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).

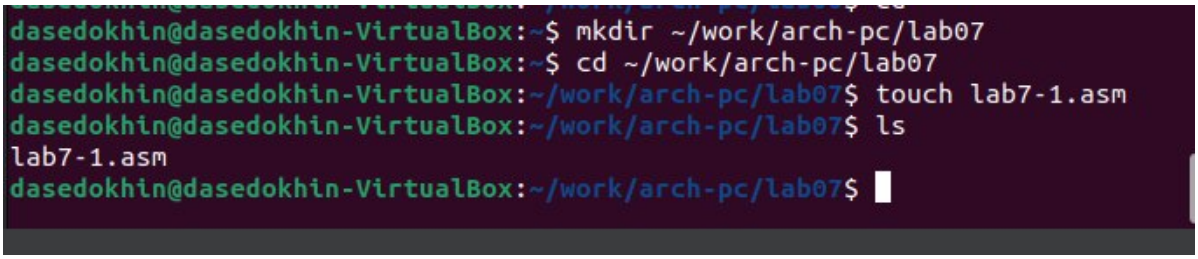

Рис. 2.1: Создание каталога для лабораторной работы №7 и файла lab7-1.asm

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).

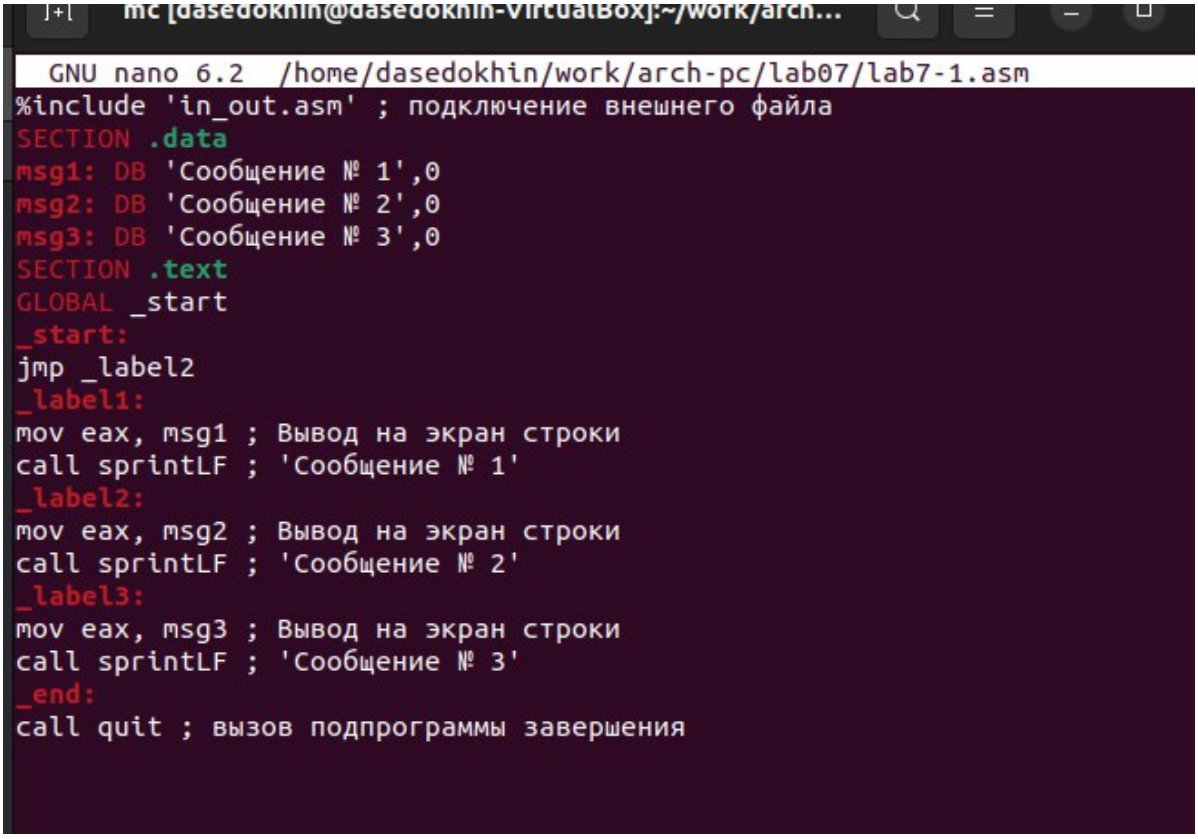

Рис. 2.2: Ввод в файл lab7-1.asm текст программы из листинга 7.1

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

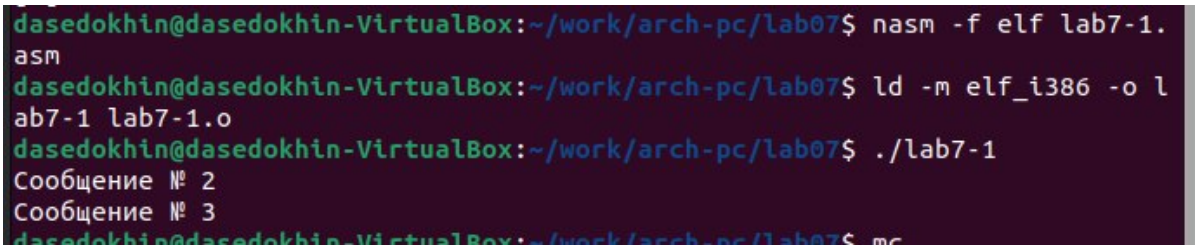

Рис. 2.3: Создание и проверка исполняемого файла

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).

