

Российский университет дружбы народов
Факультет физико-математических и естественных наук
Кафедра компьютерных и информационных наук

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5
Дисциплина: Архитектура компьютеров

Студент: *Алексеев Тимофей*
Группа: НК Абд-07-25

Москва
2025 г.

Содержание

1.	Цель работы	5
2.	Задание	6
3.	Теоретическое введение	7
4.	Выполнение лабораторной работы	8
4.1	Основы работы с Midnight Commander	8
4.2	Работа в NASM	12
4.3	Подключение внешнего файла	14
4.4	Задание для самостоятельной работы	18
5.	Выводы	22
6.	Список литературы	23

Список иллюстраций

4.1 Открытие Midnight Commander	8
4.2 Интерфейс Midnight Commander	9
4.3 Открытый каталог arch-pc	10
4.4 Создание рабочего подкаталога	10
4.5 Создание файла в Midnight Commander	11
4.6 Редактирование файла в Midnight Commander	12
4.7 Проверка сохранения сделанных изменений	13
4.8 Трансляция, компоновка и последующий запуск программы	14
4.9 Копирование файла в рабочий каталог	14
4.10 Создание копии файла в Midnight Commander	15
4.11 Изменение программы	16
4.12 Запуск измененной программы	17
4.13 Запуск изменной программы с другой подпрограммой	18
4.14 Редактирование копии	19
4.15 Запуск своей программы	19
4.16 Редактирование копии	20
4.17 Запуск своей программы	21

Список таблиц

1. Цель работы

Целью данной лабораторной работы является приобретение практических навыков работы в Midnight Commander, освоение инструкций языка ассемблера `mov` и `int`.

2. Задание

1. Основы работы с тс
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла
4. Выполнение заданий для самостоятельной работы

3. Теоретическое введение

Midnight Commander (или просто mc) — это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системой, т.е. mc является файловым менеджером. Midnight Commander позволяет сделать работу с файлами более удобной и наглядной. Программа на языке ассемблера NASM, как правило, состоит из трёх секций: секция кода программы (SECTION .text), секция инициированных (известных во время компиляции) данных (SECTION .data) и секция неинициализированных данных (тех, под которые во время компиляции только отводится память, а значение присваивается в ходе выполнения программы) (SECTION .bss). Для объявления инициированных данных в секции .data используются директивы DB, DW, DD, DQ и DT, которые резервируют память и указывают, какие значения должны храниться в этой памяти: - DB (define byte) — определяет переменную размером в 1 байт; - DW (define word) — определяет переменную размером в 2 байта (слово); - DD (define double word) — определяет переменную размером в 4 байта (двойное слово); - DQ (define quad word) — определяет переменную размером в 8 байт (учетверенное слово); - DT (define ten bytes) — определяет переменную размером в 10 байт. Директивы используются для объявления простых переменных и для объявления массивов. Для определения строк принято использовать директиву DB в связи с особенностями хранения данных в оперативной памяти. Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике.

`mov dst,src`

Здесь операнд dst — приёмник, а src — источник. В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и непосредственные значения (const). Инструкция языка ассемблера int предназначена для вызова прерывания с указанным номером.

`int n`

Здесь n — номер прерывания, принадлежащий диапазону 0–255. При программировании в Linux с использованием вызовов ядра sys_calls n=80h (принято задавать в шестнадцатеричной системе счисления).

4. Выполнение лабораторной работы

4.1 Основы работы с Midnight Commander

Введя соответствующую команду в терминале (рис. 4.1), я открываю Midnight Commander (рис. 4.2).

