**Oтчёт по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

**Панявкина Ирина Васильевна**

**НКАбд-04-24**

Содержание

[1 Цель работы 3](#__RefHeading___Toc301_744737701)

[2 Задание 3](#__RefHeading___Toc303_744737701)

[3 Теоретическое введение 3](#__RefHeading___Toc305_744737701)

[4 Выполнение лабораторной работы 4](#__RefHeading___Toc307_744737701)

[4.1 Реализация переходов в NASM 4](#__RefHeading___Toc309_744737701)

[4.2 Изучение структуры файла листинга 10](#__RefHeading___Toc317_744737701_Copy_1)

[4.3        Выполнение заданий для самостоятельной работы 13](#__RefHeading___Toc694_1879274325)

[5 Выводы 21](#__RefHeading___Toc321_744737701)

[Список литературы 22](#__RefHeading___Toc323_744737701)

# 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

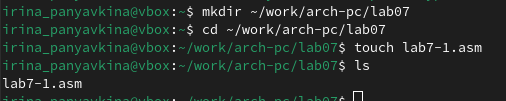
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

* условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
* безусловный переход – выполнение передачи управления в определенную точку про- граммы без каких-либо условий.

# 4 Выполнение лабораторной работы

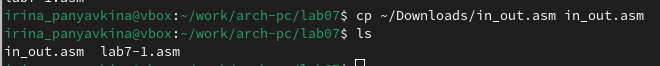
## 4.1 Реализация переходов в NASM

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



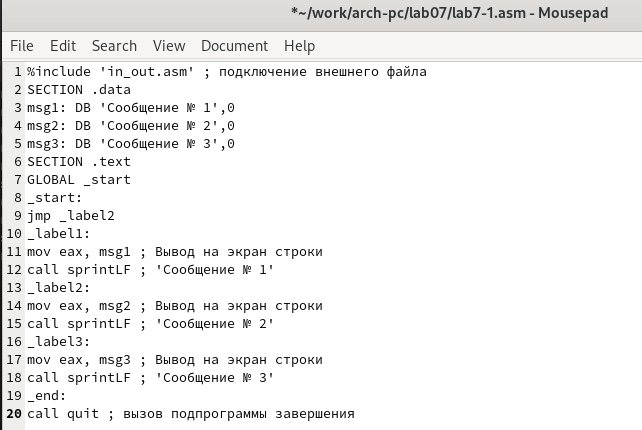
*Рис. 4.1: Создание каталога и файла для программы*

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



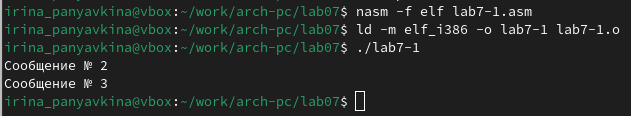
*Рис. 4.2: Создание копии файла*

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



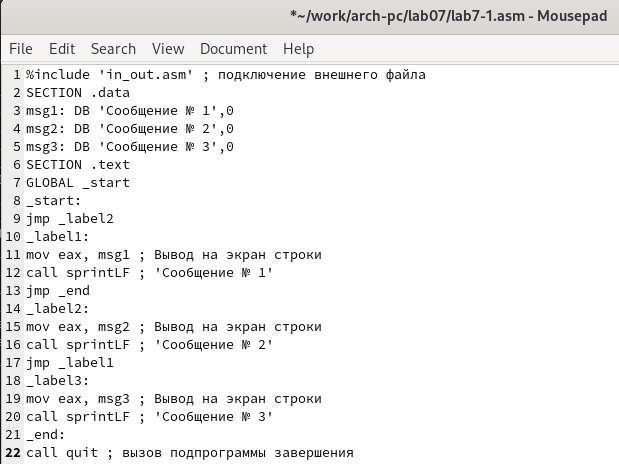
*Рис. 4.3: Редактирование файла и сохранение программы*

