

# **Отчёта по лабораторной работе №7**

**Команды безусловного и условного переходов в Nasm.  
Программирование ветвлений.**

Новичков Максим Алексеевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Реализация переходов в NASM . . . . .	6
3.2	Изучение структуры файлы листинга . . . . .	9
3.3	Задание для самостоятельной работы . . . . .	11
<b>4</b>	<b>Выводы</b>	<b>15</b>

# 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 . . . . .	6
3.3	Запускаем файл и смотрим на его работу . . . . .	7
3.4	Изменяем файл . . . . .	7
3.5	Запускаем файл и смотрим на его работу . . . . .	7
3.6	Создаем файл командой <code>touch</code> . . . . .	8
3.7	Заполняем файл . . . . .	8
3.8	Смотрим на работу программ . . . . .	9
3.9	Создаем файл листинга . . . . .	9
3.10	Изучаем файл . . . . .	9
3.11	Удаляем операндум из файла . . . . .	10
3.12	Транслируем файл . . . . .	10
3.13	Изучаем файл с ошибкой . . . . .	11
3.14	Создаем файл командой <code>touch</code> . . . . .	11
3.15	Пишем программу . . . . .	12
3.16	Смотрим на работу программы(всё верно) . . . . .	12
3.17	Пишем программу . . . . .	13
3.18	Проверяем работу программы . . . . .	14

# 1 Цель работы

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

## 2 Задание

Написать программы для решения системы выражений.

## 3 Выполнение лабораторной работы

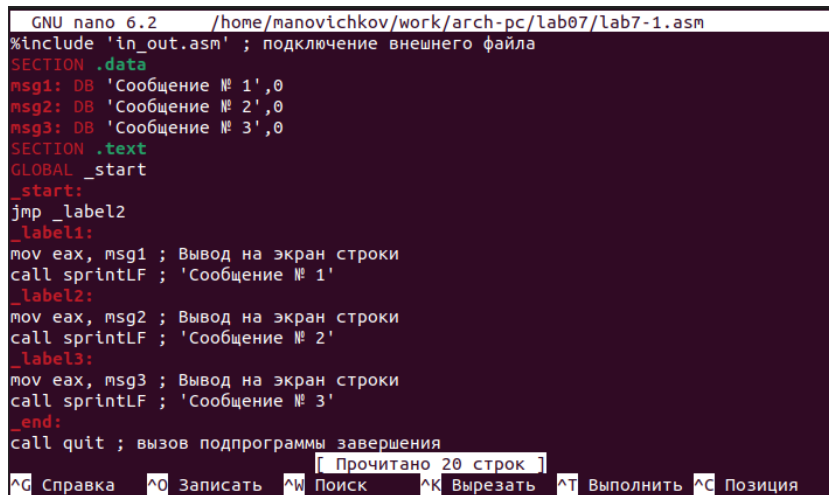
### 3.1 Реализация переходов в NASM

Создаем каталог для программ ЛБ7, и в нем создаем файл (рис. 3.1).

```
manovichkov@manovichkov-VirtualBox:~$ ^C
manovichkov@manovichkov-VirtualBox:~$ mkdir ~/work/arch-pc/lab07
manovichkov@manovichkov-VirtualBox:~$ cd ~/work/arch-pc/lab07
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ touch lab7-1.asm
```

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 7.1 (рис. 3.2).



```
GNU nano 6.2 /home/manovichkov/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
[ Прочитано 20 строк ]
^C Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^S Позиция
```

Рис. 3.2: Заполняем файл

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

```
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 3.3: Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его 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 sprintLF ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintLF ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintLF ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения
```

Рис. 3.4: Изменяем файл

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

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

Рис. 3.5: Запускаем файл и смотрим на его работу

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

Рис. 3.6: Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.3 (рис. 3.7).

```
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'
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
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'
```

Рис. 3.7: Заполняем файл

Создаем исполняемый файл и проверяем его работу, вводя разные значения B (рис. 3.8).



```

manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50

```

Рис. 3.8: Смотрим на работу программ

## 3.2 Изучение структуры файлы листинга

Создаем файл листинга для программы lab7-2.asm (рис. 3.9).

```

manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ touch lab7-2.lst

```

Рис. 3.9: Создаем файл листинга

Открываем файл листинга с помощью команды `mcedit` и изучаем его (рис. 3.10).

```

/home/ma~7-2.lst [----] 0 L: [ 1+ 0 1/225] *(0 /14458b) 0032 0x020 [*][X]
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>
20                                     <1> ;----- sprint -----
21                                     <1> ; Функция печати сообщения
22                                     <1> ; входные данные: mov eax, <message>