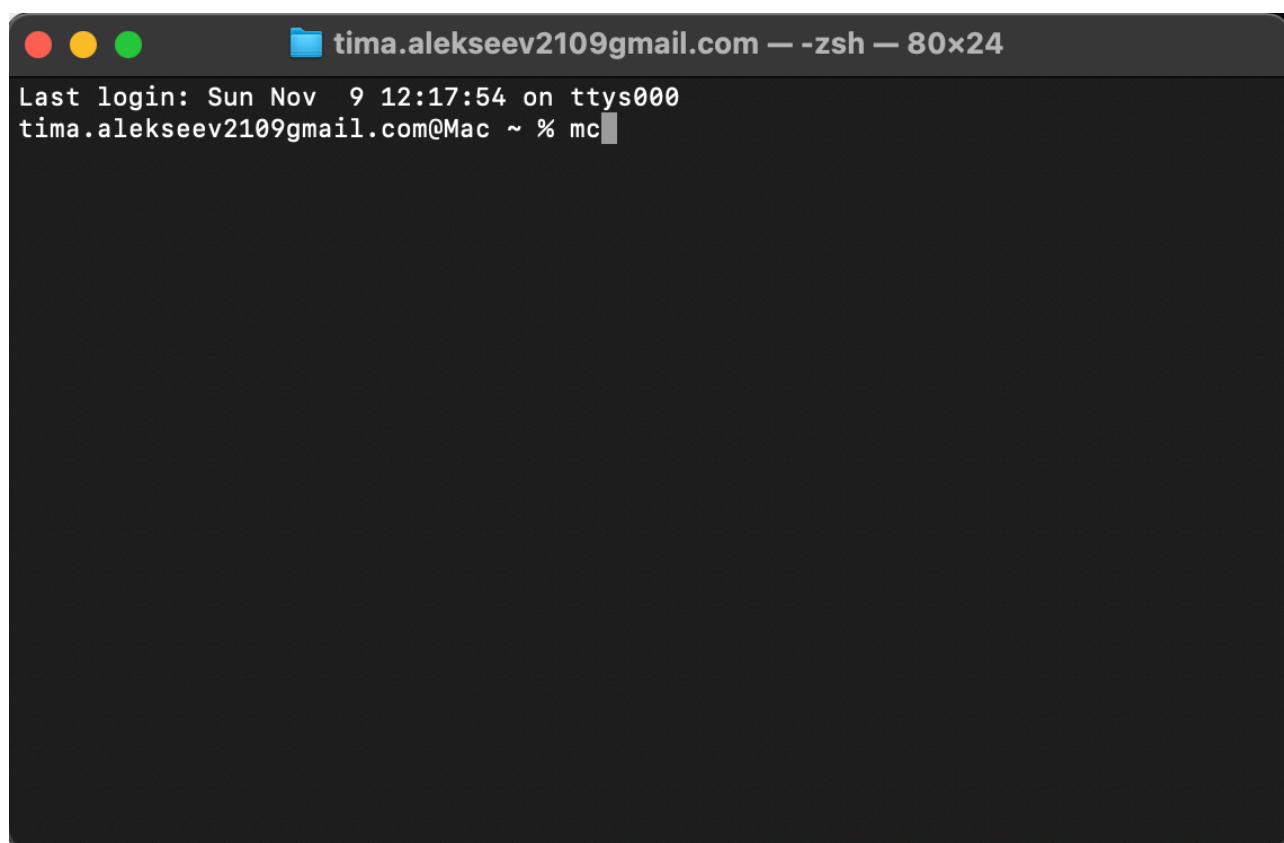


Рис. 4.1: Открытие Midnight Commander

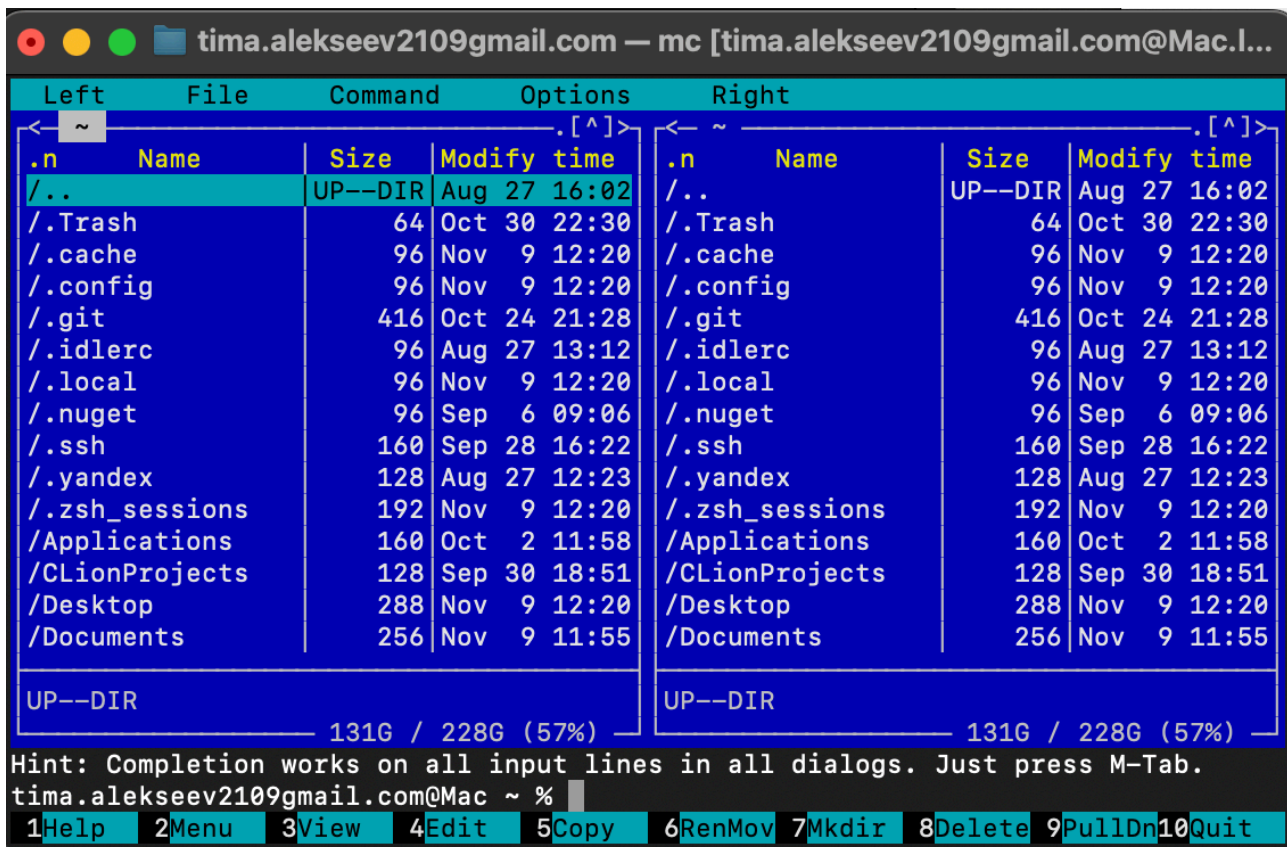


Рис. 4.2: Интерфейс Midnight Commander

Перехожу в созданный каталог в предыдущей лабораторной работе (рис. 4.3).

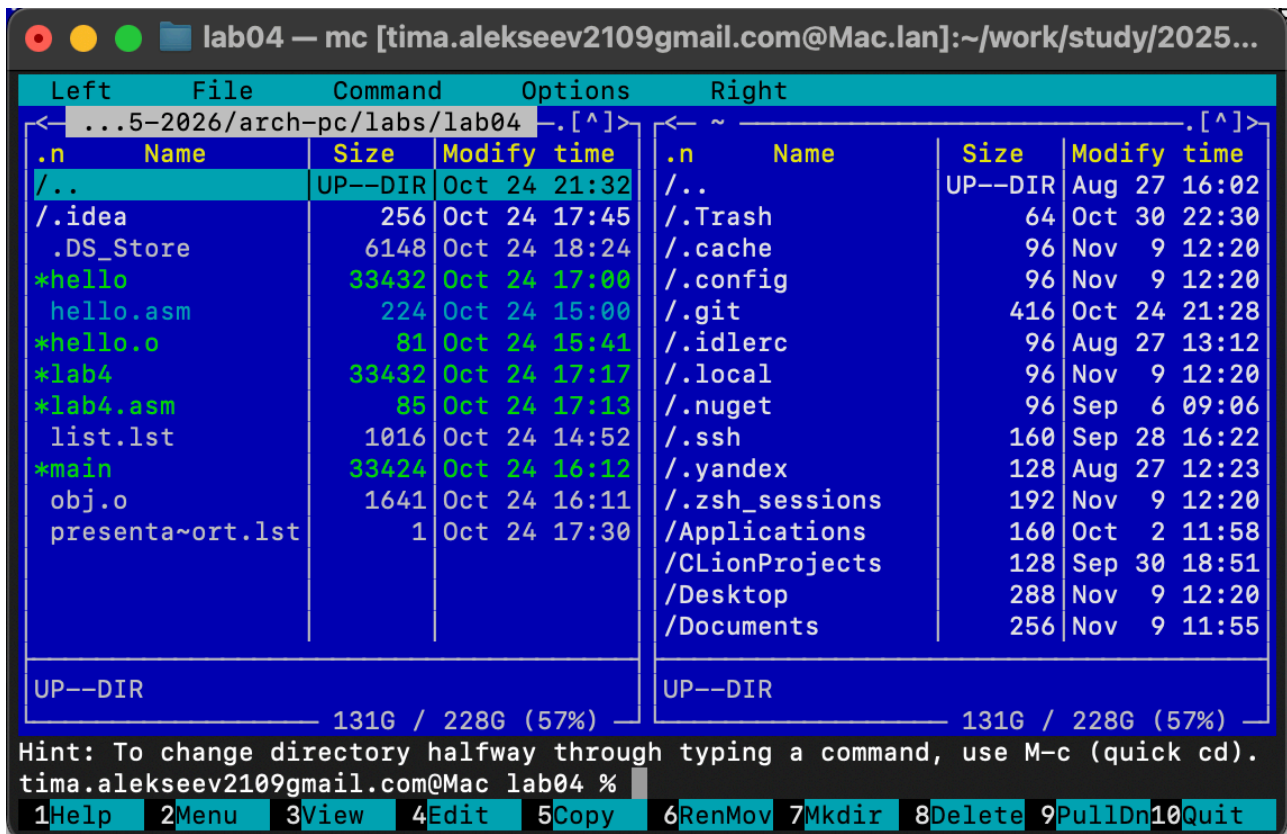


Рис. 4.3: Открытый каталог arch-pc

С помощью функциональной клавиши, я создаю подкаталог lab05, в котором буду работать (рис. 4.4).