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



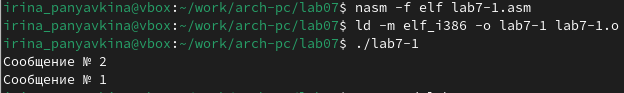
*Рис. 4.4: Запуск исполняемого файла*

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



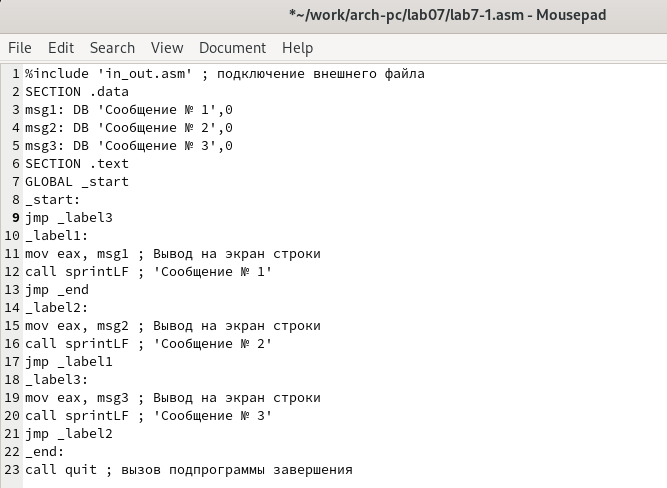
*Рис. 4.5: Редактирование файла*

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



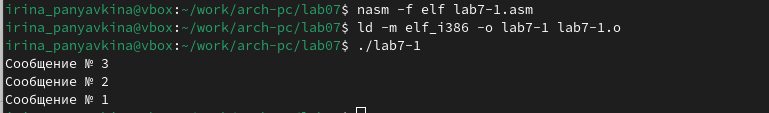
*Рис. 4.6: Запуск исполняемого файла (изменённой программы)*

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

**

*Рис. 4.7: Редактирование файла и сохранение программы*

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



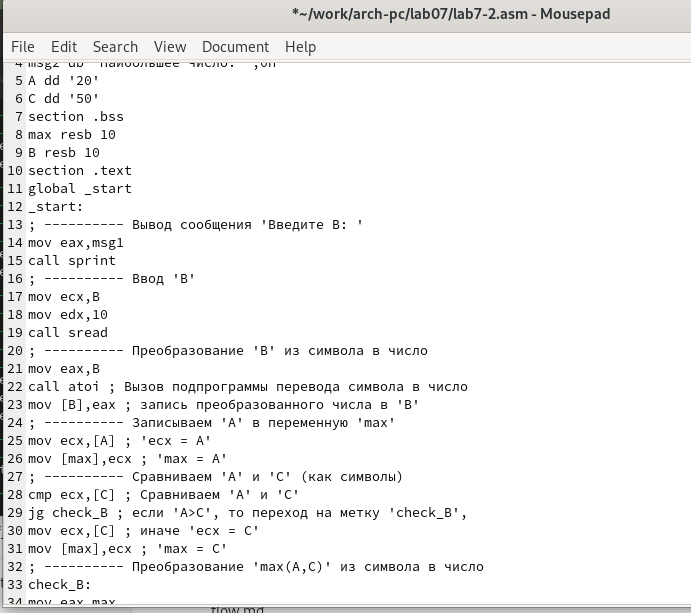
*Рис. 4.8: Запуск исполняемого файла (измененной программы)*

