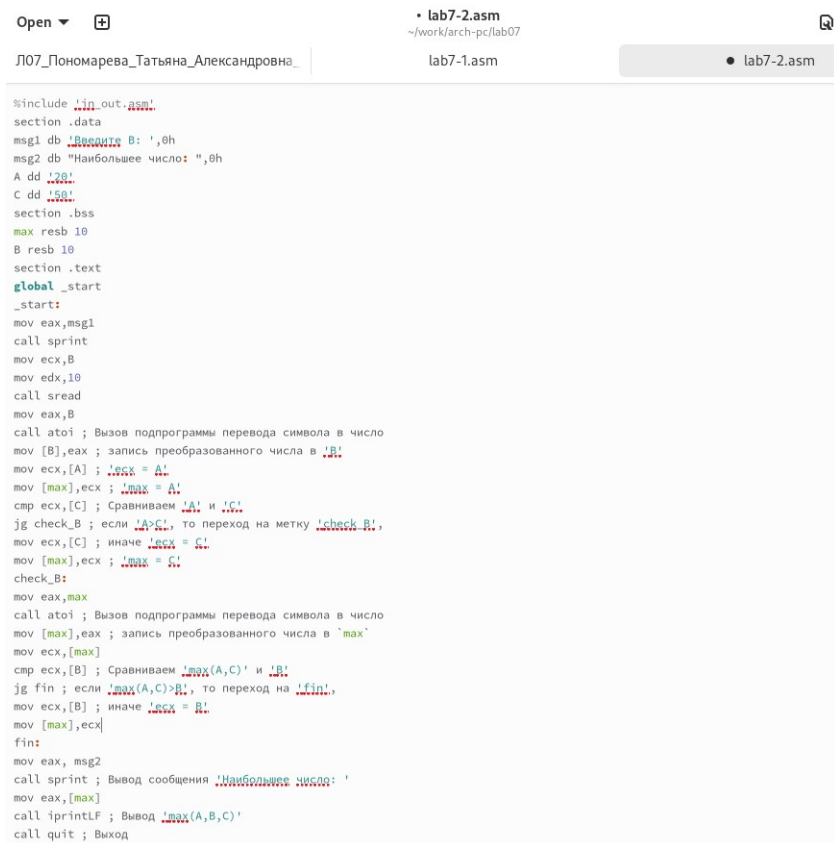
Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 (рис. 3.8).



```
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ touch lab7-2.asm
```

Рис. 3.8: Терминал. Создание файла lab7-2.asm

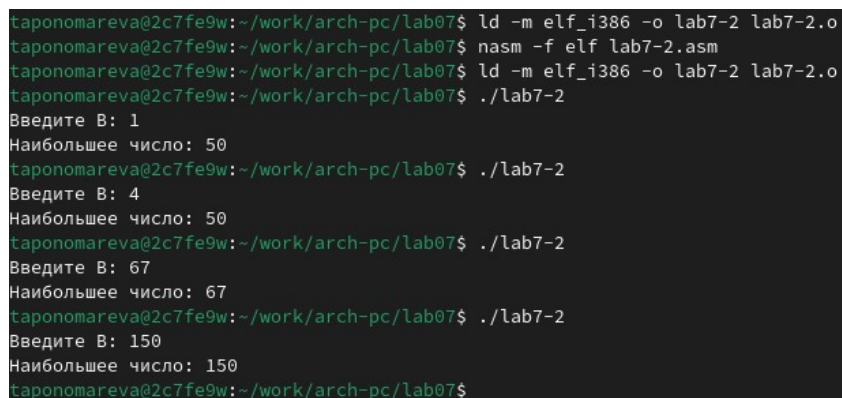
Ввожу текст программы из листинга 7.3 в lab7-2.asm (рис. 3.9).



```
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '28'
C dd '68'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,B
mov edx,10
call sread
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
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'
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
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
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintf ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 3.9: Окно Text Editor. Содержание файла lab7-2.asm

Создаю исполняемый файл lab7-2, запускаю его и проверяю корректность его работы (рис. 3.10). Программа выводит наибольшее число.



```
taronomareva@2c7fe9w:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
taronomareva@2c7fe9w:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
taronomareva@2c7fe9w:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
taronomareva@2c7fe9w:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 1
Наибольшее число: 50
taronomareva@2c7fe9w:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 4
Наибольшее число: 50
taronomareva@2c7fe9w:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 67
Наибольшее число: 67
taronomareva@2c7fe9w:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 150
Наибольшее число: 150
taronomareva@2c7fe9w:~/work/arch-pc/lab07$
```

Рис. 3.10: Терминал. Создание исполняемого файла lab7-2 и проверка его работы на корректность

## 3.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm при помощи команды `nasm -f elf -l lab7-2.lst lab7-2.asm` (рис. 3.11).

```
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 3.11: Терминал. Создание файла листинга lab7-2.lst

Открываю файл листинга lab7-2.lst с помощью `mcedit` (рис. 3.12).