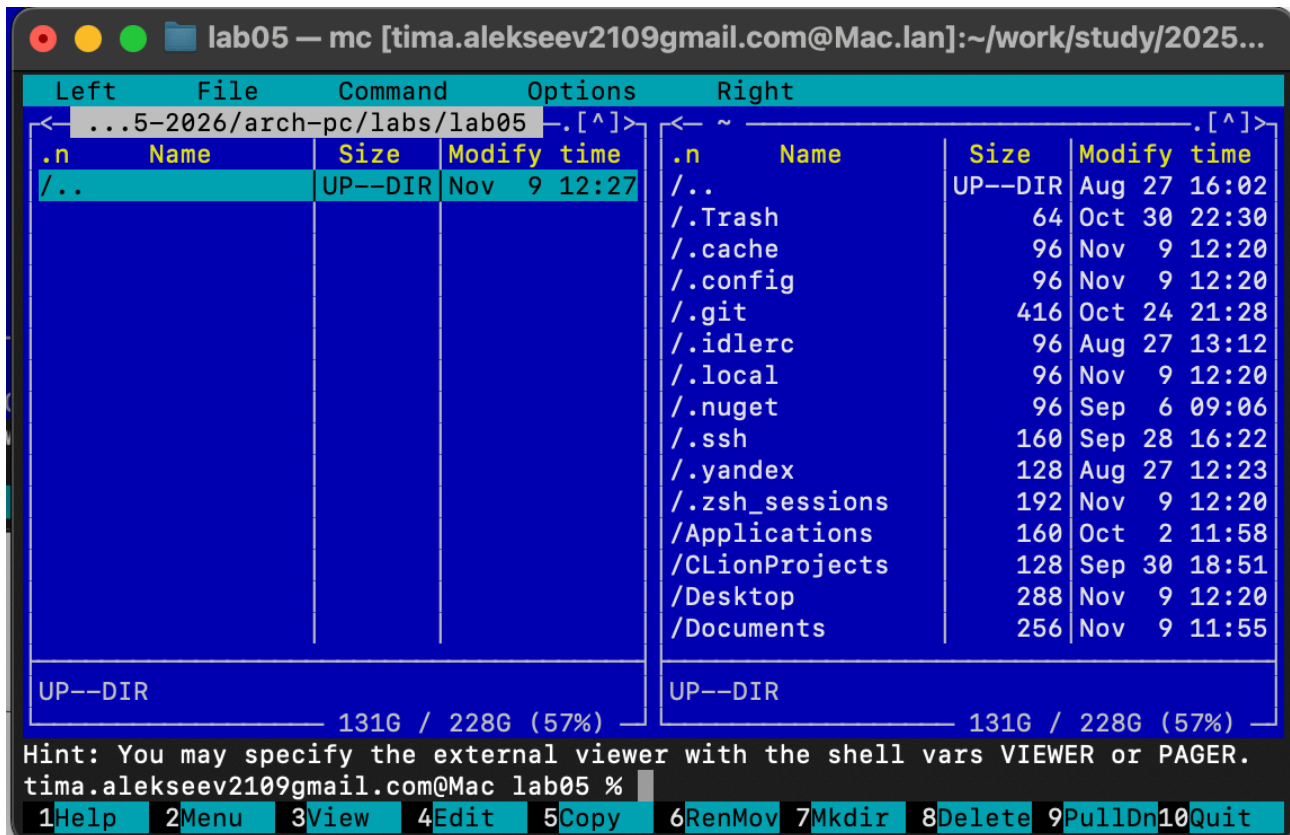


Рис. 4.4: Создание рабочего подкаталога

В строке ввода вводжу команду `touch` и создаю файл (рис. 4.5).

