

# **Отчет по лабораторной работе №5**

**Дисциплина: архитектура компьютера**

**Бизев Никита Владимирович**

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Основы работы с тс . . . . .	8
4.2	Структура программы на языке ассемблера NASM . . . . .	10
4.3	Подключение внешнего файла . . . . .	14
4.4	Задания для самостоятельной работы . . . . .	20
<b>5</b>	<b>Выводы</b>	<b>26</b>
<b>6</b>	<b>Список литературы</b>	<b>27</b>

## Список иллюстраций

4.1	Открытый mc . . . . .	8
4.2	Перемещение между директориями . . . . .	9
4.3	Создание каталога . . . . .	9
4.4	Создание файла . . . . .	10
4.5	Открытие файла для редактирования . . . . .	11
4.6	Редактирование файла . . . . .	11
4.7	Открытие файла для просмотра . . . . .	12
4.8	Компиляция файла и передача на обработку компоновщику . . . . .	13
4.9	Исполнение файла . . . . .	14
4.10	Скачанный файл . . . . .	15
4.11	Копирование файла . . . . .	15
4.12	Копирование файла . . . . .	16
4.13	Редактирование файла . . . . .	17
4.14	Исполнение файла . . . . .	18
4.15	Отредактированный файл . . . . .	19
4.16	Исполнение файла . . . . .	19
4.17	Создание копии файла . . . . .	20
4.18	Внесение изменений в программу . . . . .	21
4.19	Запуск исполняющего файла . . . . .	22
4.20	Создание копии файла . . . . .	23
4.21	Внесение изменений в программу . . . . .	23
4.22	Запуск исполняющего файла . . . . .	25

# 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

### 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 (четырёх- рённое 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 mc

Открываю Midnight Commander, введя в терминал mc (рис. [4.1]).

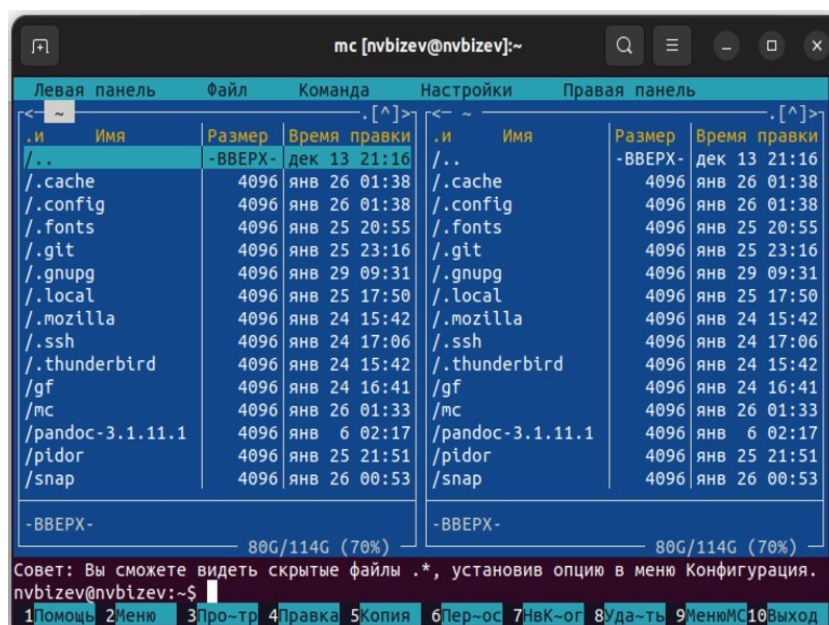


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/study/2023-2024/Архитектура Компьютера/arch-рс, используя файловый менеджер mc (рис. [4.2])



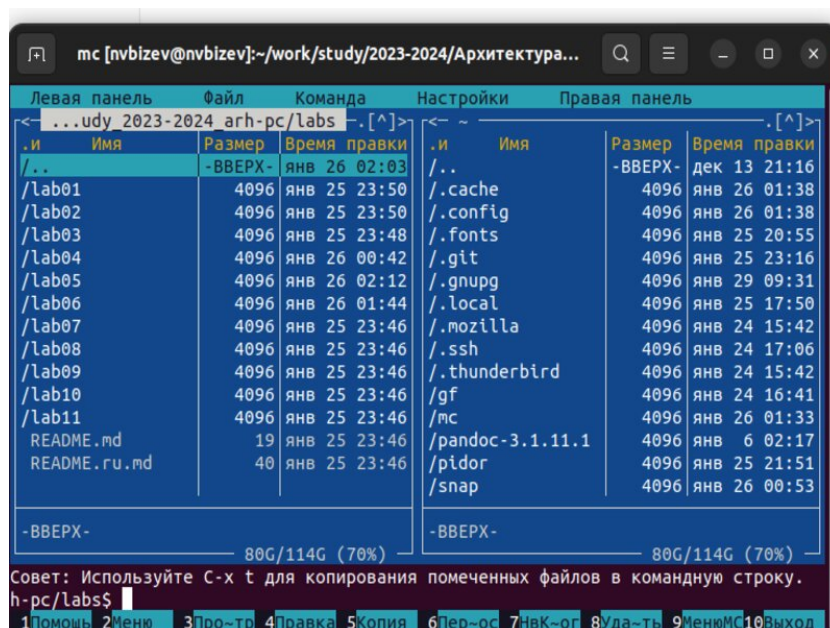


Рис. 4.2: Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab5 (рис. [4.3]).