```

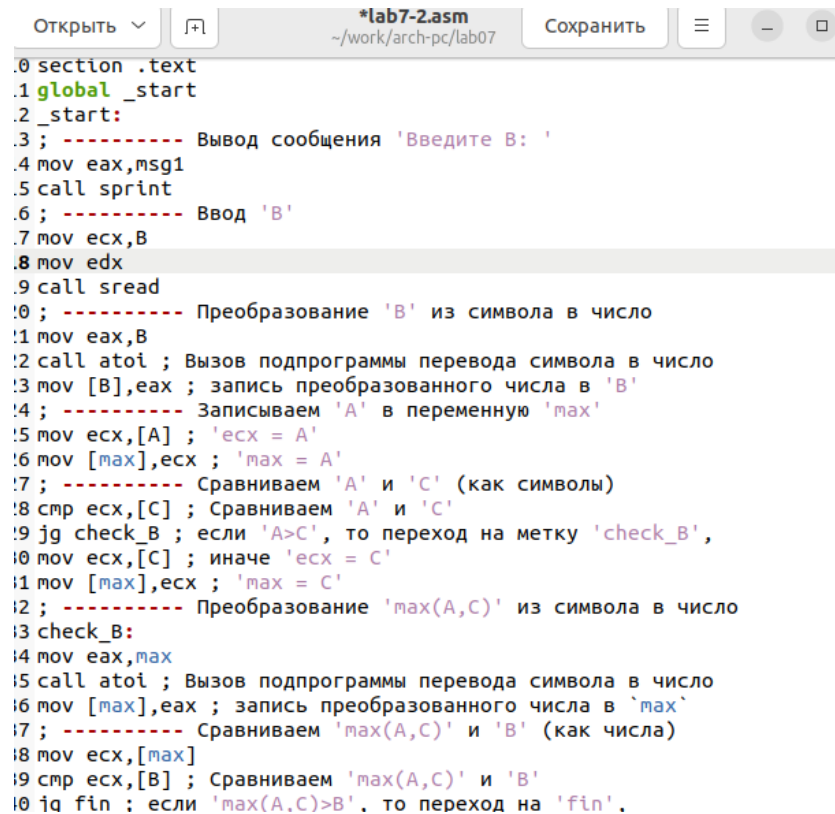
Рис. 3.10: Изучаем файл

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.

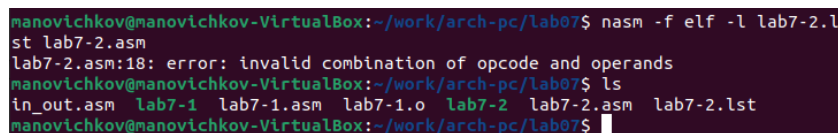
Открываем файл и удаляем один операндум (рис. 3.11).



```
Открыть ▾ | *lab7-2.asm | ~/work/arch-pc/lab07 | Сохранить | ≡ | - | □
.0 section .text
.1 global _start
.2 _start:
.3 ; ----- Вывод сообщения 'Введите B: '
.4 mov eax,msg1
.5 call sprint
.6 ; ----- Ввод 'B'
.7 mov ecx,B
.8 mov edx
.9 call sread
!0 ; ----- Преобразование 'B' из символа в число
!1 mov eax,B
!2 call atoi ; Вызов подпрограммы перевода символа в число
!3 mov [B],eax ; запись преобразованного числа в 'B'
!4 ; ----- Записываем 'A' в переменную 'max'
!5 mov ecx,[A] ; 'ecx = A'
!6 mov [max],ecx ; 'max = A'
!7 ; ----- Сравниваем 'A' и 'C' (как символы)
!8 cmp ecx,[C] ; Сравниваем 'A' и 'C'
!9 jg check_B ; если 'A>C', то переход на метку 'check_B',
!0 mov ecx,[C] ; иначе 'ecx = C'
!1 mov [max],ecx ; 'max = C'
!2 ; ----- Преобразование 'max(A,C)' из символа в число
!3 check_B:
!4 mov eax,max
!5 call atoi ; Вызов подпрограммы перевода символа в число
!6 mov [max],eax ; запись преобразованного числа в 'max'
!7 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
!8 mov ecx,[max]
!9 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
!0 iq fin ; если 'max(A,C)>B'. то переход на 'fin'.
```

Рис. 3.11: Удаляем операндум из файла

Транслируем с получением файла листинга (рис. 3.12).