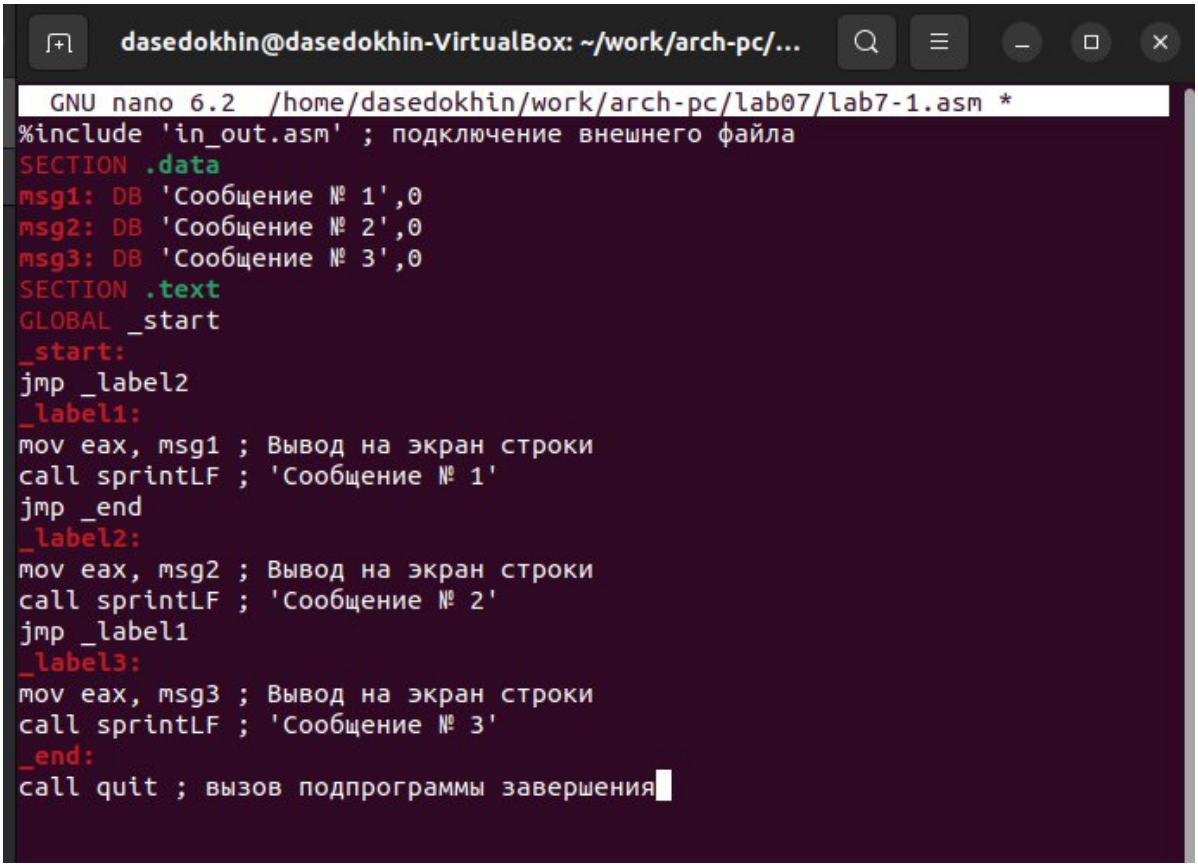

Рис. 2.4: Изменение текста программы в соответствии с листингом 7.2

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

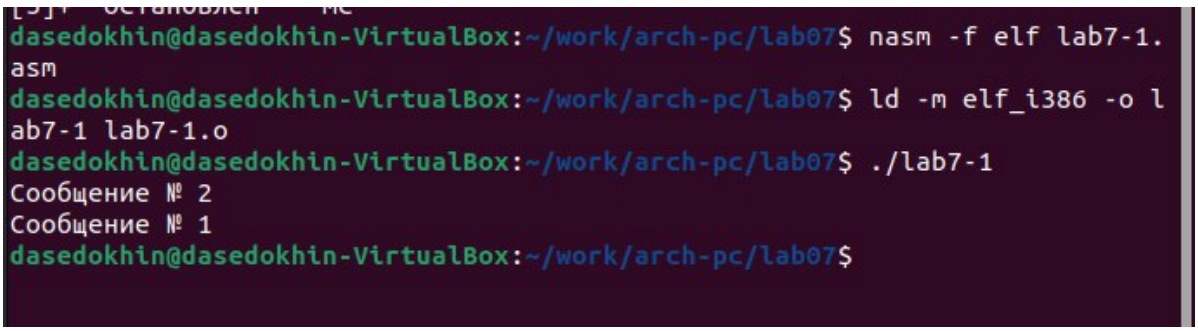

Рис. 2.5: Создание и проверка исполняемого файла

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