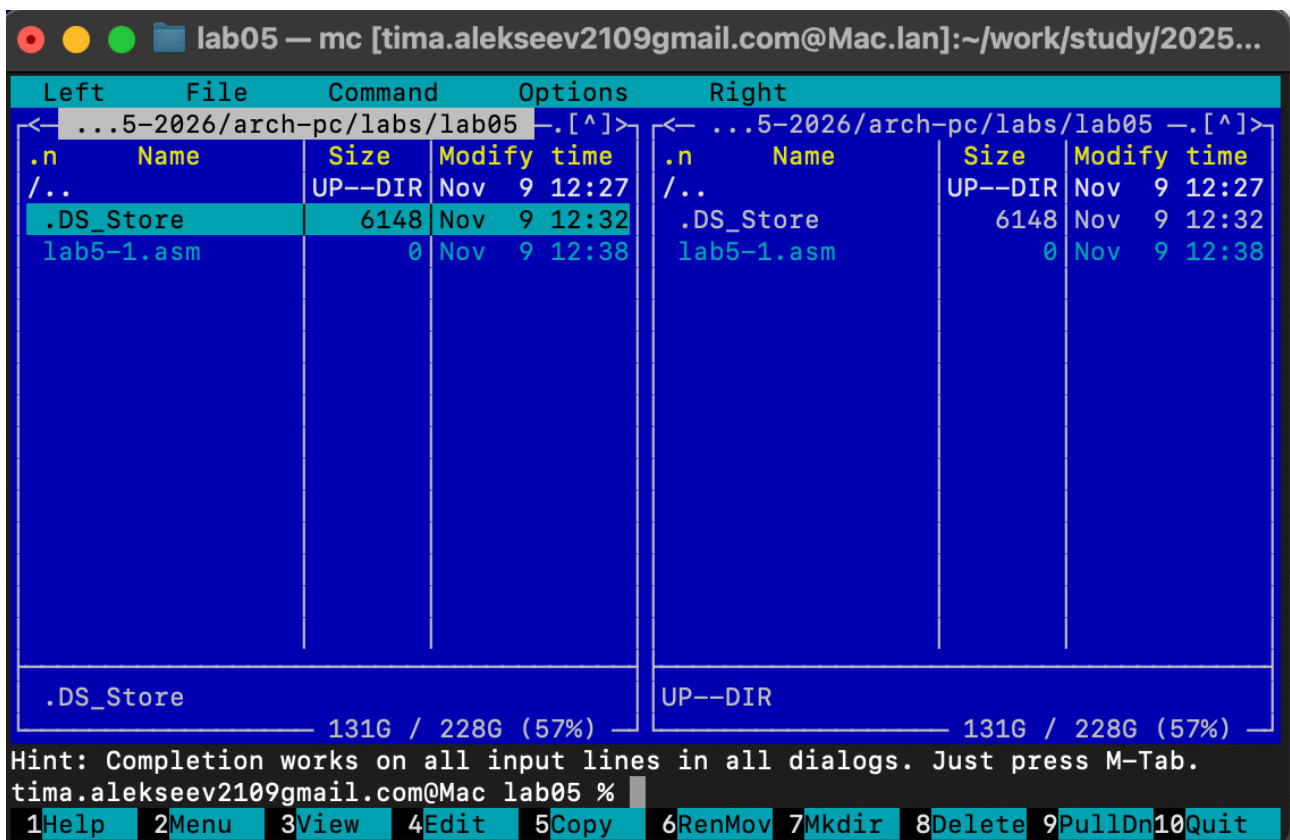
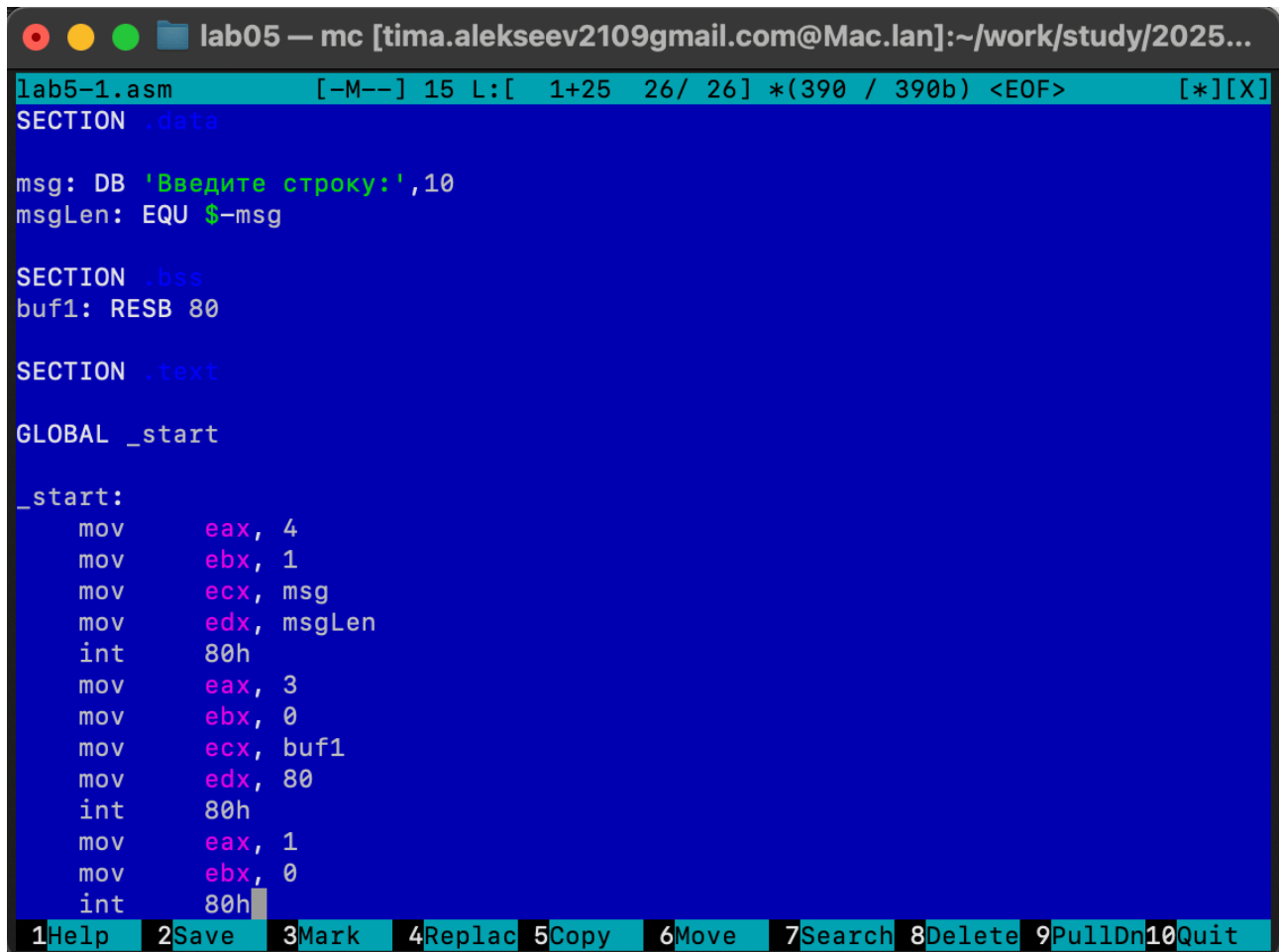


Рис. 4.5: Создание файла в Midnight Commander

4.2 Работа в NASM

С помощью F4 открываю только что созданный файл и вношу код с листинга (рис. 4.6).



```
lab5-1.asm      [-M--] 15 L:[ 1+25 26/ 26] *(390 / 390b) <EOF>      [*][X]
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text

GLOBAL _start

_start:
    mov     eax, 4
    mov     ebx, 1
    mov     ecx, msg
    mov     edx, msgLen
    int     80h
    mov     eax, 3
    mov     ebx, 0
    mov     ecx, buf1
    mov     edx, 80
    int     80h
    mov     eax, 1
    mov     ebx, 0
    int     80h
```

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit

Рис. 4.6: Редактирование файла в Midnight Commander

Проверяю сохраненные изменения с помощью клавиши F3 (рис. 4.7).