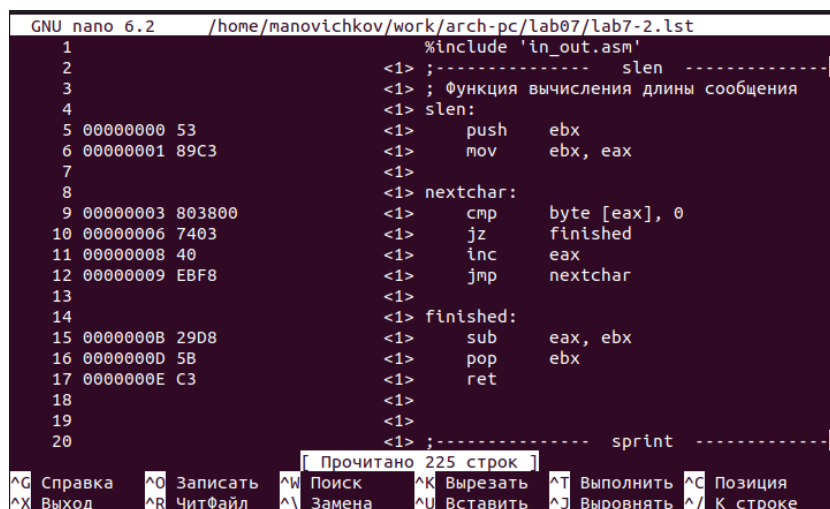
```
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.l
st lab7-2.asm
lab7-2.asm:18: error: invalid combination of opcode and operands
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 3.12: 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. 3.13).



```
GNU nano 6.2 /home/manovichkov/work/arch-pc/lab07/lab7-2.lst
1      %include 'in_out.asm'
2
3      <1> ;----- slen ----->
4      <1> ; Функция вычисления длины сообщения
5      <1> slen:
6      <1>     push    ebx
7      <1>     mov     ebx, eax
8
9      <1> nextchar:
10     <1>     cmp     byte [eax], 0
11     <1>     jz      finished
12     <1>     inc     eax
13     <1>     jmp     nextchar
14
15     <1> finished:
16     <1>     sub     eax, ebx
17     <1>     pop     ebx
18     <1>     ret
19
20     <1> ;----- sprint ----->
```

Прочитано 225 строк

^G Справка ^O Записать ^M Поиск ^K Вырезать ^T Выполнить ^C Позиция  
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^/\_ К строке

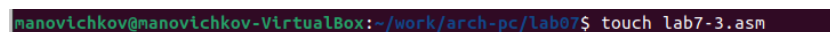
Рис. 3.13: Изучаем файл с ошибкой

### 3.3 Задание для самостоятельной работы

ВАРИАНТ-20

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных  $x, y, z$  и с. 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. 3.14).



```
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ touch lab7-3.asm
```

Рис. 3.14: Создаем файл командой touch

Открываем его и пишем программу, которая выберет наименьшее число из трех (2 числа уже в программе, 3е вводится из консоли) (рис. 3.15).

```

1 %include 'in_out.asm'
2 section .data
3 msg2 db "Наименьшее число - ",0h
4 A dd '24'
5 B dd '98'
6 C dd '15'
7 section .bss
8 min resb 10
9 section .text
10 global _start
11 _start:
12 ; ----- Преобразование 'B' из символа в число
13 mov eax,B
14 call atoi ; Вызов подпрограммы перевода символа в число
15 mov [B],eax ; запись преобразованного числа в 'B'
16 mov eax,A
17 call atoi ; Вызов подпрограммы перевода символа в число
18 mov [A],eax ; запись преобразованного числа в 'A'
19 mov eax,C
20 call atoi ; Вызов подпрограммы перевода символа в число
21 mov [C],eax ; запись преобразованного числа в 'C'
22 ; ----- Записываем 'A' в переменную 'min'
23 mov ecx,[A] ; 'ecx = A'
24 mov [min],ecx ; 'min = A'
25 ; ----- Сравниваем 'A' и 'C' (как символы)
26 cmp ecx,[C] ; Сравниваем 'A' и 'C'
27 jl check_B ; если 'A<C', то переход на метку 'check_B',
28 mov ecx,[C] ; иначе 'ecx = C'
29 mov [min],ecx ; 'min = C'
30 ; ----- Преобразование 'min(A,C)' из символа в число
31 check_B:
32 ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
33 mov ecx,[min]
34 cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
35 jl fin ; если 'min(A,C)<B', то переход на 'fin',
36 mov ecx,[B] ; иначе 'ecx = B'

```

Рис. 3.15: Пишем программу

Транслируем файл и смотрим на работу программы (рис. 3.16).

