

# **Отчёт по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

**Панявкина Ирина Васильевна**

**НКАбд-04-24**

# Содержание

1 Цель работы.....	3
2 Задание.....	3
3 Теоретическое введение.....	3
4 Выполнение лабораторной работы.....	4
4.1 Реализация переходов в NASM.....	4
4.2 Изучение структуры файла листинга.....	10
4.3    Выполнение заданий для самостоятельной работы.....	13
5 Выводы.....	21
Список литературы.....	22

# 1 Цель работы

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

## 2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Самостоятельное написание программ по материалам лабораторной работы

## 3 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.

## 4 Выполнение лабораторной работы

### 4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7, а также файл lab7-1.asm (рис. 4.1).

```
irina_panyavkina@vbox:~$ mkdir ~/work/arch-pc/lab07
irina_panyavkina@vbox:~$ cd ~/work/arch-pc/lab07
irina_panyavkina@vbox:~/work/arch-pc/lab07$ touch lab7-1.asm
irina_panyavkina@vbox:~/work/arch-pc/lab07$ ls
lab7-1.asm
```

Рис. 4.1: Создание каталога и файла для программы

Копирую в текущий каталог файл in\_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. 4.2).

```
irina_panyavkina@vbox:~/work/arch-pc/lab07$ cp ~/Downloads/in_out.asm in_out.asm
irina_panyavkina@vbox:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1.asm
```

Рис. 4.2: Создание копии файла

Копирую код из листинга в файл будущей программы. (рис. 4.3).

```
*~/work/arch-pc/lab07/lab7-1.asm - Mousepad
File Edit Search View Document Help
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 sprintLF ; 'Сообщение № 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintLF ; 'Сообщение № 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintLF ; 'Сообщение № 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

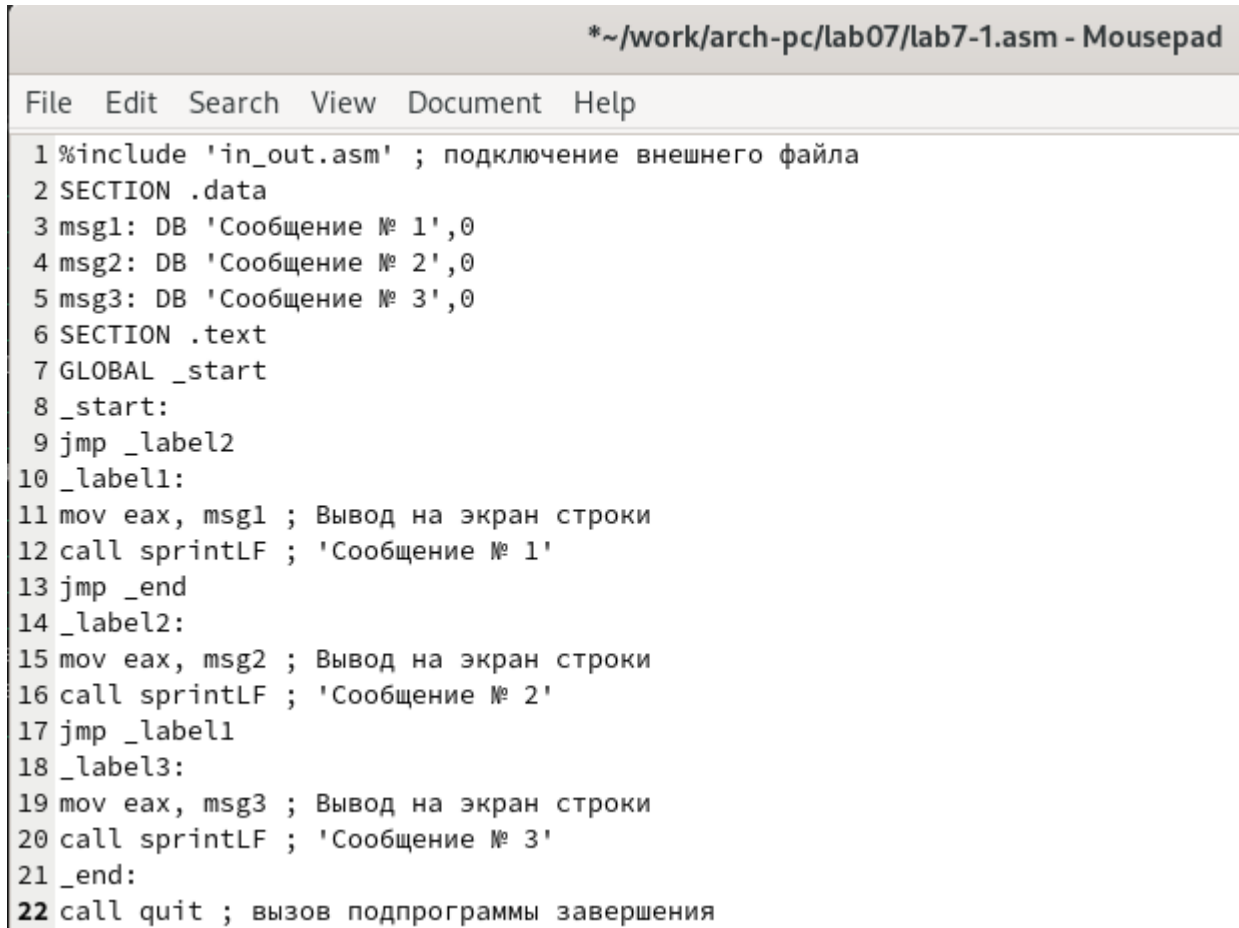
Рис. 4.3: Редактирование файла и сохранение программы

При запуске программы я убедилась в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. 4.4).

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

Рис. 4.4: Запуск исполняемого файла

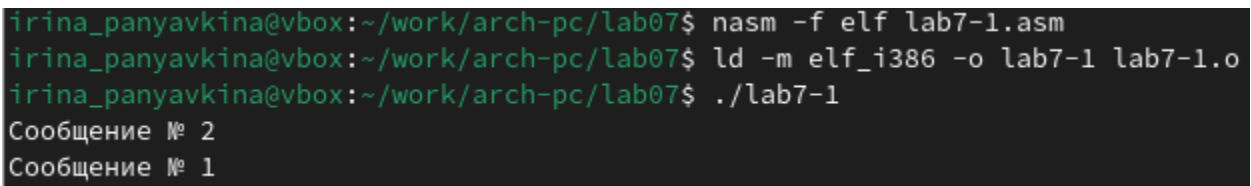
Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. 4.5).



