

Лабораторная работа №7

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

Седохин Даниил Алексеевич

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Задание для самостоятельной работы	15
4	Выводы	20

Список иллюстраций

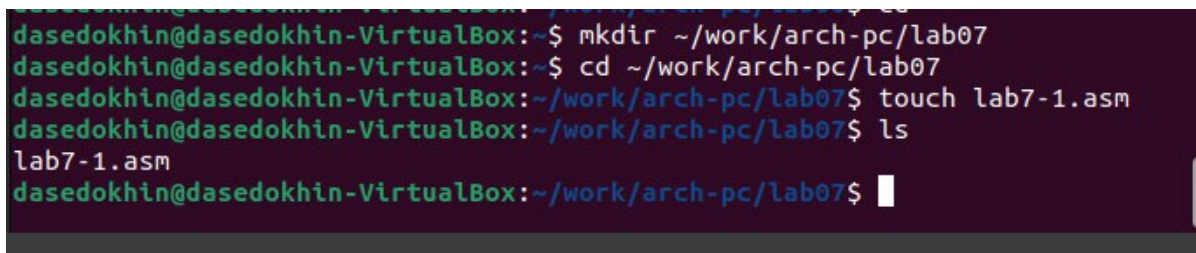
2.1	Создание каталога для лабораторной работы №7 и файла lab7-1.asm	5
2.2	Ввод в файл lab7-1.asm текст программы из листинга 7.1	6
2.3	Создание и проверка исполняемого файла	6
2.4	Изменение текста программы в соответствии с листингом 7.2 . .	7
2.5	Создание и проверка исполняемого файла	7
2.6	Редактирование текста программы	8
2.7	Создание и проверка исполняемого файла	9
2.8	Создание файла lab7-2.asm	9
2.9	Ввод текста программы из листинга 7.3 в lab7-2.asm	10
2.10	Создание и проверка исполняемого файла	11
2.11	Создание и проверка исполняемого файла	11
2.12	Просмотра файла листинга lab7-2.lst	12
2.13	Выбранные строки	13
2.14	Изменённая строка	13
2.15	Изменения в созданном файле листинга	14
2.16	Создание файла листинга	14
3.1	Создание файла srab1.asm	15
3.2	Редактирование текста файла srab1.asm	16
3.3	Создание и проверка исполняемого файла	19

1 Цель работы

Изучить команды условного и безусловного переходов. Приобрести навыки написания программ с использованием переходов. Ознакомиться с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

- 1) Создадим каталог для лабораторной работы № 7, перейдём в него и создадим файл lab7-1.asm (рис. 2.1).



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

Рис. 2.1: Создание каталога для лабораторной работы №7 и файла lab7-1.asm

- 2) Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Введём в файл lab7-1.asm текст программы из листинга 7.1. (рис. 2.2).

```
GNU nano 6.2 /home/dasedokhin/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
```