Сообщение № 3

Сообщение № 2

Сообщение № 1

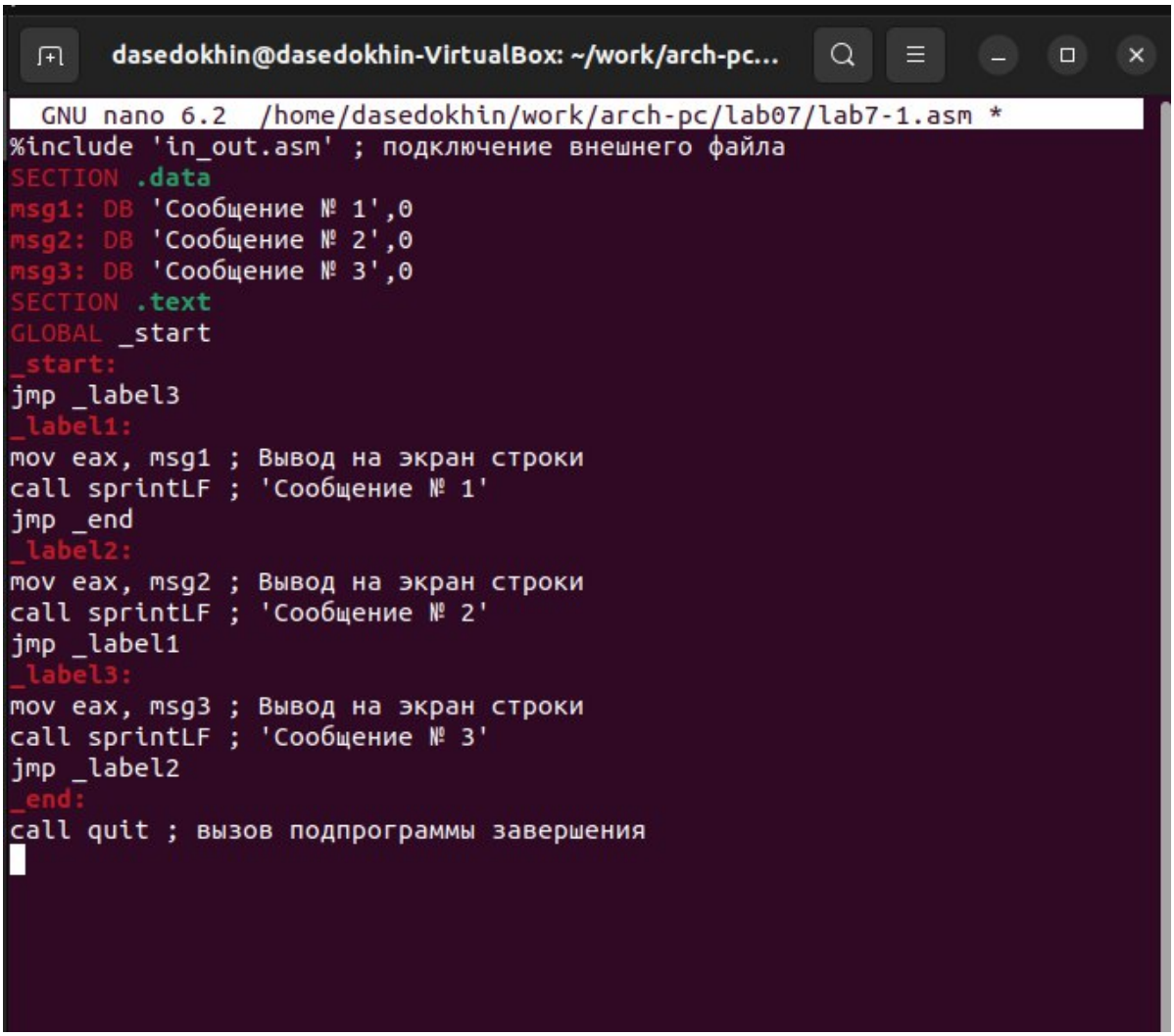

Рис. 2.6: Редактирование текста программы

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: A,B и C. Значения для A и C задаются в программе, значение B вводиться 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).