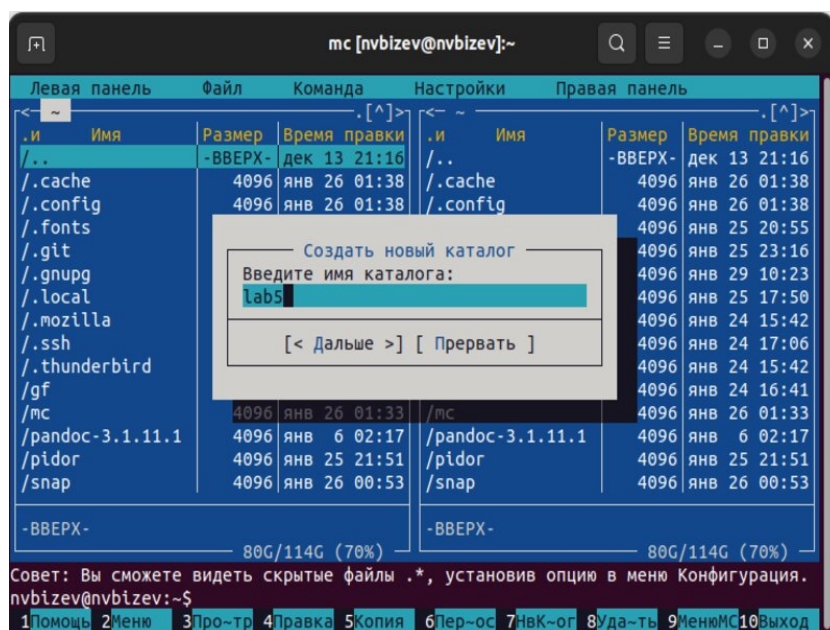


Рис. 4.3: Создание каталога

В строке ввода прописываю команду `touch lab5-1.asm`, чтобы создать файл, в

котором буду работать (рис. [4.4]).