Рис. 2.2: Ввод в файл lab7-1.asm текст программы из листинга 7.1

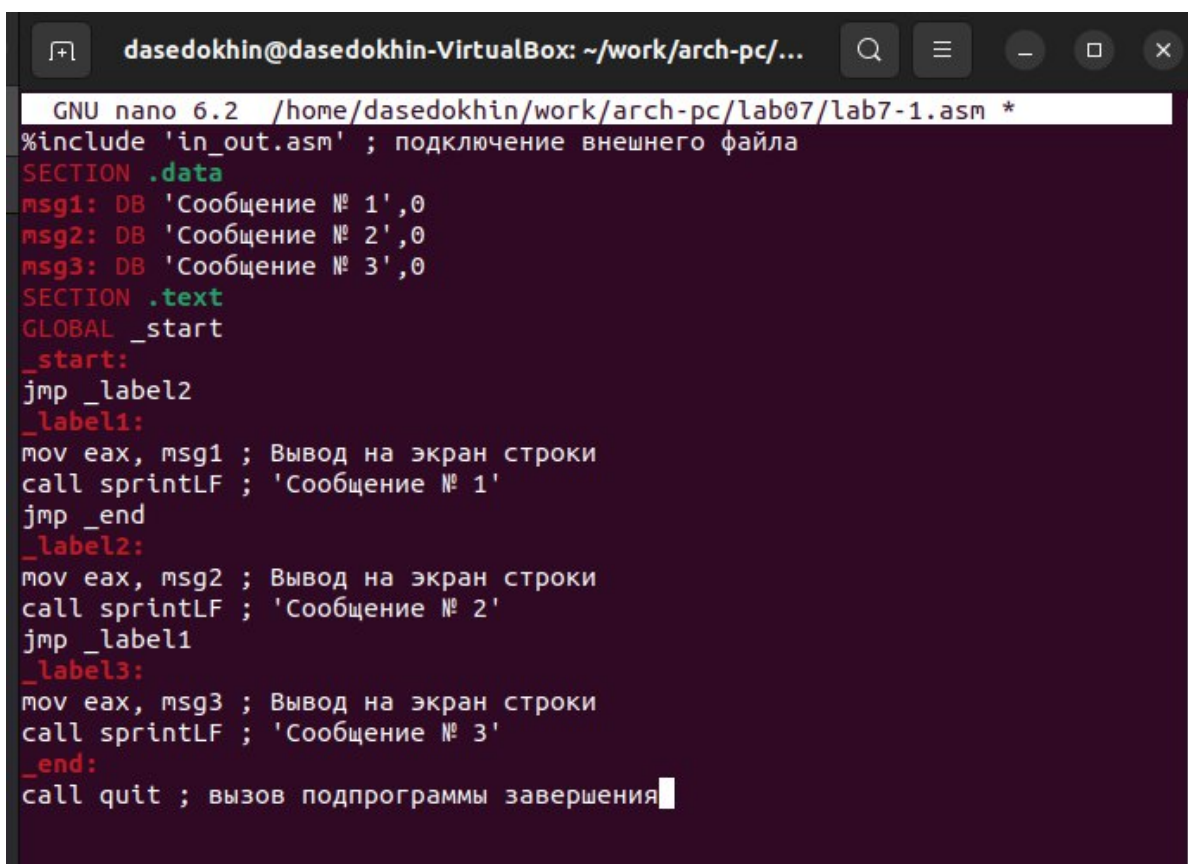
Создадим исполняемый файл и запустим его. (рис. 2.3).

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

Рис. 2.3: Создание и проверка исполняемого файла

Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е.

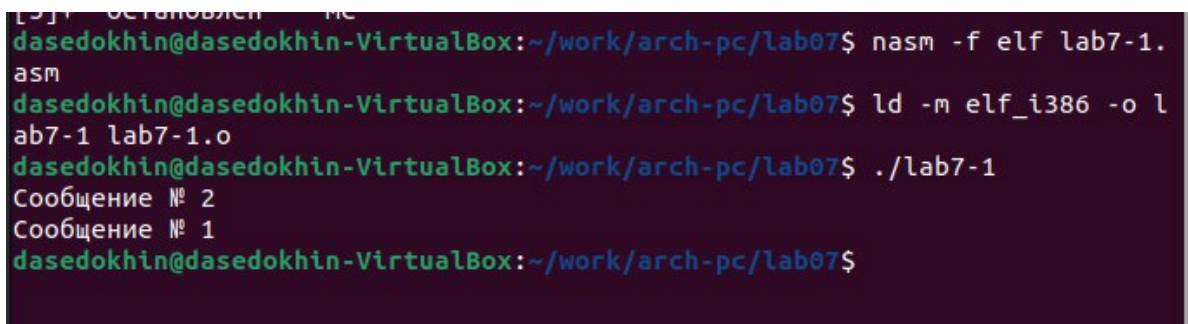
переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Изменим текст программы в соответствии с листингом 7.2 (рис. 2.4).



```
GNU nano 6.2 /home/dasedokhin/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 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
```

Рис. 2.4: Изменение текста программы в соответствии с листингом 7.2

Создадим исполняемый файл и проверим его работу. (рис. 2.5).