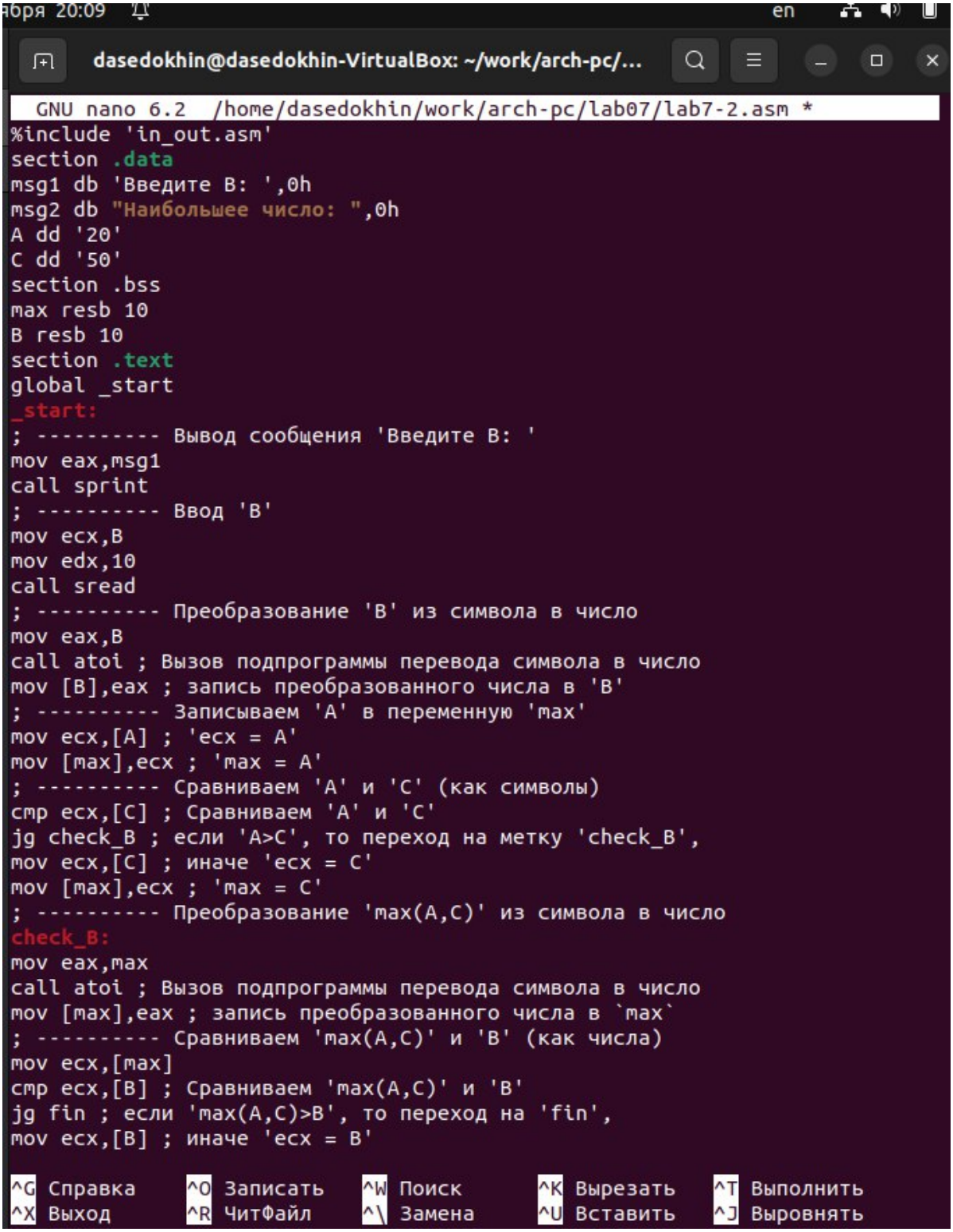

Рис. 2.9: Ввод текста программы из листинга 7.3 в lab7-2.asm

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

Рис. 2.10: Создание и проверка исполняемого файла

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

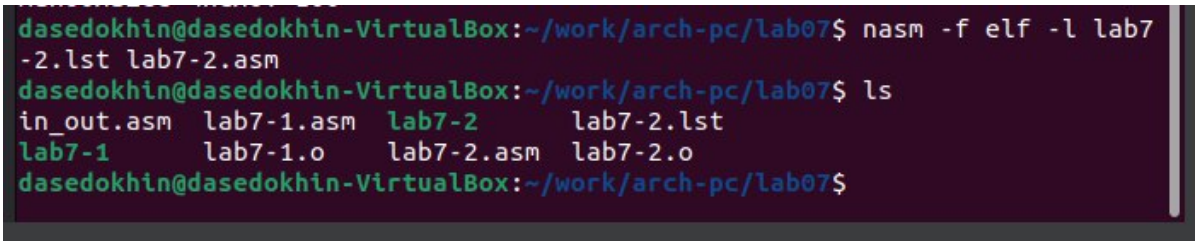

Рис. 2.11: Создание и проверка исполняемого файла