```
*~/work/arch-pc/lab07/lab7-1.asm - Mousepad
File Edit Search View Document Help
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 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения
```

Рис. 4.5: Редактирование файла

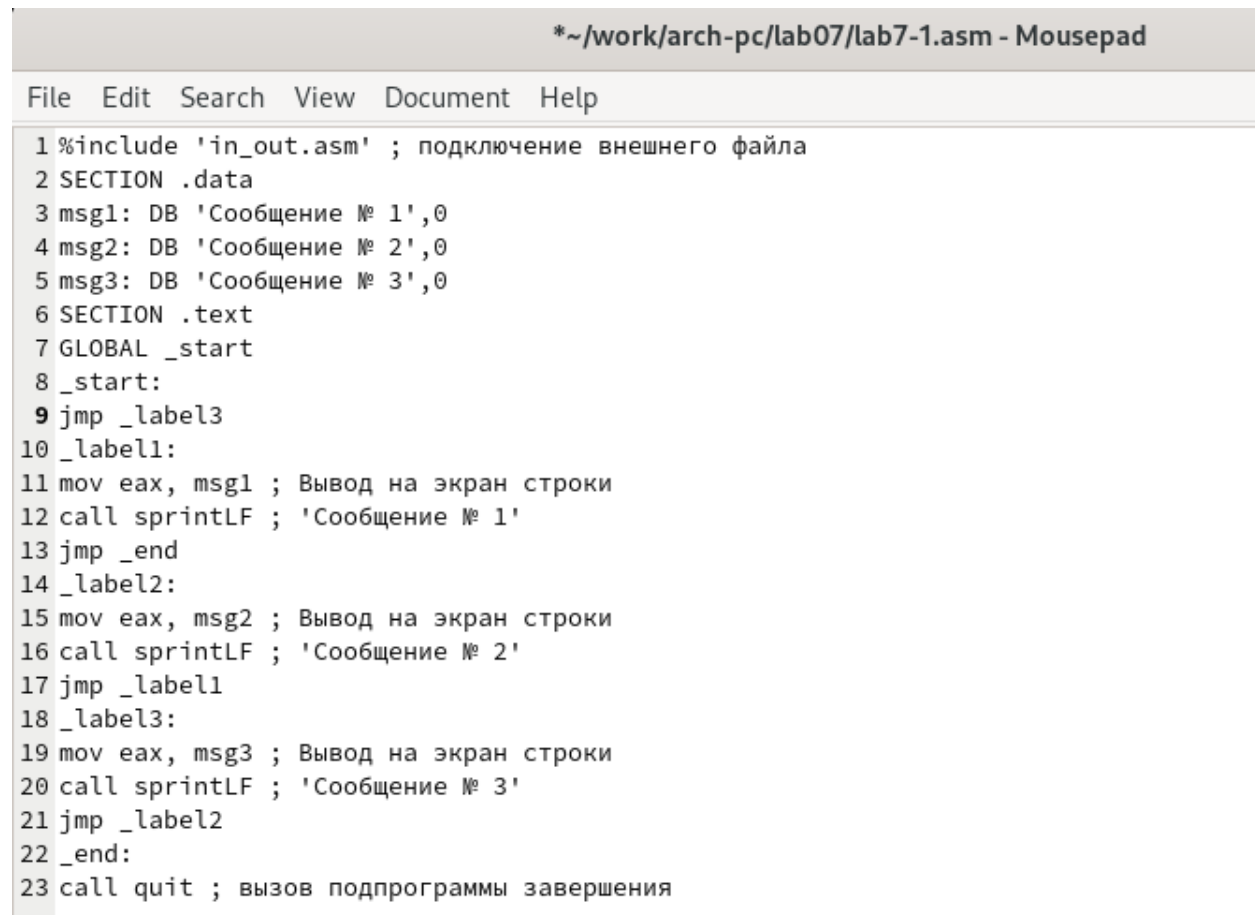
Запускаю программу и проверяю, что применённые изменения верны (рис. 4.6).



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

*Рис. 4.6: Запуск исполняемого файла (изменённой программы)*

Теперь меняю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 4.7).



The screenshot shows a text editor window titled `*~/work/arch-pc/lab07/lab7-1.asm - Mousepad`. The menu bar includes `File`, `Edit`, `Search`, `View`, `Document`, and `Help`. The code is as follows:

```
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. 4.7: Редактирование файла и сохранение программы*

Работа выполнена верно, программа в нужном мне порядке выводит сообщения (рис. 4.8).

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

*Рис. 4.8: Запуск исполняемого файла (измененной программы)*

Создаю новый рабочий файл lab7-2.asm (рис. 4.9).

```
irina_panyavkina@vbox:~/work/arch-pc/lab07$ touch lab7-2.asm
irina_panyavkina@vbox:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2.asm
```

*Рис. 4.9: Создание файла*

Вставляю в него код из следующего листинга (рис. 4.10)



```

*~/work/arch-pc/lab07/lab7-2.asm - Mousepad
File Edit Search View Document Help
4 msgz db 'наибольшее число: ',0
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'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max

```

Рис. 4.10: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы (рис. 4.11).

```
irina_panyavkina@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
irina_panyavkina@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
irina_panyavkina@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60
```

Рис. 4.11: Запуск исполняемого файла

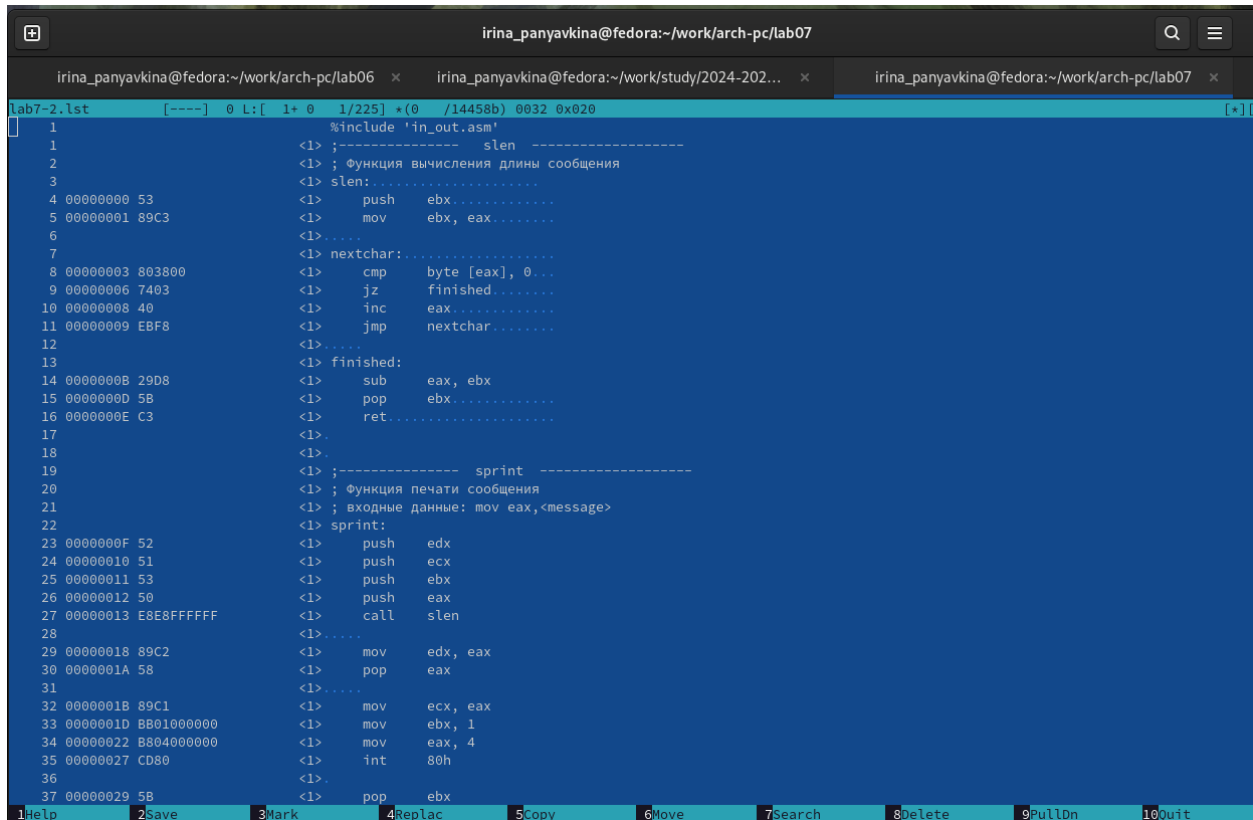
## 4.2 Изучение структуры файла листинга

Получаю файл листинга, указав ключ -l и задав имя файла листинга в командной строке с помощью команды nasm (рис. 4.12).

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

Рис. 4.12: Создание файла листинга

Открываю его с помощью текстового редактора mcedit (рис. 4.13).



```
lab7-2.lst  [----] 0 L: [ 1+ 0 1/225] *(0 /14458b) 0032 0x020  [*]
1          %include 'in_out.asm'
2          <1> ;----- slen -----
3          <1> ; Функция вычисления длины сообщения
4          <1> slen:
5          00000000 53          <1> push    ebx
6          00000001 89C3       <1> mov     ebx, eax
7          <1> .....
8          <1> nextchar:
9          00000003 803800     <1> cmp     byte [eax], 0
10         00000006 7403       <1> jz      finished
11         00000008 40         <1> inc     eax
12         00000009 EBF8       <1> jmp     nextchar
13         <1> .....
14         <1> finished:
15         0000000B 29D8       <1> sub     eax, ebx
16         0000000D 5B         <1> pop     ebx
17         0000000E C3         <1> ret
18         <1> .....
19         <1> ;----- sprint -----
20         <1> ; Функция печати сообщения
21         <1> ; входные данные: mov eax,<message>
22         <1> sprint:
23         0000000F 52         <1> push    edx
24         00000010 51         <1> push    ecx
25         00000011 53         <1> push    ebx
26         00000012 50         <1> push    eax
27         00000013 E8E8FFFFFF <1> call    slen
28         <1> .....
29         00000018 89C2       <1> mov     edx, eax
30         0000001A 58         <1> pop     eax
31         <1> .....
32         0000001B 89C1       <1> mov     ecx, eax
33         0000001D B801000000 <1> mov     ebx, 1
34         00000022 B804000000 <1> mov     eax, 4
35         00000027 CD80       <1> int     80h
36         <1> .....
37         00000029 5B         <1> pop     ebx
```