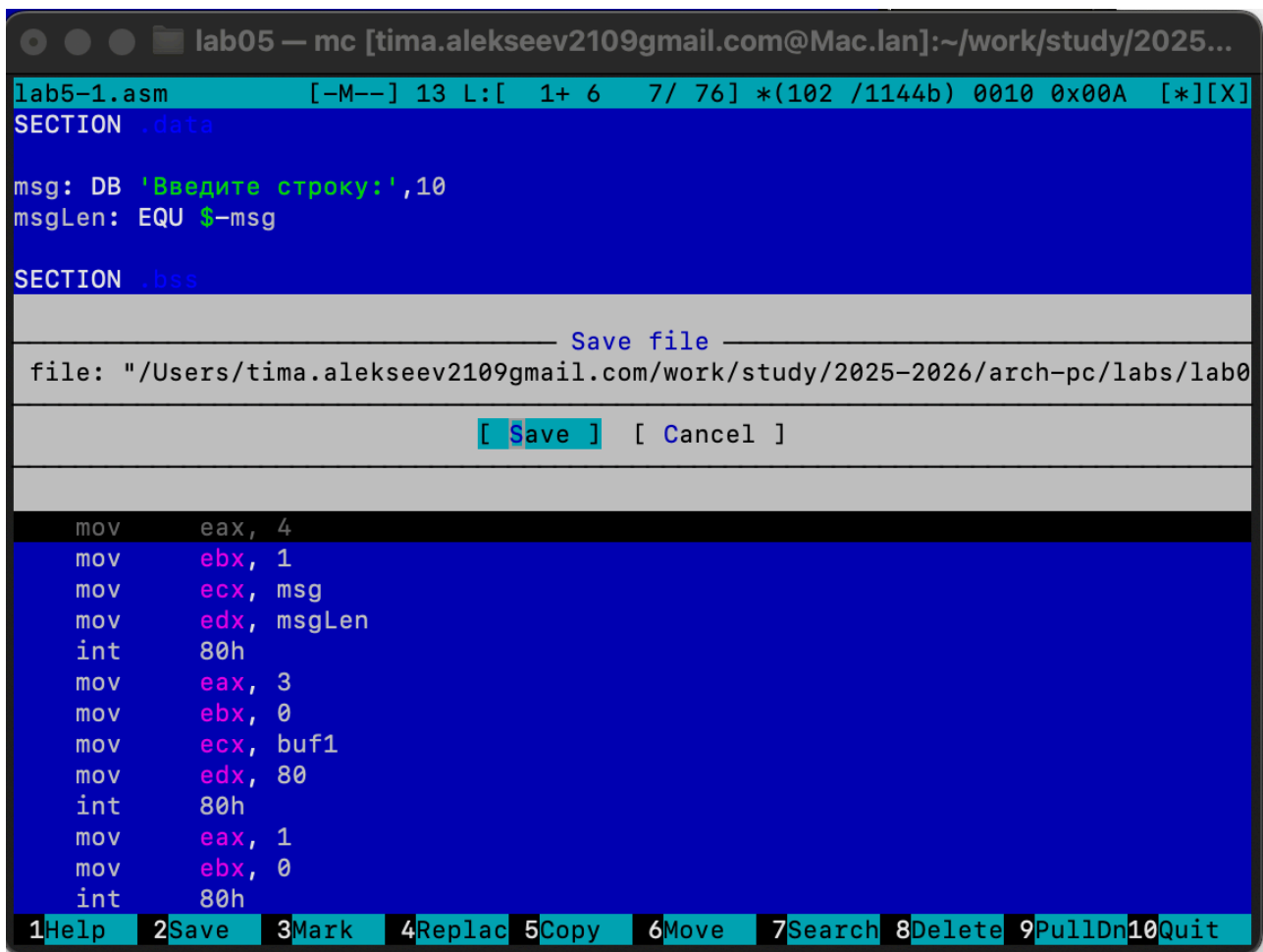


Рис. 4.7: Сохранение сделанных изменений

Транслирую и компоную измененный файл, запуская (рис. 4.8).

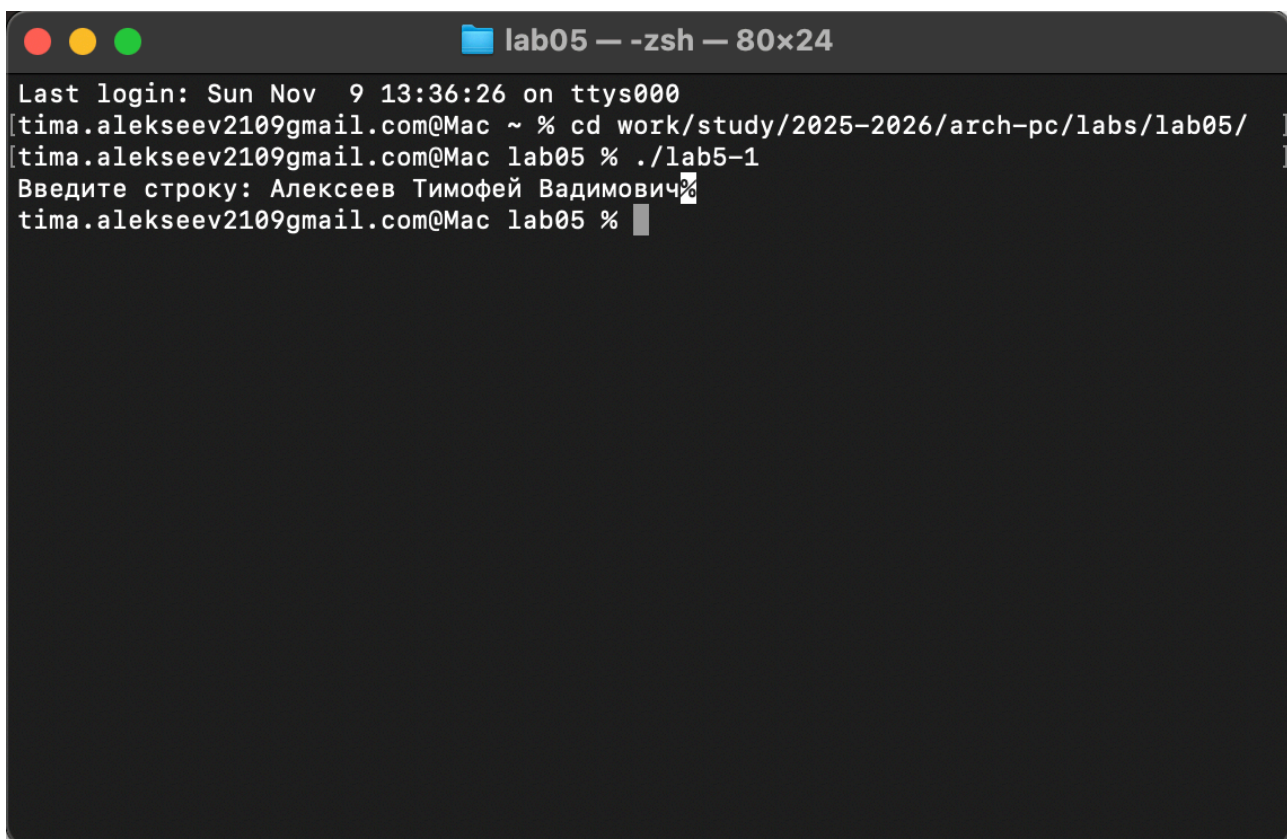


Рис. 4.8: Трансляция, компоновка и последующий запуск программы

4.3 Подключение внешнего файла

Скачанный с ТУИС файл сохраняю в общую папку на своем компьютере, на виртуальной машине в интерфейсе Midnight Commander перехожу в директорию общей папки, копирую файл в рабочий подкаталог. (рис. 4.9).