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

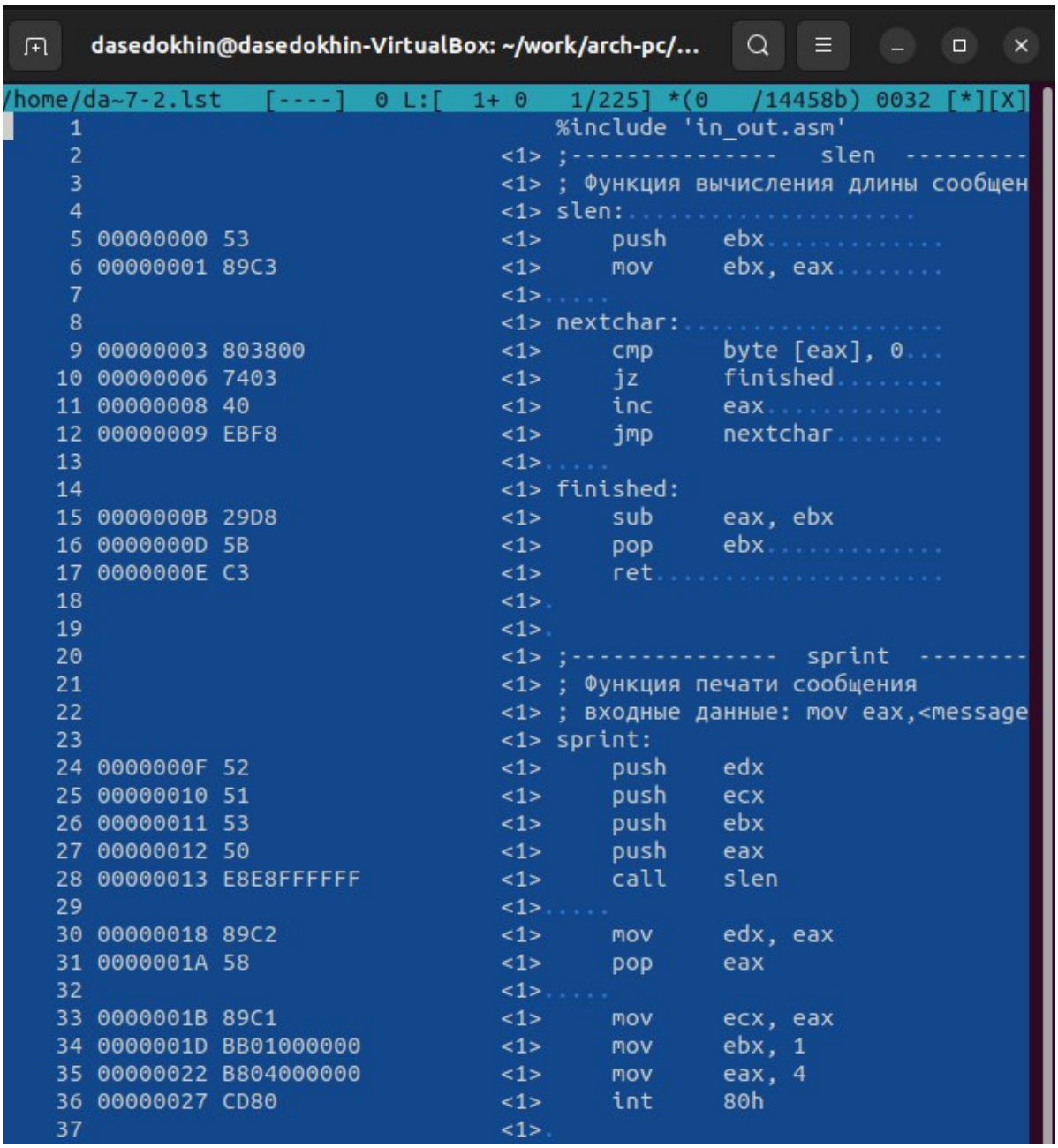

Рис. 2.12: Просмотра файла листинга lab7-2.lst

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

вызывается функция sprint. Она в свою очередь выводит собщение 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).