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



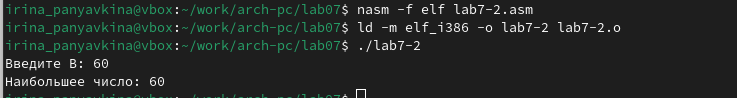
*Рис. 4.9: Создание файла*

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



*Рис. 4.10: Сохранение новой программы*

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

**

*Рис. 4.11: Запуск исполняемого файла*

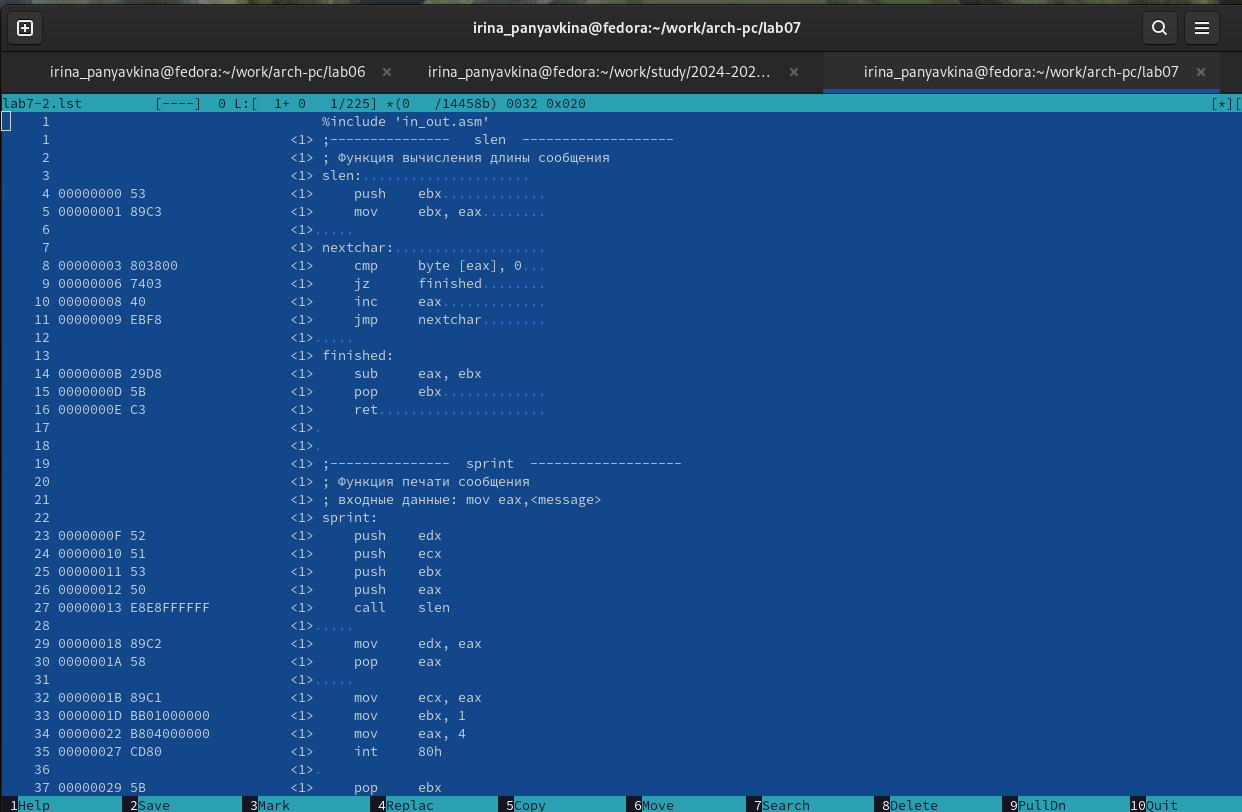
## 4.2 Изучение структуры файла листинга

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



*Рис. 4.12: Создание файла листинга*

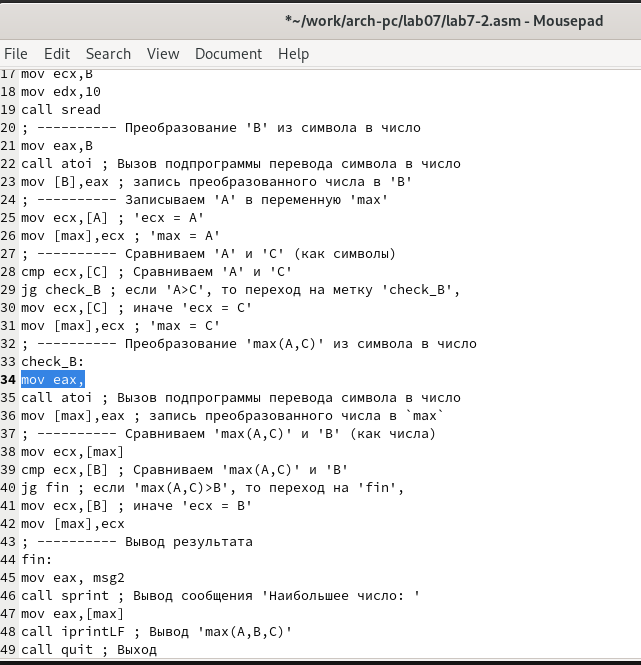
Открываю его с помощью текстового редактора mcedit (рис. 4.13).



*Рис. 4.13: Проверка файла листинга*

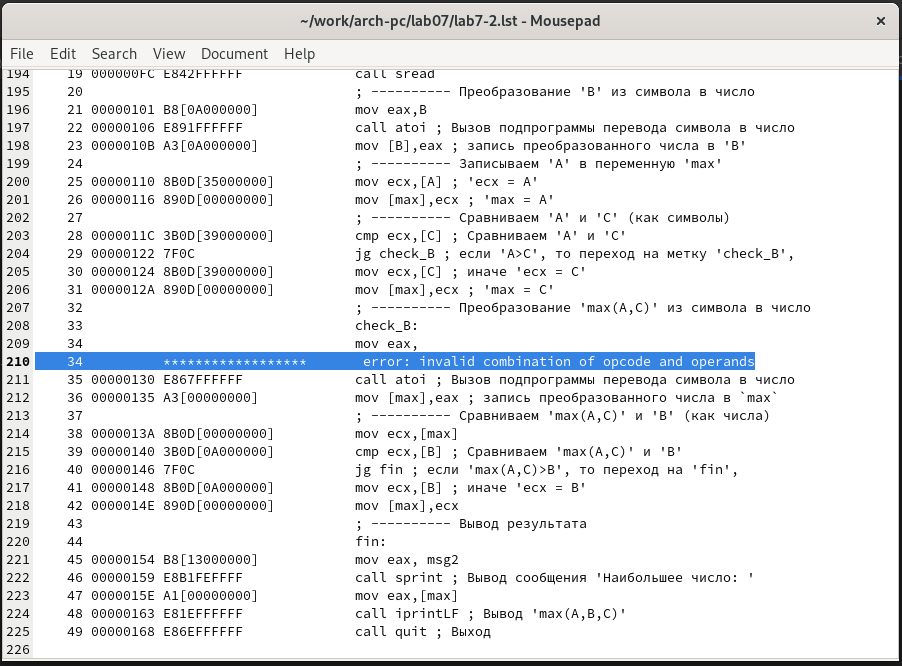
Первое значение в файле листинга - номер строки, и он может совсем не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем конечно же идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Открываю файл в текстовом редакторе mousepad, 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).