Рис. 4.13: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может совсем не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем конечно же идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Открываю файл в текстовом редакторе `mouesrad`, так как в нем работать с файлом все-таки удобнее. Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 4.14).

File Edit Search View Document Help

```
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'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в 'max'
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
43 ; ----- Вывод результата
44 fin:
45 mov eax,msg2
46 call sprint ; Вывод сообщения 'Наибольшее число: '
47 mov eax,[max]
48 call iprintLF ; Вывод 'max(A,B,C)'
49 call quit ; Выход
```

Рис. 4.14: Редактирование файла (Удаление операнда из программы)

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 4.15).

```

~/work/arch-pc/lab07/lab7-2.lst - Mousepad
File Edit Search View Document Help
194 19 000000FC E842FFFFFF call sread
195 20 ; ----- Преобразование 'B' из символа в число
196 21 00000101 B8[0A000000] mov eax,B
197 22 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
198 23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
201 26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
202 27 ; ----- Сравниваем 'A' и 'C' (как символы)
203 28 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
204 29 00000122 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
205 30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
206 31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
207 32 ; ----- Преобразование 'max(A,C)' из символа в число
208 33 check_B:
209 34 mov eax,
210 34 ***** error: invalid combination of opcode and operands
211 35 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
212 36 00000135 A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 0000013A 8B0D[00000000] mov ecx,[max]
215 39 00000140 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
216 40 00000146 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
217 41 00000148 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
218 42 0000014E 890D[00000000] mov [max],ecx
219 43 ; ----- Вывод результата
220 44 fin:
221 45 00000154 B8[13000000] mov eax,msg2
222 46 00000159 E8B1FFFFFF call sprint ; Вывод сообщения 'Наибольшее число: '
223 47 0000015E A1[00000000] mov eax,[max]
224 48 00000163 E81EFFFFFF call iprintLF ; Вывод 'max(A,B,C)'
225 49 00000168 E86EFFFFFF call quit ; Выход
226

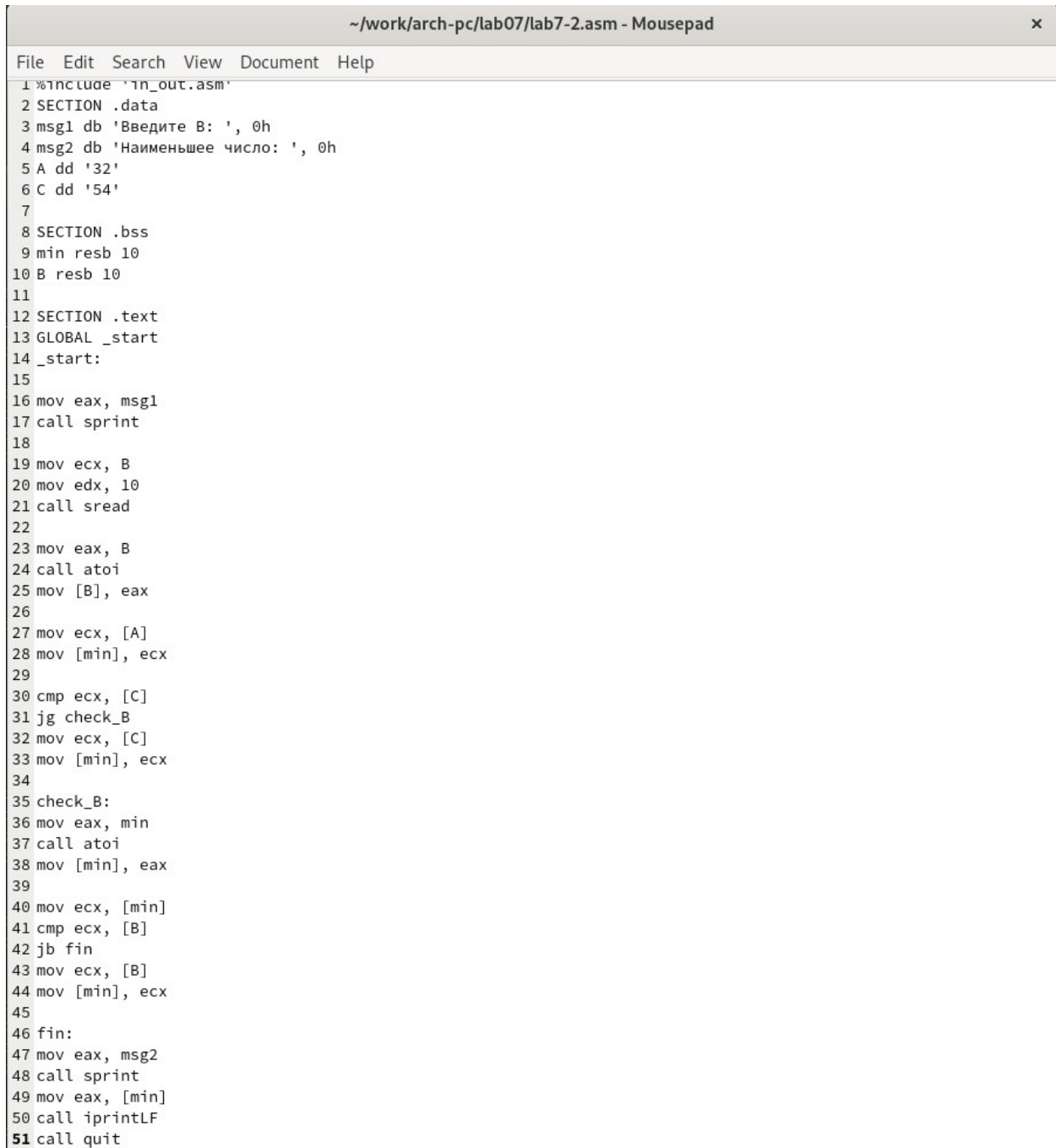
```