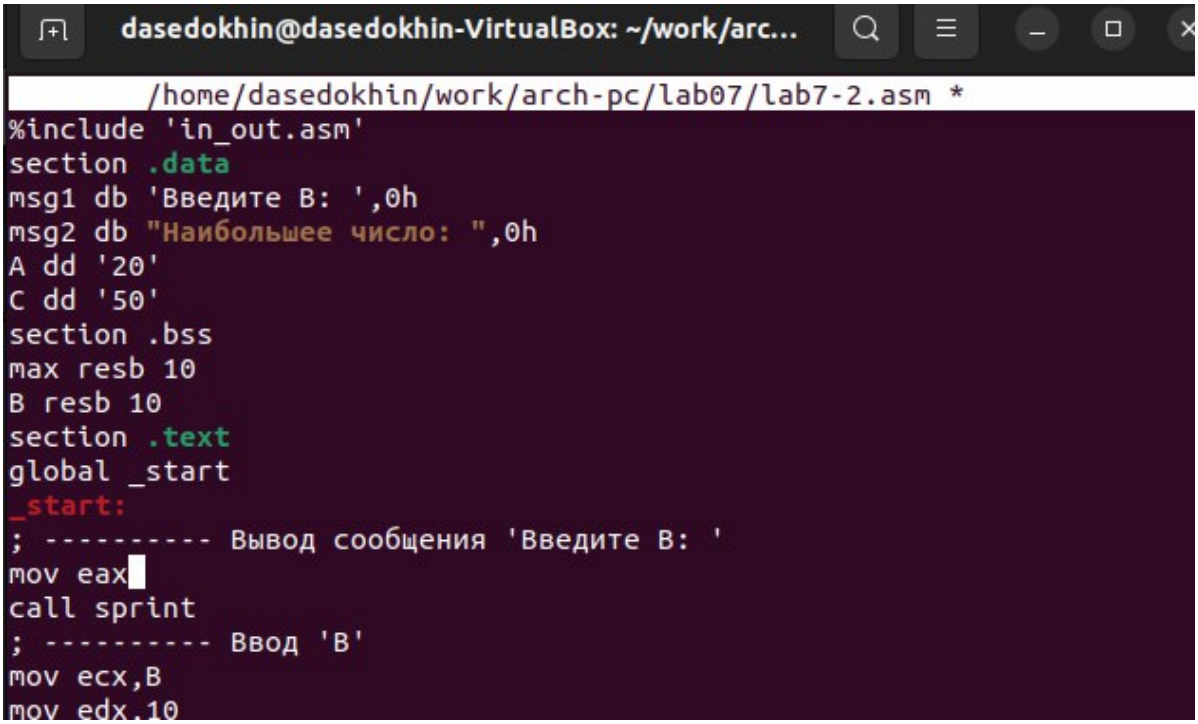

Рис. 2.14: Изменённая строка

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

Рис. 2.15: Изменения в созданном файле листинга

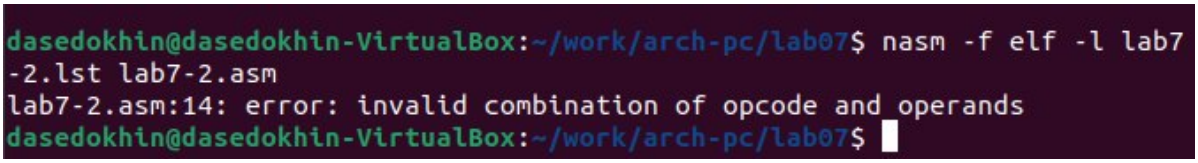

Рис. 2.16: Создание файла листинга

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).

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).