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

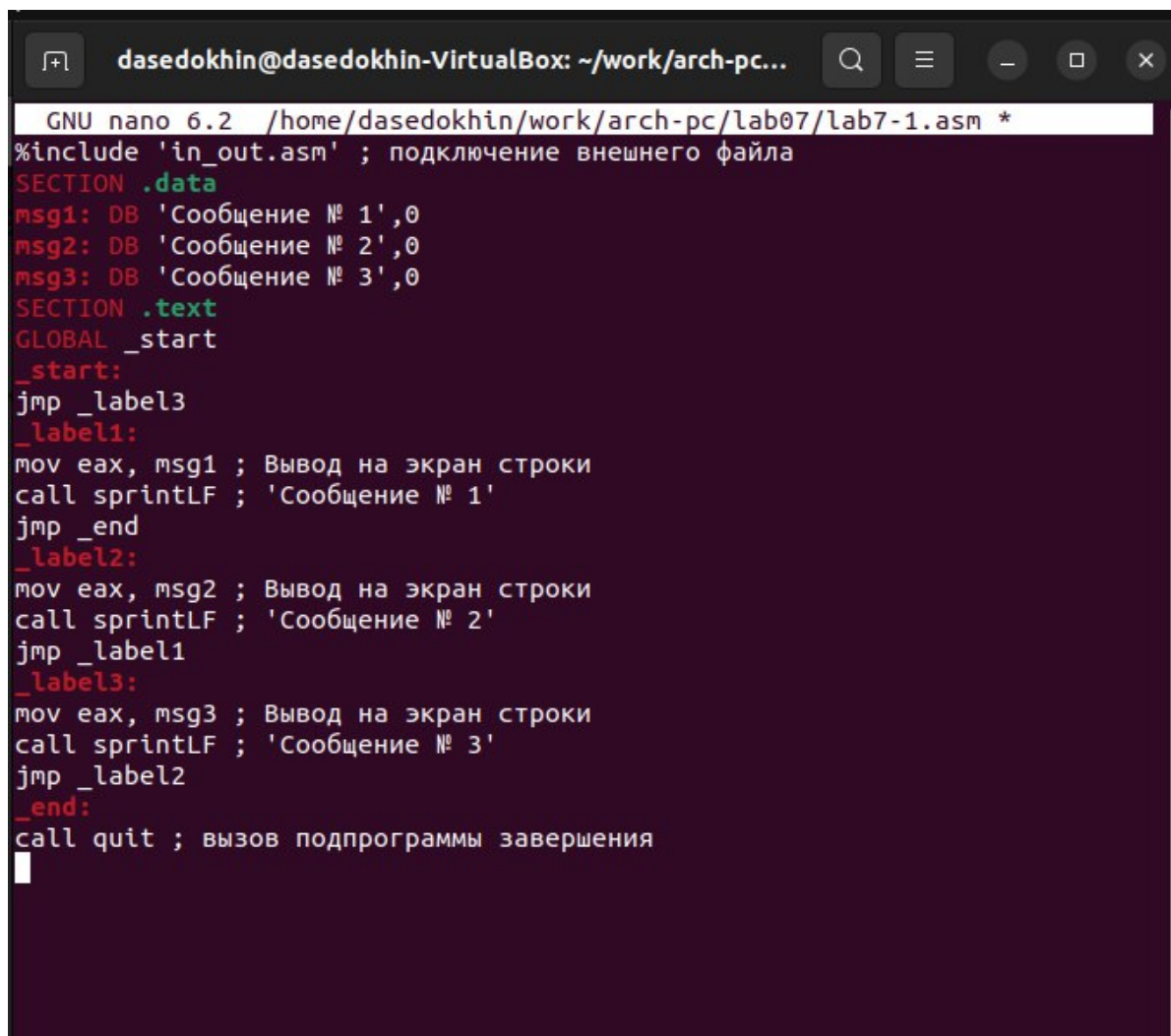
Рис. 2.5: Создание и проверка исполняемого файла

Изменим текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим: (рис. 2.6).

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
GNU nano 6.2 /home/dasedokhin/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
```

Рис. 2.6: Редактирование текста программы

Создадим исполняемый файл и проверим его. (рис. 2.7).


```

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

```

Рис. 2.7: Создание и проверка исполняемого файла

- 3) Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры. Создадим файл `lab7-2.asm` в каталоге `~/work/arch-pc/lab07`. (рис. 2.8).

```

dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ touch lab7-2.asm ~/
work/arch-pc/lab07
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$

```

Рис. 2.8: Создание файла `lab7-2.asm`

Введём текст программы из листинга 7.3 в `lab7-2.asm`. (рис. 2.9).