Рис. 4.15: Просмотр ошибки в файле листинга

## 4.3 Выполнение заданий для самостоятельной работы

При выполнении предыдущей лабораторной работы, с помощью программы я выяснила, что мой вариант — 15. Мне нужно использовать следующие переменные:  $a = 32$ ,  $b = 6$ ,  $c = 54$ . Возвращаю операнд к функции в программе и

изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 4.16).



```
~/work/arch-pc/lab07/lab7-2.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите B: ', 0h
4 msg2 db 'Наименьшее число: ', 0h
5 A dd '32'
6 C dd '54'
7
8 SECTION .bss
9 min resb 10
10 B resb 10
11
12 SECTION .text
13 GLOBAL _start
14 _start:
15
16 mov eax, msg1
17 call sprint
18
19 mov ecx, B
20 mov edx, 10
21 call sread
22
23 mov eax, B
24 call atoi
25 mov [B], eax
26
27 mov ecx, [A]
28 mov [min], ecx
29
30 cmp ecx, [C]
31 jg check_B
32 mov ecx, [C]
33 mov [min], ecx
34
35 check_B:
36 mov eax, min
37 call atoi
38 mov [min], eax
39
40 mov ecx, [min]
41 cmp ecx, [B]
42 jb fin
43 mov ecx, [B]
44 mov [min], ecx
45
46 fin:
47 mov eax, msg2
48 call sprint
49 mov eax, [min]
50 call iprintLF
51 call quit
```

Рис. 4.16: Первая программа самостоятельной работы

Код первой программы:

```
%include 'in_out.asm'

SECTION .data

msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '32'
C dd '54'

SECTION .bss

min 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
mov [B], eax

mov ecx, [A]
mov [min], ecx
```

```
cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx
```

```
check_B:
mov eax, min
call atoi
mov [min], eax
```

```
mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx
```

```
fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
call quit
```

Проверяю корректность написания первой программы (рис. 4.17).



```
irina_panyavkina@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
irina_panyavkina@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
irina_panyavkina@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 6
Наименьшее число: 6
```

*Рис. 4.17: Запуск исполняемого файла*

Создаю программу, которая будет вычислять значение заданной функции, согласно моему варианту, для введенных с клавиатуры переменных а и х (рис. 4.18).