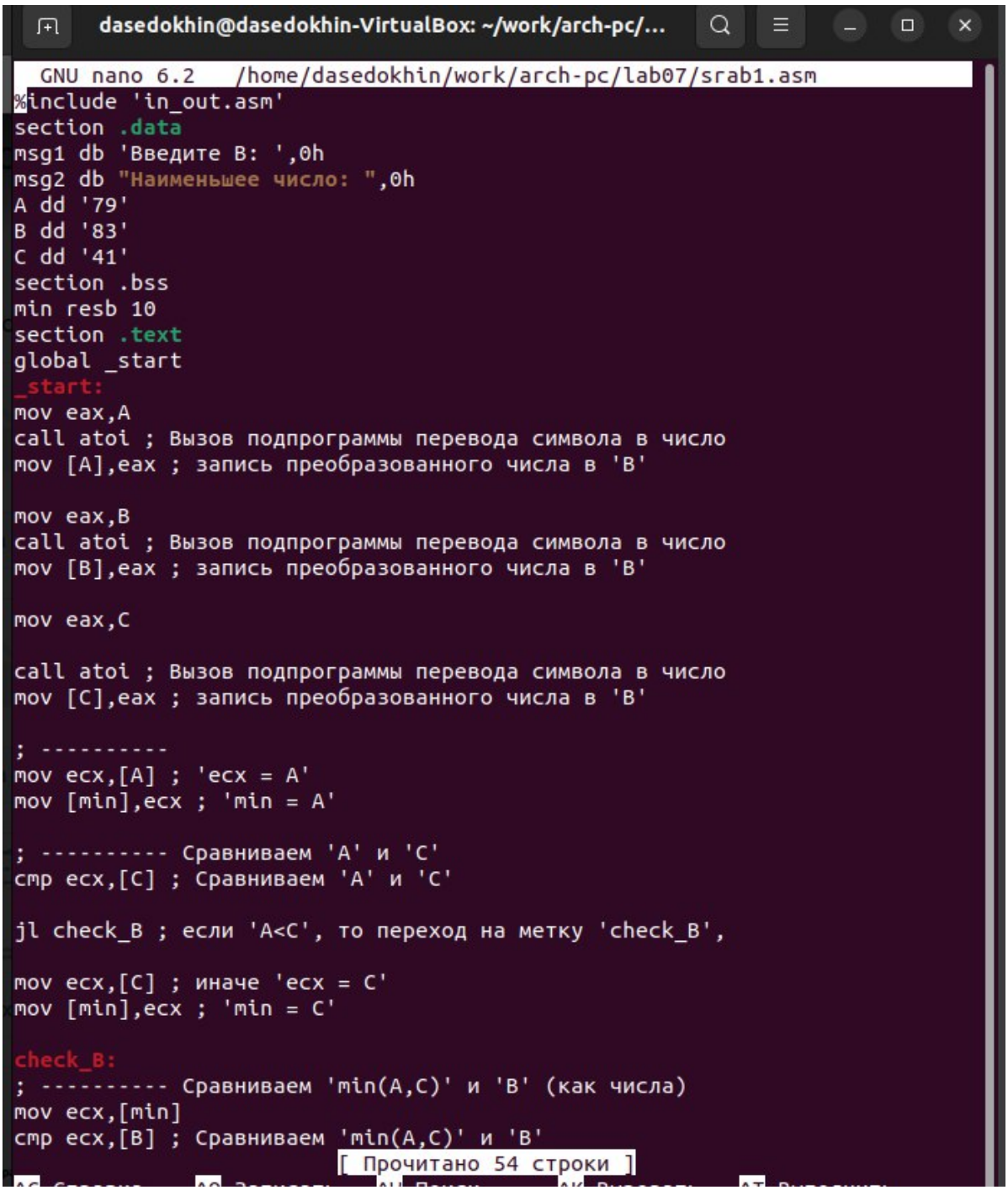

Рис. 3.2: Редактирование текста файла srab1.asm

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 sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход
```

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

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.