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

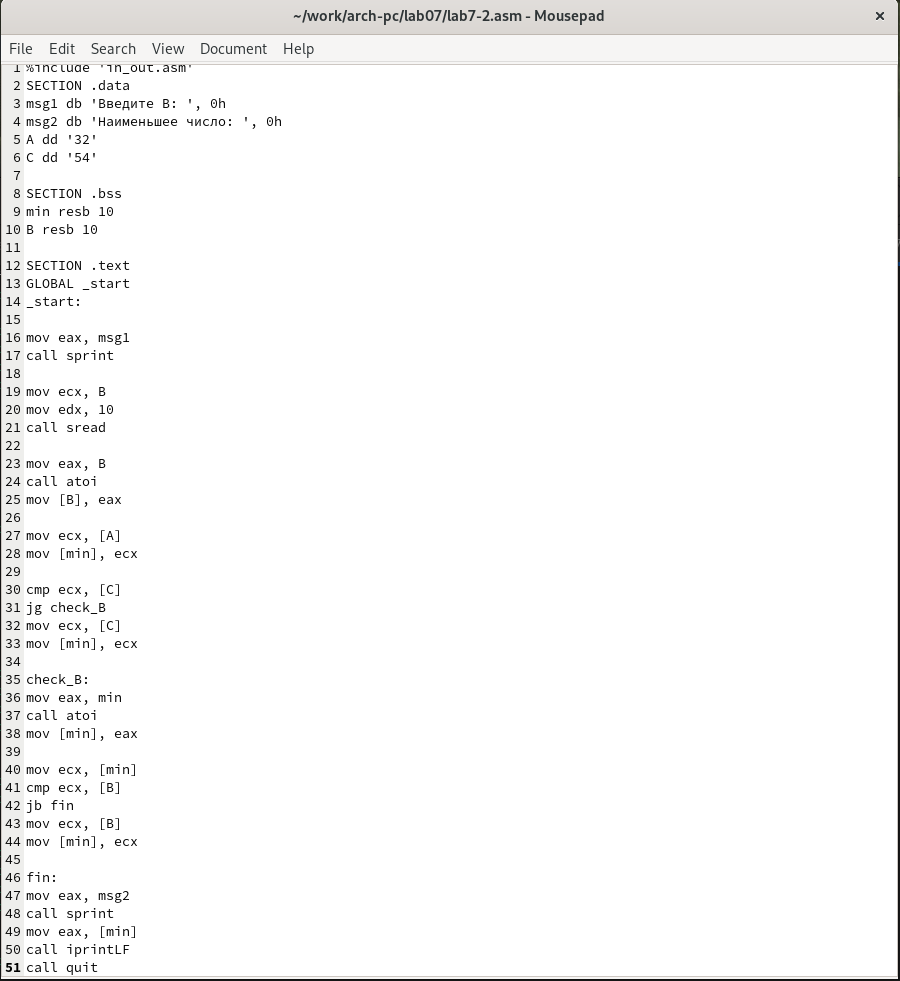
В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 4.15).



*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).