```

manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число - 15
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$

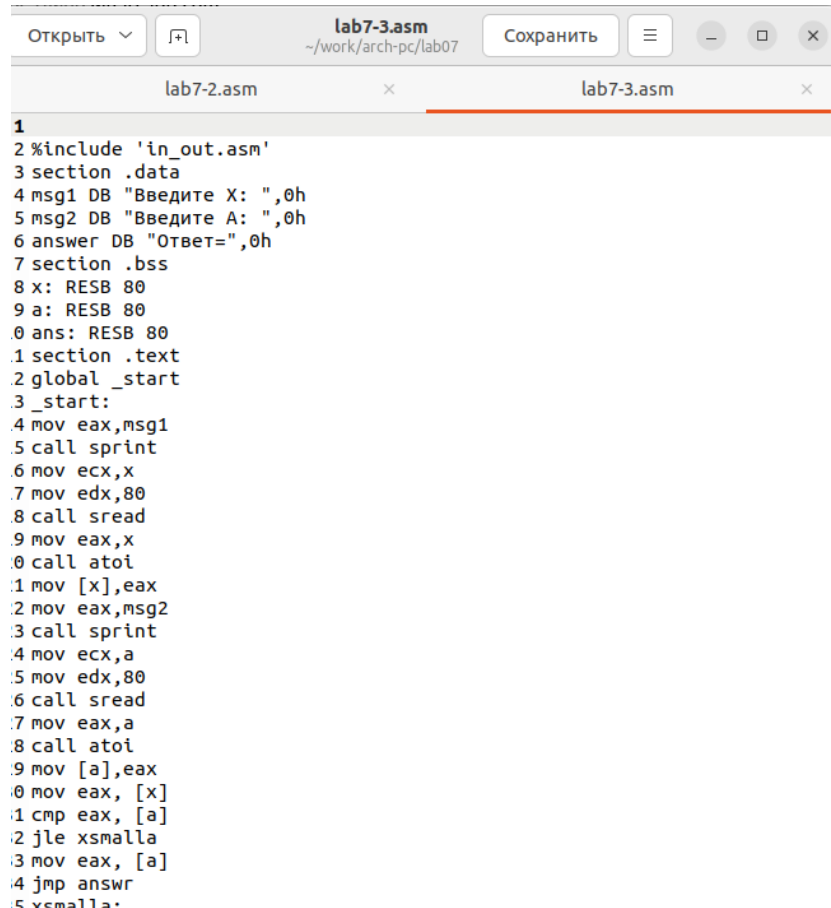
```

Рис. 3.16: Смотрим на работу программы(всё верно)

2. Напишите программу, которая для введенных с клавиатуры значений  $\boxed{x}$  и  $\boxed{y}$  вычисляет значение заданной функции  $\boxed{f(x,y)}$  и выводит результат вычислений. Вид функции  $\boxed{f(x,y)}$  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

❌ и ❌ из 7.6.

Открываем файл и пишем программу, которая решит систему уравнений, при данных, введенных в консоль (рис. 3.17).



```
1
2 %include 'in_out.asm'
3 section .data
4 msg1 DB "Введите X: ",0h
5 msg2 DB "Введите A: ",0h
6 answer DB "Ответ=",0h
7 section .bss
8 x: RESB 80
9 a: RESB 80
10 ans: RESB 80
11 section .text
12 global _start
13 _start:
14 mov eax,msg1
15 call sprint
16 mov ecx,x
17 mov edx,80
18 call sread
19 mov eax,x
20 call atoi
21 mov [x],eax
22 mov eax,msg2
23 call sprint
24 mov ecx,a
25 mov edx,80
26 call sread
27 mov eax,a
28 call atoi
29 mov [a],eax
30 mov eax,[x]
31 cmp eax,[a]
32 jle xsmalla
33 mov eax,[a]
34 jmp answer
35 xsmalla
```

Рис. 3.17: Пишем программу

Транслируем файл и проверяем его работу при  $x=6$  и  $a=4$ (рис. 3.18).

![Проверяем работу программы](image/Снимок экрана от 2023-11-29 21-13-09.png){#fig:021 width=70%}

Транслируем файл и проверяем его работу при  $x=6$  и  $a=4$ (рис. 3.18).

```
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Введите X: 5
Введите A: 7
Ответ=12
manovichkov@manovichkov-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 3.18: Проверяем работу программы

## 4 Выводы

Я Изучил команды условных и безусловных переходов. Приобрел навыки написания программ с использованием переходов. Познакомился с назначением и структурой файла листинга.