```
ября 20:09 en
dasedokhin@dasedokhin-VirtualBox: ~/work/arch-pc/...
GNU nano 6.2 /home/dasedokhin/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'
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'

^G Справка      ^O Записать     ^W Поиск        ^K Вырезать     ^T Выполнить
^X Выход        ^R ЧитФайл     ^\ Замена       ^U Вставить     ^J Выровнять
```

Рис. 2.9: Ввод текста программы из листинга 7.3 в lab7-2.asm

Создадим исполняемый файл и проверим его работу для разных значений В.
(рис. 2.10).

```
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 10
Наибольшее число: 50
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 70
Наибольшее число: 70
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 100
Наибольшее число: 100
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.10: Создание и проверка исполняемого файла

4) Создадим файл листинга для программы из файла lab7-2.asm. (рис. 2.11).

```
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1.asm  lab7-2      lab7-2.lst
lab7-1      lab7-1.o    lab7-2.asm  lab7-2.o
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.11: Создание и проверка исполняемого файла

Откроем файл листинга lab7-2.lst с помощью любого текстового редактора mcedit. (рис. 2.12).

```

dasedokhin@dasedokhin-VirtualBox: ~/work/arch-pc/...
/home/da~7-2.lst [----] 0 L:[ 1+ 0 1/225] *(0 /14458b) 0032 [*][X]
1      %include 'in_out.asm'
2      <1> ;----- slen -----
3      <1> ; Функция вычисления длины сообщен
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
23     <1> sprint:
24     0000000F 52      <1>      push     edx
25     00000010 51      <1>      push     ecx
26     00000011 53      <1>      push     ebx
27     00000012 50      <1>      push     eax
28     00000013 E8E8FFFFFF <1>      call     slen
29     <1>.....
30     00000018 89C2    <1>      mov      edx, eax
31     0000001A 58      <1>      pop      eax
32     <1>.....
33     0000001B 89C1    <1>      mov      ecx, eax
34     0000001D BB01000000 <1>      mov      ebx, 1
35     00000022 B804000000 <1>      mov      eax, 4
36     00000027 CD80    <1>      int      80h
37     <1>.....

```

Рис. 2.12: Просмотра файла листинга lab7-2.lst

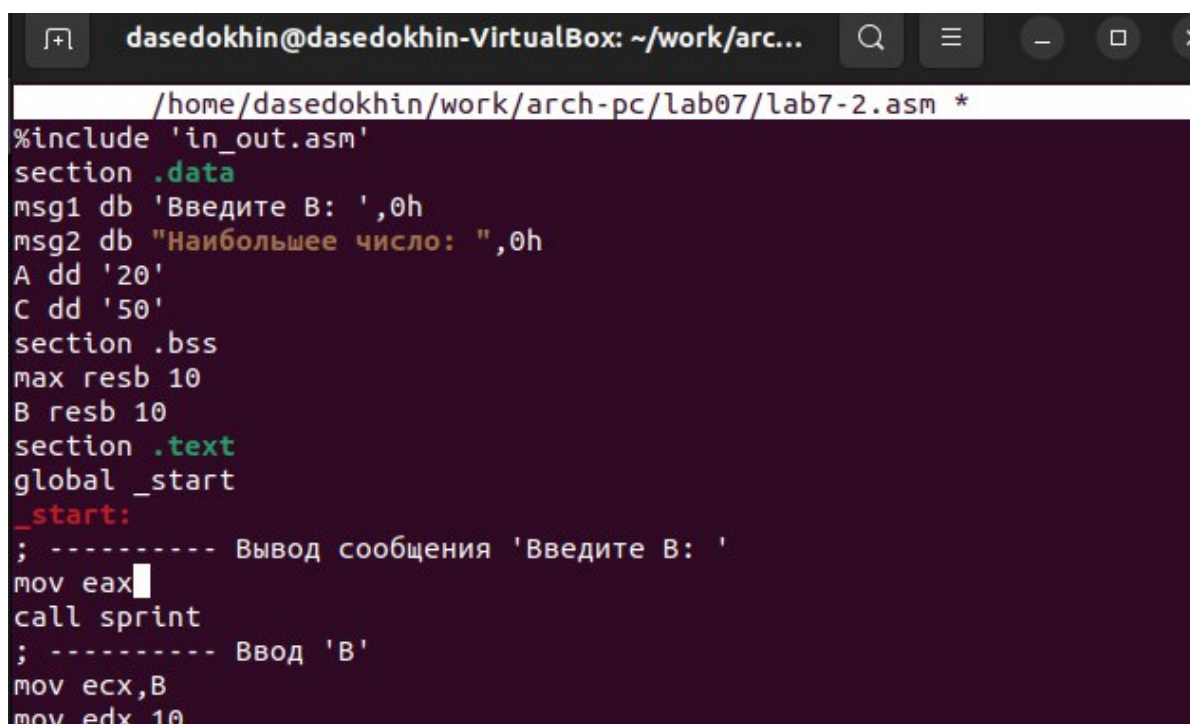
Три строки выбранных мною следующие: (рис. 2.13): - 45 - строка, 00000159 адрес, B8[13000000] - машинный код, mov eax, msg2 - исходный тест программы. В котором записывается адрес msg2 в EAX. - 46 - строка, 0000015E адрес, E8ACFEFFFF - машинный код, call sprint - исходный тест программы. В котором

вызывается функция `sprint`. Она в свою очередь выводит сообщение содержащееся в переменной `msg2` - 47 - строчка, 00000163 адрес, `A1[00000000]` - машинный код, `mov eax,[max]` - исходный тест программы. В котором записывается адрес переменной `[max]` в `EAX`.