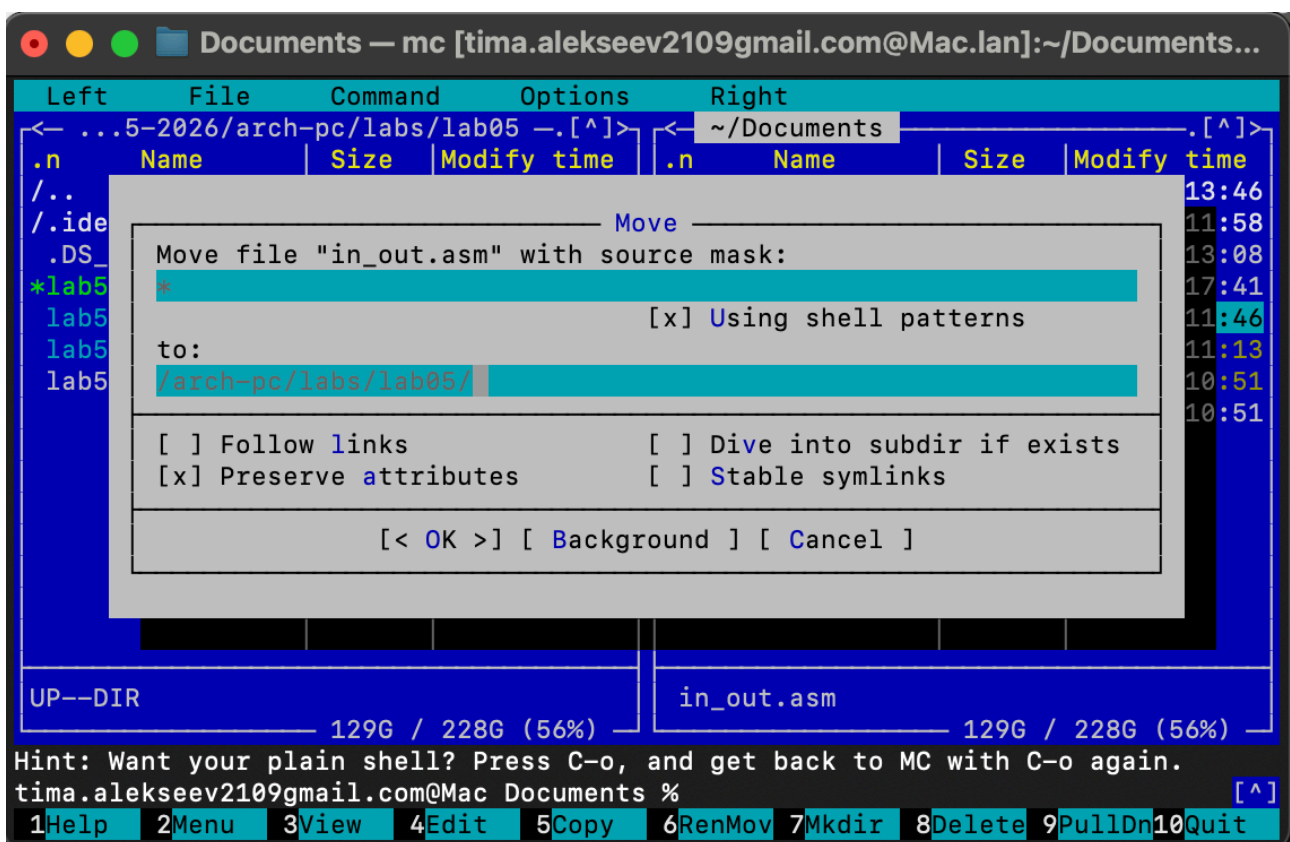


Рис. 4.9: Копирование файла в рабочий каталог

Создаю копию файла для последующей работы с ним (рис. 4.10).

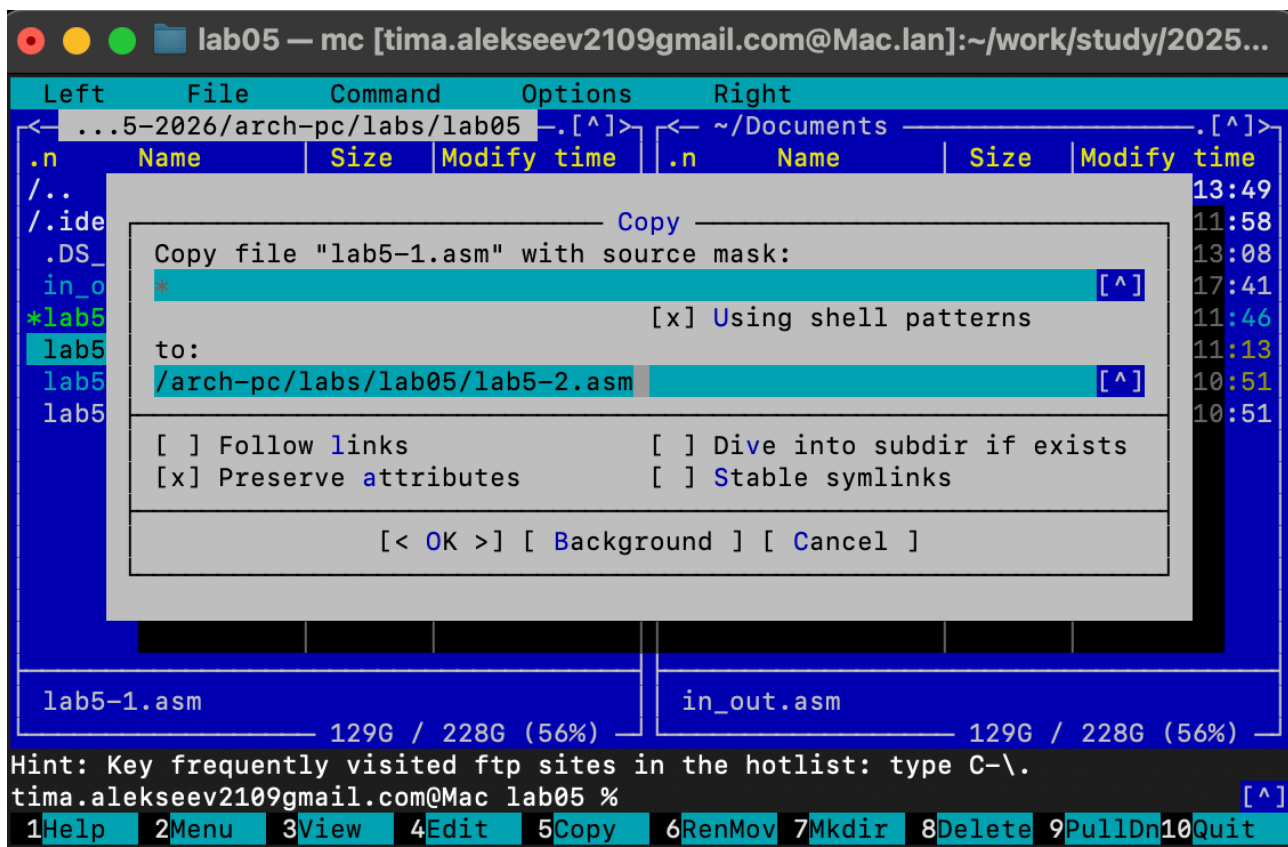



Рис. 4.10: Создание копии файла в Midnight Commander

В копии файла подключаю подпрограмм из подключенного файла (рис. 4.11).