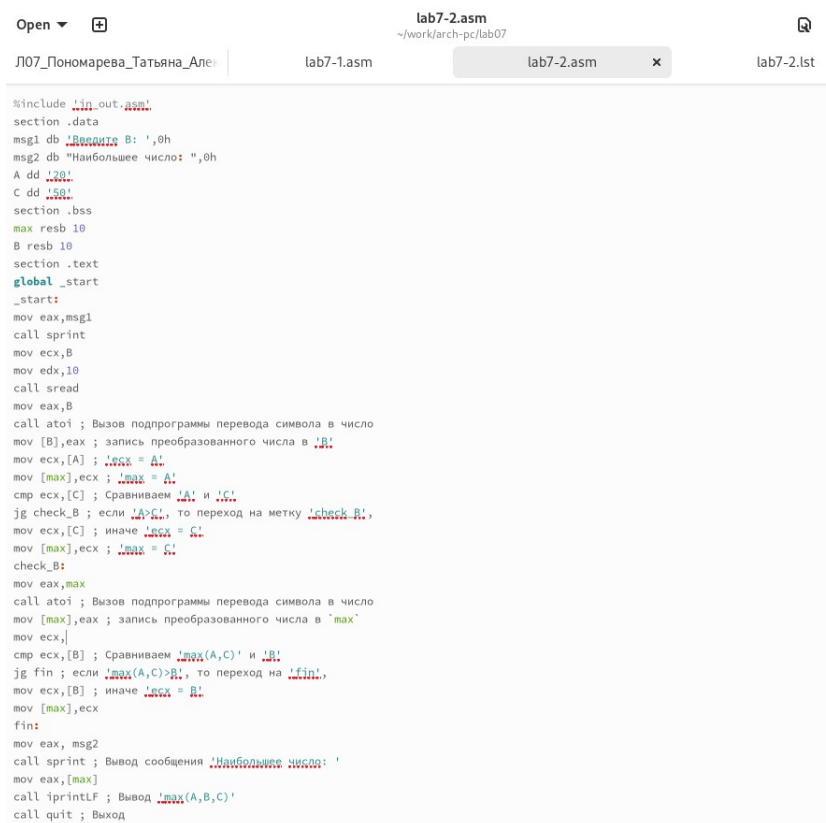
Рис. 3.12: Содержание листинга lab7-2.lst

Объяснение трех строк из листинга lab7-2.lst:

8 00000003 803800	<1>	cmp	byte [eax], 0	- Эта строка проверяет, р
9 00000006 7403	<1>	jz	finished	- Если байт по адресу [ea
10 00000008 40	<1>	inc	eax	- Если байт по адресу [ea

8, 9, 10 - номера строк кода. 00000003, 00000006, 00000008 - смещения инструкций в па-  
битным числовым значением., 38 говорит процессору выполнить команду cmp (сравнение) дл

Открываю файл lab7-2.asm и удаляю один из операндов (рис. 3.13).



```
%include 'in_out.asm'
section .data
msg1 db 'Result B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '10'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,B
mov edx,10
call sread
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
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'
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
mov 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
fin:
mov eax,msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call 'printf' ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 3.13: Окно Text Editor. Содержание файла lab7-2.asm без операнда

Создаю файл листинга для программы из измененного файла lab7-2.asm и  
проверяю его на корректную работу (рис. 3.14).

```
lab7-2.lst [-----] 56 L: [195+21 216/218] *(13625/13712b) 1042 0x412 [*] [X]
20 00000108 A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
21 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
22 00000116 890D[00000000] mov [max],ecx ; 'max = A'
23 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
24 00000122 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
25 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
26 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
27 check_B:
28 00000130 B8[00000000] mov eax,max
29 00000135 E862FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
30 0000013A A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
31 mov ecx,
error: invalid combination of opcode and operands
32 0000013F 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
33 00000145 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
34 00000147 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
35 0000014D 890D[00000000] mov [max],ecx
36 fin:
37 00000153 B8[13000000] mov eax,msg2
38 00000158 E8B2FFFFFF call sprintf ; Вывод сообщения 'Наибольшее число: '
39 0000015D A1[00000000] mov eax,[max]
40 00000162 E81FFFFFFF call printf ; Вывод 'max(A,B,C)'
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Рис. 3.14: Окно Midnight Commander. Содержание листинга lab7-2.lst

Файл листинга lab7-2.lst дает ошибку при транслировании файла. В этом случае никакие выходные файлы не создаются и ничего в листинг не добавляется.