```
45          <1> ; Функция печати сообщения с переводом строки
46          <1> ; входные данные: mov eax,<message>
47          <1> sprintLF:
```

Рис. 2.13: Выбранные строки

Откроем файл с программой `lab7-2.asm` и уберём операнд из 14 строки. (рис. 2.14).



```
/home/dasedokhin/work/arch-pc/lab07/lab7-2.asm *
%include 'in_out.asm'
section .data
msg1 db 'Введите В: ',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 В: '
mov eax,
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
```

Рис. 2.14: Изменённая строка

Создадим файл листинга в котором на 14 строке появилась ошибка в связи с нашими изменениями операнды. (рис. 2.15 2.16).

```
14          mov eax
14          ***** error: invalid combination of opcode and operands
15 000000F8_F022FFFFFF call sprintf
```

Рис. 2.15: Изменения в созданном файле листинга

```
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.16: Создание файла листинга

3 Задание для самостоятельной работы

Напишем программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных будут равны (79, 83, 41) в соответствии с 6-м вариантом.

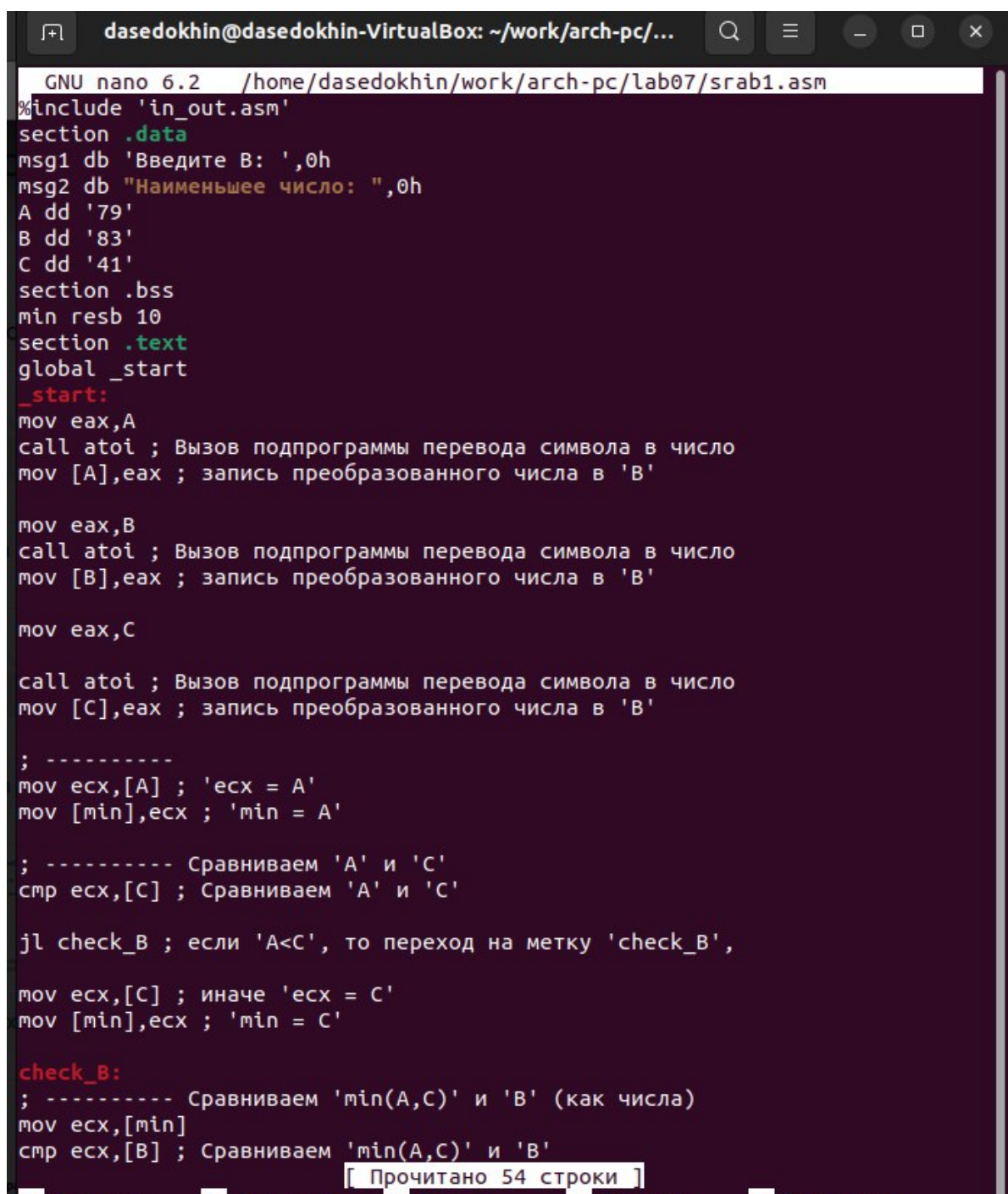
Для начала создадим файл srab1.asm в каталоге ~/work/arch-pc/lab07. (рис. 3.1).