*Рис. 4.16: Первая программа самостоятельной работы*

Код первой программы:

%include 'in\_out.asm'

SECTION .data

msg1 db 'Введите В: ', 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

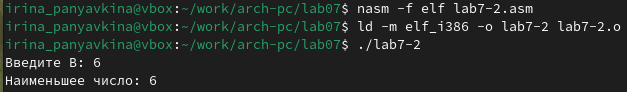
call sprint

mov eax, [min]

call iprintLF

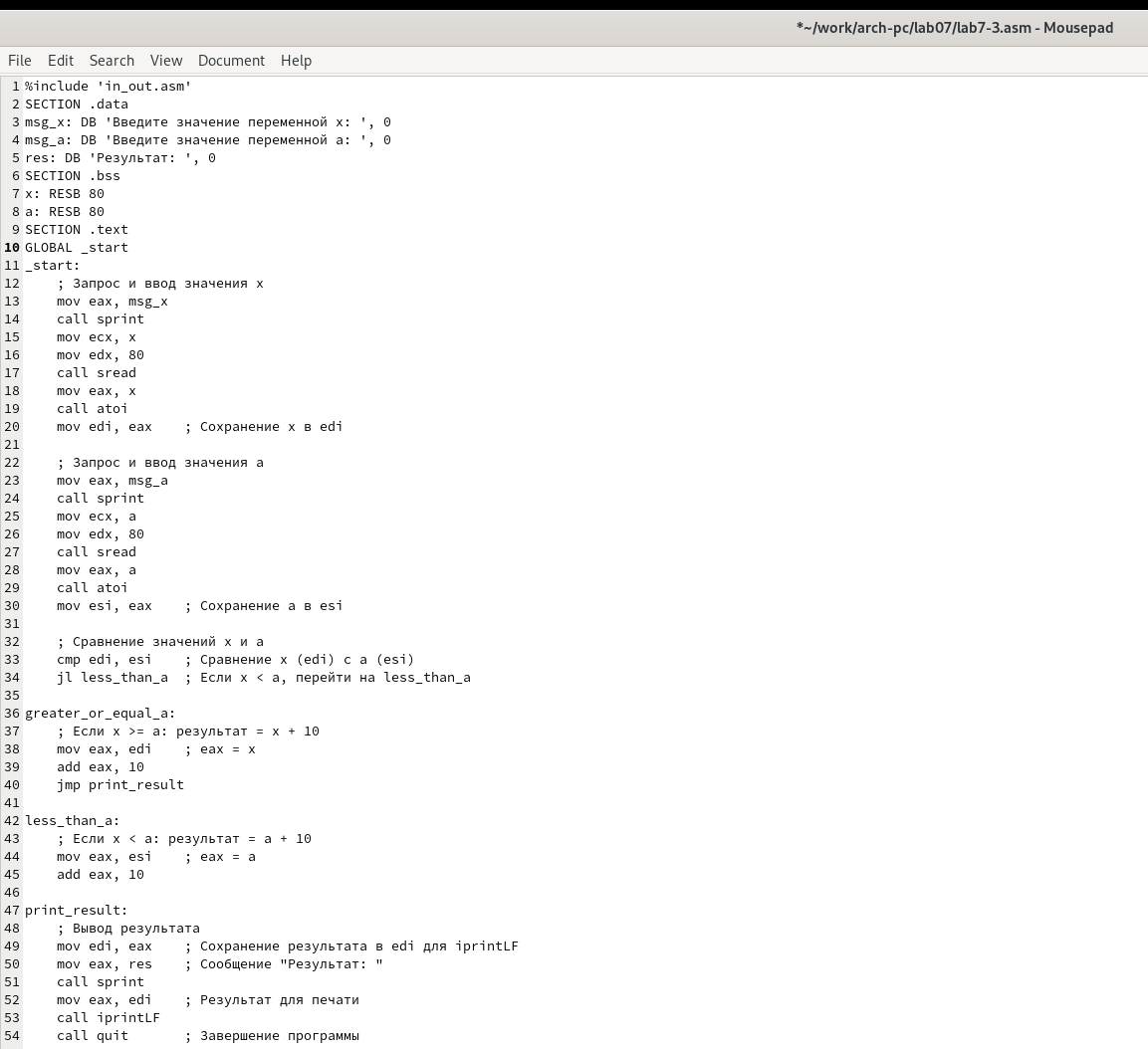
call quit

Проверяю корректность написания первой программы (рис. 4.17).



*Рис. 4.17: Запуск исполняемого файла*

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



*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: "

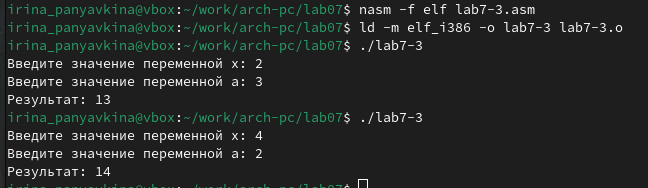
call sprint

mov eax, edi ; Результат для печати

call iprintLF

call quit ; Завершение программы

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



*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>.

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).