File Edit Search View Document Help

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg_x: DB 'Введите значение переменной x: ', 0
4 msg_a: DB 'Введите значение переменной a: ', 0
5 res: DB 'Результат: ', 0
6 SECTION .bss
7 x: RESB 80
8 a: RESB 80
9 SECTION .text
10 GLOBAL _start
11 _start:
12     ; Запрос и ввод значения x
13     mov eax, msg_x
14     call sprint
15     mov ecx, x
16     mov edx, 80
17     call sread
18     mov eax, x
19     call atoi
20     mov edi, eax    ; Сохранение x в edi
21
22     ; Запрос и ввод значения a
23     mov eax, msg_a
24     call sprint
25     mov ecx, a
26     mov edx, 80
27     call sread
28     mov eax, a
29     call atoi
30     mov esi, eax    ; Сохранение a в esi
31
32     ; Сравнение значений x и a
33     cmp edi, esi    ; Сравнение x (edi) с a (esi)
34     jl less_than_a  ; Если x < a, перейти на less_than_a
35
36 greater_or_equal_a:
37     ; Если x >= a: результат = x + 10
38     mov eax, edi    ; eax = x
39     add eax, 10
40     jmp print_result
41
42 less_than_a:
43     ; Если x < a: результат = a + 10
44     mov eax, esi    ; eax = a
45     add eax, 10
46
47 print_result:
48     ; Вывод результата
49     mov edi, eax    ; Сохранение результата в edi для iprintLF
50     mov eax, res    ; Сообщение "Результат: "
51     call sprint
52     mov eax, edi    ; Результат для печати
53     call iprintLF
54     call quit      ; Завершение программы

```

*Рис. 4.18: Вторая программа самостоятельной работы*

Код второй программы:

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

```
SECTION .data
```

```
msg_x: DB 'Введите значение переменной x: ', 0
```

```
msg_a: DB 'Введите значение переменной a: ', 0
```

```
res: DB 'Результат: ', 0
```

```
SECTION .bss
```

```
x: RESB 80
```

```
a: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
    ; Запрос и ввод значения x
```

```
    mov eax, msg_x
```

```
    call sprint
```

```
    mov ecx, x
```

```
    mov edx, 80
```

```
    call sread
```

```
    mov eax, x
```

```
    call atoi
```

```
    mov edi, eax    ; Сохранение x в edi
```

```
    ; Запрос и ввод значения a
```

```
    mov eax, msg_a
```

```
    call sprint
```

```
    mov ecx, a
```

```
    mov edx, 80
```

```
    call sread
```

```
    mov eax, a
```

```
    call atoi
```

```
    mov esi, eax    ; Сохранение a в esi
```

```
; Сравнение значений x и a
cmp edi, esi ; Сравнение x (edi) с a (esi)
jl less_than_a ; Если x < a, перейти на less_than_a
```

greater\_or\_equal\_a:

```
; Если x >= a: результат = x + 10
mov eax, edi ; eax = x
add eax, 10
jmp print_result
```

less\_than\_a:

```
; Если x < a: результат = a + 10
mov eax, esi ; eax = a
add eax, 10
```

print\_result:

```
; Вывод результата
mov edi, eax ; Сохранение результата в edi для iprintLF
mov eax, res ; Сообщение "Результат: "
call sprint
mov eax, edi ; Результат для печати
call iprintLF
call quit ; Завершение программы
```

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений x и a, мне предложено использовать значения (2;3) и (4;2) (рис. 4.19).

```
irina_panyavkina@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
irina_panyavkina@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
irina_panyavkina@vbox:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 2
Введите значение переменной a: 3
Результат: 13
irina_panyavkina@vbox:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 4
Введите значение переменной a: 2
Результат: 14
irina_panyavkina@vbox:~/work/arch-pc/lab07$
```

*Рис. 4.19: Запуск исполняемого файла*

## 5 ВЫВОДЫ

Во время выполнения лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файлов листинга.

## Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.Org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).

16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).