A screenshot of a terminal window with a dark background. The prompt is 'dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07\$'. The command entered is 'touch srab1.asm ~/work/arch-pc/lab07'.

```
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ touch srab1.asm ~/work/arch-pc/lab07
```

Рис. 3.1: Создание файла srab1.asm

За основу программы возьмём текст из листинга 7.3. Внесём необходимые изменения в текст файла. (рис. 3.2).



```
GNU nano 6.2 /home/dasedokhin/work/arch-pc/lab07/srab1.asm
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '79'
B dd '83'
C dd '41'
section .bss
min resb 10
section .text
global _start
_start:
mov eax,A
call atoi ; Вызов подпрограммы перевода символа в число
mov [A],eax ; запись преобразованного числа в 'B'

mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'

mov eax,C
call atoi ; Вызов подпрограммы перевода символа в число
mov [C],eax ; запись преобразованного числа в 'B'

; -----
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'

; ----- Сравниваем 'A' и 'C'
cmp ecx,[C] ; Сравниваем 'A' и 'C'

jnl check_B ; если 'A<C', то переход на метку 'check_B',

mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'

check_B:
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
```

Прочитано 54 строки

Рис. 3.2: Редактирование текста файла srab1.asm

Изменённый текст файла srab1.asm


```

%include 'in_out.asm'

section .data
msg2 db "Наименьшее число: ",0h
A dd '79'
B dd '83'
C dd '41'

section .bss
min resb 10

section .text
global _start
_start:

mov eax,A
call atoi ; Вызов подпрограммы перевода символа в число
mov [A],eax ; запись преобразованного числа в 'B'

mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'

mov eax,C
call atoi ; Вызов подпрограммы перевода символа в число
mov [C],eax ; запись преобразованного числа в 'B'

; -----
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'

```

```

; ----- Сравниваем 'A' и 'C'
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'

check_B:
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'

jl fin

mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprintf ; Вывод сообщения 'Наибольшее число: '
mov eax,[min]
call fprintf ; Вывод 'min(A,B,C)'
call _exit ; Выход

```

Создадим исполняемый файл и проверим его работу. (рис. 3.3).

```
[25] ~ - установлен gcc
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf srab1.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o srab1 srab1.o
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$ ./srab1
Наименьшее число: 41
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 3.3: Создание и проверка исполняемого файла

4 Выводы

Я изучил команды условного и безусловного переходов. Приобрёл навыки написания программ с использованием переходов. Ознакомился с назначением и структурой файла листинга.