## 4 Задания для самостоятельной работы

- 1) Из лабораторной работы №6 у меня вариант 3. Значения a, b, c: 94, 5, 58  
Код программы, находящей наименьшее из 3 переменных из файла lab7-4-1.asm

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg1 db 'Введите B: ', 0h
```

```
msg2 db 'Наименьшее число: ', 0h
```

```
A dd 94
```

```
C dd 58
```

```
SECTION .bss
```

```
B resd 1 ; Переменная для хранения числа B
```

```
min resd 1 ; Переменная для хранения минимального числа
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
; Вывод сообщения "Введите B: "
```

```
mov eax, msg1
```

```
call sprint
```

```
; Чтение строки и преобразование в число
```

```
mov ecx, B
```

```
mov edx, 10
```

```
call sread
```

```
mov eax, B
```

```
call atoi
```

```
mov [B], eax ; Сохраняем введенное число B
```

```
; Инициализируем min значением A
```

```
mov eax, [A]
```

```
mov [min], eax
```

```
; Сравнение с C
```

```
mov eax, [min]
```

```
cmp eax, [C]
```

```
jg check_B
```

```
mov eax, [C]
```

```
mov [min], eax
```

```
check_B:
```

```
; Сравнение с B
```

```
mov eax, [min]
```

```
cmp eax, [B]
```

```
jb fin
```

```
mov eax, [B]
```

```
mov [min], eax
```

```

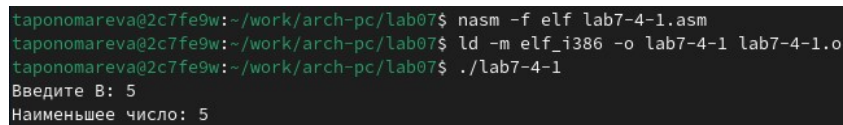
fin:
    ; Вывод сообщения "Наименьшее число: "
    mov eax, msg2
    call sprint

    ; Вывод результата
    mov eax, [min]
    call iprintLF

    ; Завершение программы
    call quit

```

Создаю исполняемый файл lab7-4-1, запускаю его и проверяю корректность его работы (рис. 4.1). Программа выводит наименьшее число.



```

taponomareva@2c7fe9w:~/work/arch-pc/lab07$ nasm -f elf lab7-4-1.asm
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4-1 lab7-4-1.o
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ ./lab7-4-1
Введите В: 5
Наименьшее число: 5

```

Рис. 4.1: Терминал. Создание исполняемого файла lab7-4-1 и его запуск

- 2) Вариант 3. Программа, которая принимает на вход  $x$ , а и выводит значение функции  $f(x)$ .

```

#include 'in_out.asm'

SECTION .data
msgX db 'Enter x: ', 0
msgA db 'Enter a: ', 0
resultMsg db 'f(x) = ', 0

```



```
SECTION .bss
```

```
x resd 1
```

```
a resd 1
```

```
result resd 1
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
    mov eax, msgX
```

```
    call sprint
```

```
    mov ecx, x
```

```
    mov edx, 10
```

```
    call sread
```

```
    mov eax, x
```

```
    call atoi
```

```
    mov [x], eax
```

```
    mov eax, msgA
```

```
    call sprint
```

```
    mov ecx, a
```

```
    mov edx, 10
```

```
    call sread
```

```
    mov eax, a
```

```
    call atoi
```

```
    mov [a], eax
```

```
    mov eax, [x]
```

```
    cmp eax, 3
```

```

je compute_3x

mov eax, [a]
add eax, 1
jmp display_result

```

compute\_3x:

```

mov eax, [x]
imul eax, 3

```

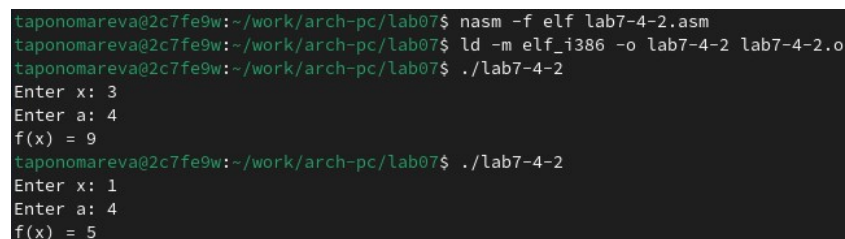
display\_result:

```

mov [result], eax
mov eax, resultMsg
call sprint
mov eax, [result]
call iprintLF
call quit

```

Создаю исполняемый файл lab7-4-2, запускаю его и проверяю корректность его работы (рис. 4.2).



```

taponomareva@2c7fe9w:~/work/arch-pc/lab07$ nasm -f elf lab7-4-2.asm
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4-2 lab7-4-2.o
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ ./lab7-4-2
Enter x: 3
Enter a: 4
f(x) = 9
taponomareva@2c7fe9w:~/work/arch-pc/lab07$ ./lab7-4-2
Enter x: 1
Enter a: 4
f(x) = 5

```

Рис. 4.2: Терминал. Создание исполняемого файла lab7-4-2 и его запуск

Загружаю на GitHub.

## **5 Выводы**

В ходе лабораторной работы были изучены команды условного и безусловного переходов. Были приобретены навыки написания программ с использованием переходов. Также было произведено знакомство с назначением и структурой файла листинга.

## **Список литературы**

1. Курс на ТУИС
2. Лабораторная работа №7