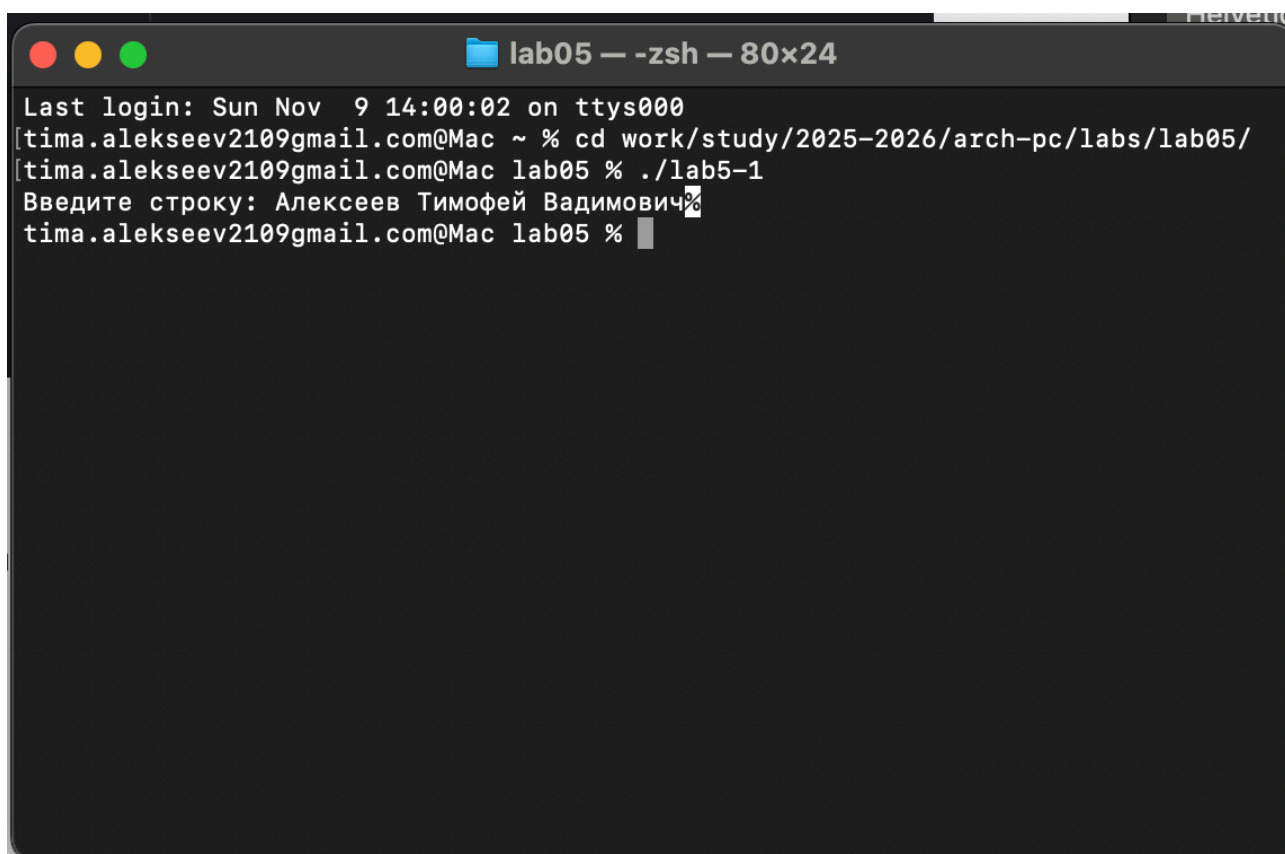

```

1      %include 'in_out.asm'
2
3      SECTION .data
4
5      msg: DB 'Введите строку: ', 0h
6      msgLen: EQU $-msg
7
8      SECTION .bss
9      buf1: RESB 80
10
11     SECTION .text
12         GLOBAL _start
13         _start:
14             mov     eax, msg
15             call    sprintLF
16
17      mov     ecx, buf1
18             mov     edx, 80
19
20             call    sread
21
22             call    quit

```

Рис. 4.11: Изменение программы

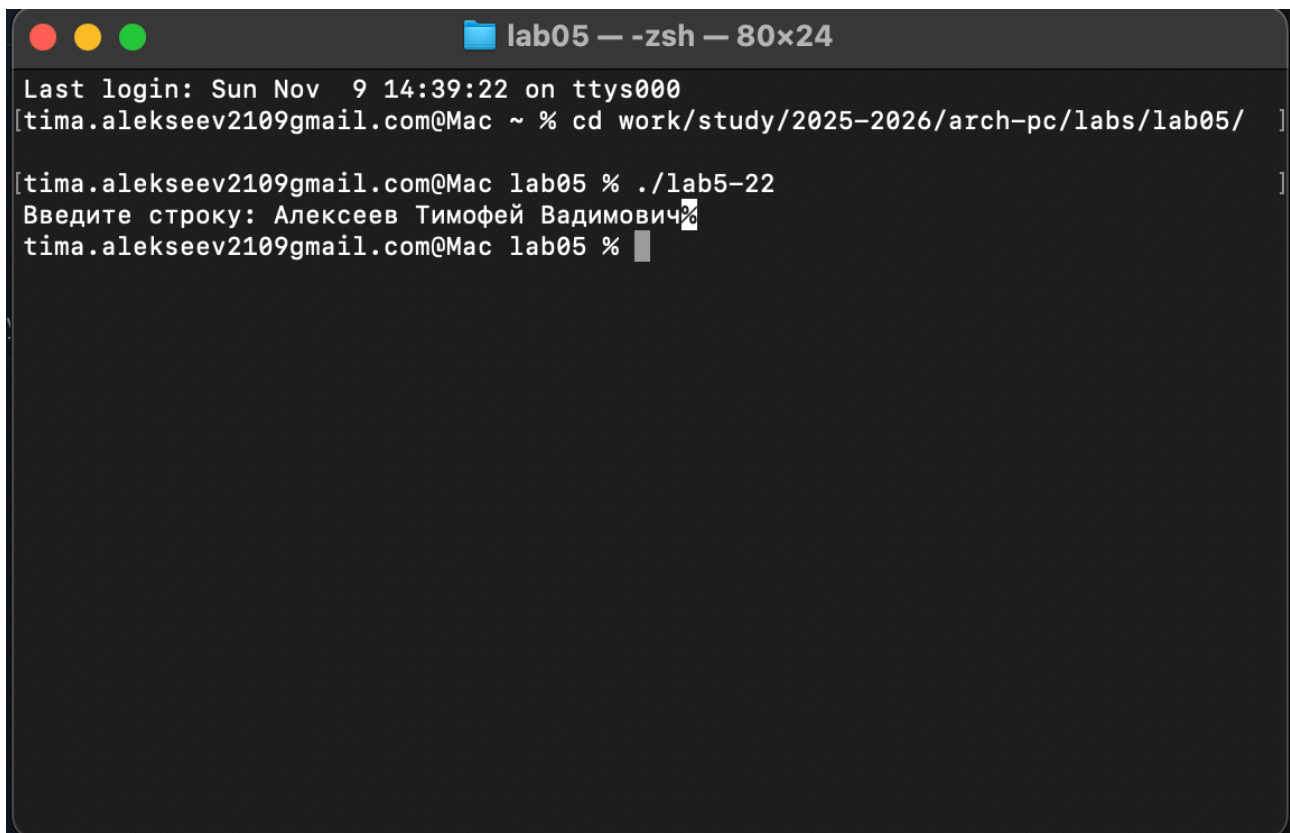
Транслирую, компоную и запускаю программу с подключенным файлом (рис. 4.12).

A terminal window titled "lab05 — -zsh — 80x24" with standard macOS window controls (red, yellow, green buttons). The terminal output shows a successful login, a directory change to "work/study/2025-2026/arch-pc/labs/lab05/", and the execution of a script "lab5-1". The script prompts for a name in Russian: "Введите строку: Алексеев Тимофей Вадимович". The user has entered the name, and the prompt is now waiting for the next command.

```
lab05 — -zsh — 80x24
Last login: Sun Nov  9 14:00:02 on ttys000
[tima.alekseev2109gmail.com@Mac ~ % cd work/study/2025-2026/arch-pc/labs/lab05/
[tima.alekseev2109gmail.com@Mac lab05 % ./lab5-1
Введите строку: Алексеев Тимофей Вадимович%
tima.alekseev2109gmail.com@Mac lab05 %
```

Рис. 4.12: Запуск измененной программы

4.4 Задания для самостоятельной работы



```
lab05 — -zsh — 80x24
Last login: Sun Nov  9 14:39:22 on ttys000
tima.alekseev2109gmail.com@Mac ~ % cd work/study/2025-2026/arch-pc/labs/lab05/
tima.alekseev2109gmail.com@Mac lab05 % ./lab5-22
Введите строку: Алексеев Тимофей Вадимович%
tima.alekseev2109gmail.com@Mac lab05 %
```

Рис. 4.13: Запуск изменной программы с другой подпрограммой

Создаю копию lab5-1.asm, редактирую так, чтобы в конце выводилась введенная мною строка с клавиатуры (рис. 4.14).

```

SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text

GLOBAL _start

_start:
    mov     eax, 4
    mov     ebx, 1
    mov     ecx, msg
    mov     edx, msgLen
    int     80h
    mov     eax, 3
    mov     ebx, 0
    mov     ecx, buf1
    mov     edx, 80
    int     80h
    mov     eax, 4
    mov     ebx, 1
    mov     ecx, buf1
    mov     edx, buf1
    int     80h
    mov     eax, 1
    mov     ebx, 0
    int     80h

```

Рис. 4.14: Редактирование копии

Компоную, Запускаю свою программу (рис. 4.15).

```

lab05 — -zsh — 80x24
Last login: Sun Nov  9 14:55:17 on ttys000
tima.alekseev2109gmail.com@Mac ~ % cd work/study/2025-2026/arch-pc/labs/lab05/

tima.alekseev2109gmail.com@Mac lab05 % gcc lab5-1copy.c -o lab5-1copy

[tima.alekseev2109gmail.com@Mac lab05 % ./lab5-1copy.c
./lab5-1copy.c: line 4: syntax error near unexpected token `('
./lab5-1copy.c: line 4: `int main() {'
tima.alekseev2109gmail.com@Mac lab05 % ./lab5-1copy

Введите строку:
Алексеев Тимофей
Алексеев Тимофей
tima.alekseev2109gmail.com@Mac lab05 % █

```

Код прикладываю

```
SECTION .data
msg: DB 'Введите строку:', 10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, 4
mov ebx, 1
mov ecx, msg
mov edx, msgLen
int 80h
mov eax, 3
mov ebx, 0
mov ecx, buf1
mov edx, 80
int 80h
mov eax, 4
mov ebx, 1
mov ecx, buf1
mov edx, buf1
int 80h
mov eax, 1
mov ebx, 0
int 80h
```

Создаю копию lab5-2.asm, редактирую так, чтобы в конце выводилась введенная мною строка с клавиатуры (рис. 4.16).

```
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите строку: ', 0h
msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

    mov eax, msg
    call sprint

    mov ecx, buf1
    mov edx, 80

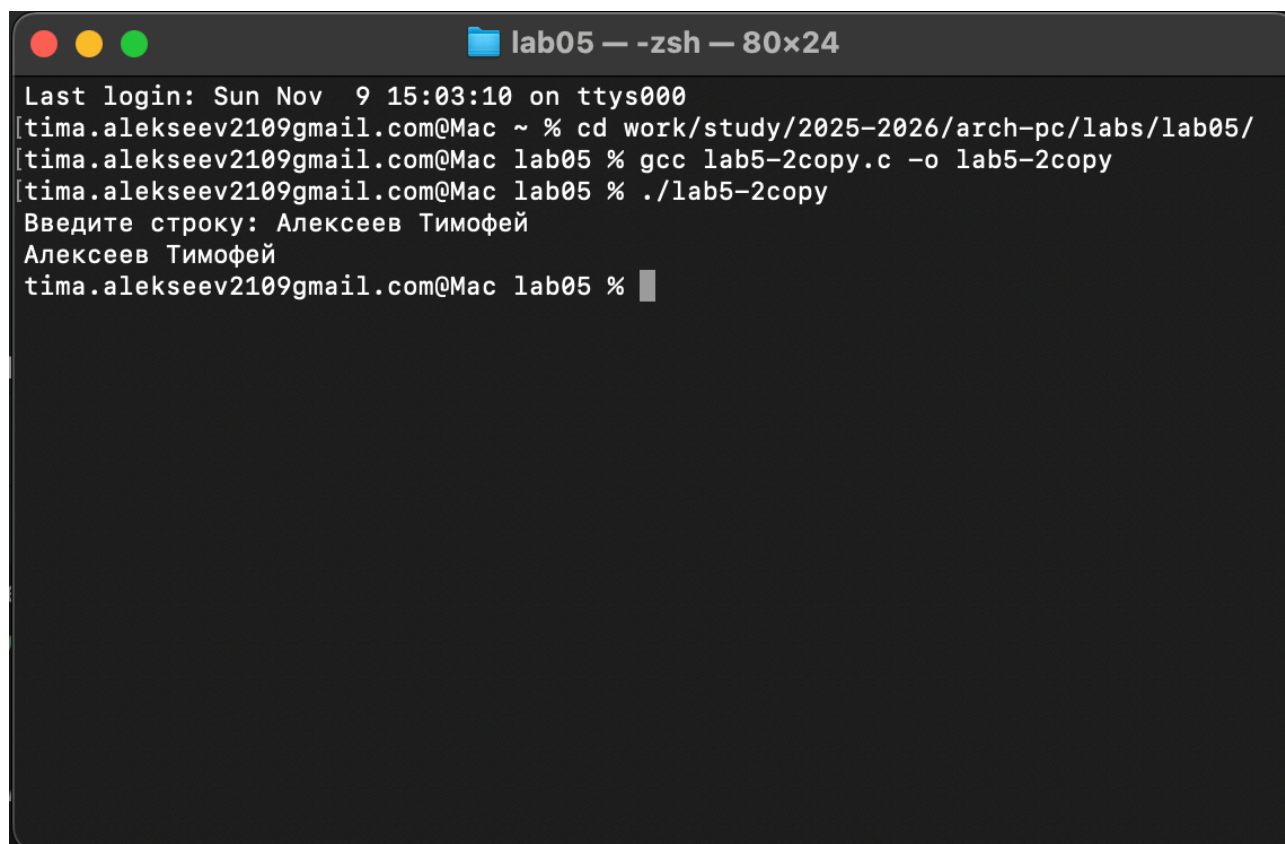
    call sread

    mov eax, 4
    mov ebx, 1
    mov ecx, buf1
    int 80h

    call quit
```

Рис. 4.16: Редактирование копии

Транслирую, компонирую и запускаю свою программу (рис. 4.17).



```
lab05 — -zsh — 80x24
Last login: Sun Nov 9 15:03:10 on ttys000
tima.alekseev2109gmail.com@Mac ~ % cd work/study/2025-2026/arch-pc/labs/lab05/
tima.alekseev2109gmail.com@Mac lab05 % gcc lab5-2copy.c -o lab5-2copy
tima.alekseev2109gmail.com@Mac lab05 % ./lab5-2copy
Введите строку: Алексеев Тимофей
Алексеев Тимофей
tima.alekseev2109gmail.com@Mac lab05 %
```

Рис. 4.17: Запуск своей программы

Код прикладываю:

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите строку: ', 0h
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, buf1
mov edx, 80
call sread
mov eax, 4
mov ebx, 1
mov ecx, buf1
int 80h
call quit
```

5. Выводы

При выполнении данной лабораторной работы я приобрёл практические навыки работы в Midnight Commander, а также освоил инструкции языка ассемблера `mov` и `int`.

Список литературы

1. Пример выполнения лабораторной работы
2. Курс на ТУИС
3. Лабораторная работа №5
4. Программирование на языке ассемблера NASM Столяров А. В.