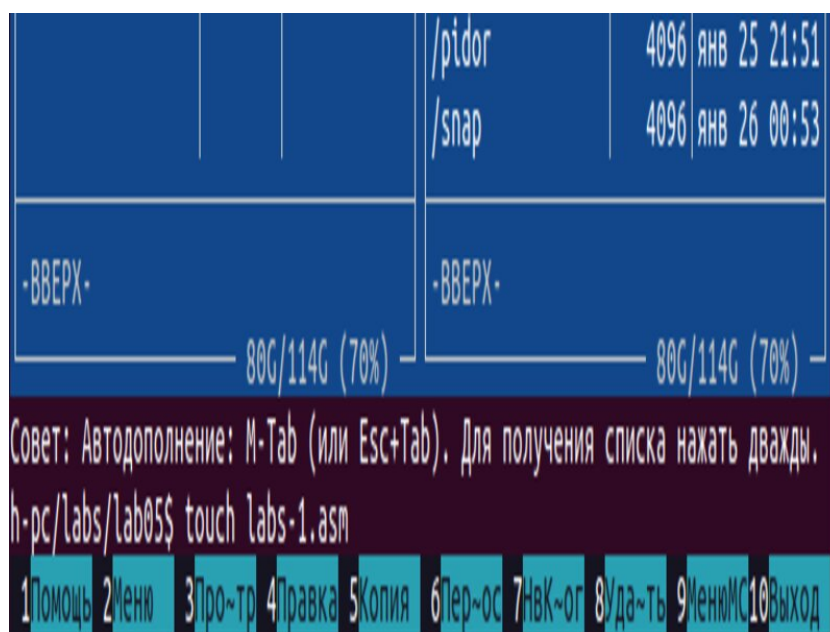


Рис. 4.4: Создание файла

## 4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в текстовом редакторе (рис. [4.5]).

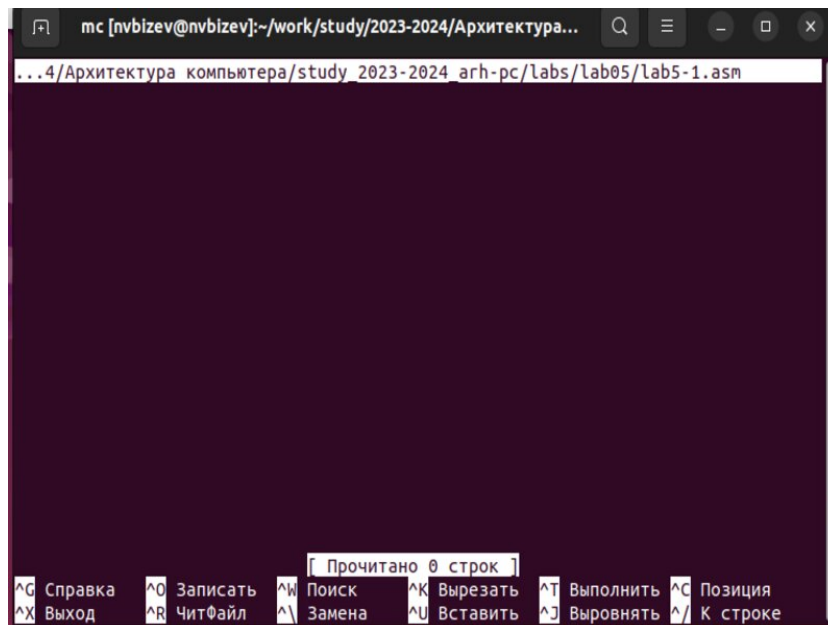


Рис. 4.5: Открытие файла для редактирования

Ввожу в файл код программы для запроса строки у пользователя (рис. [4.6]).

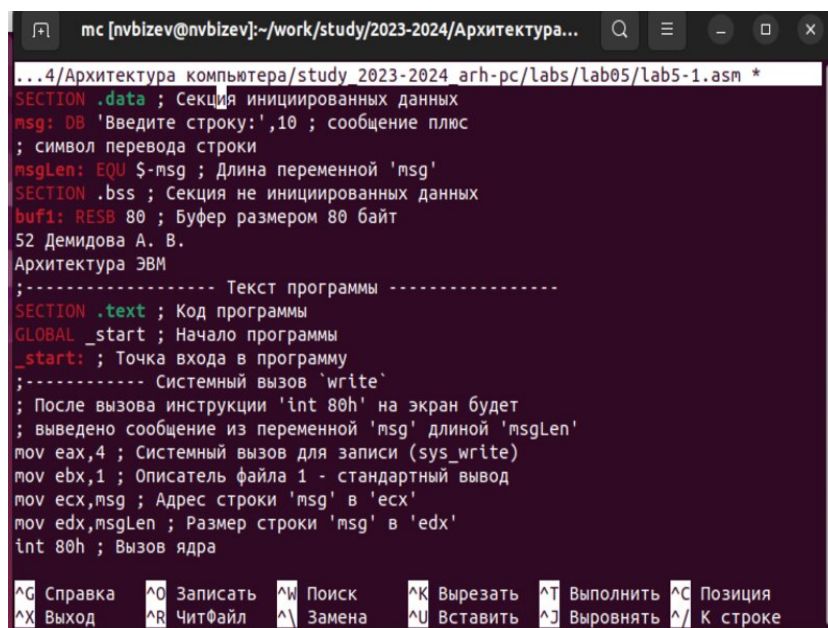
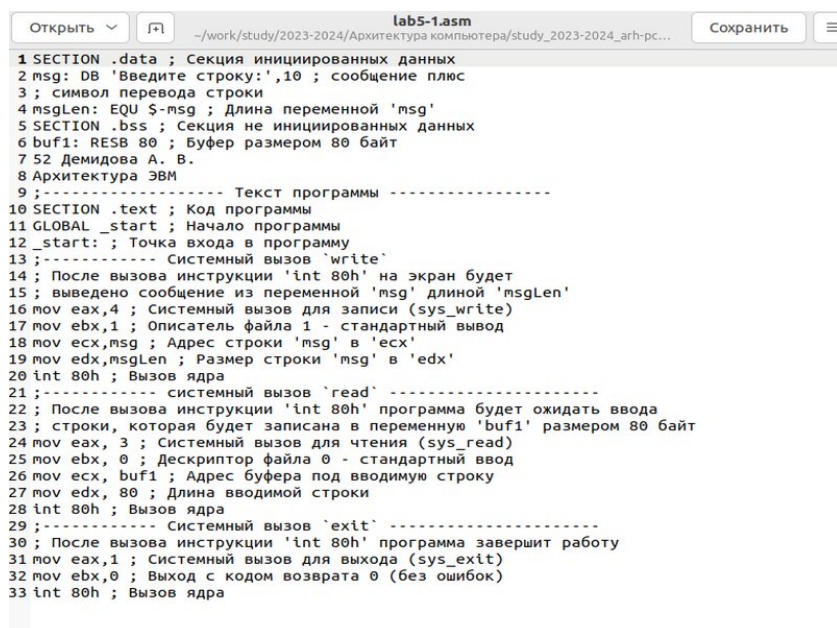


Рис. 4.6: Редактирование файла

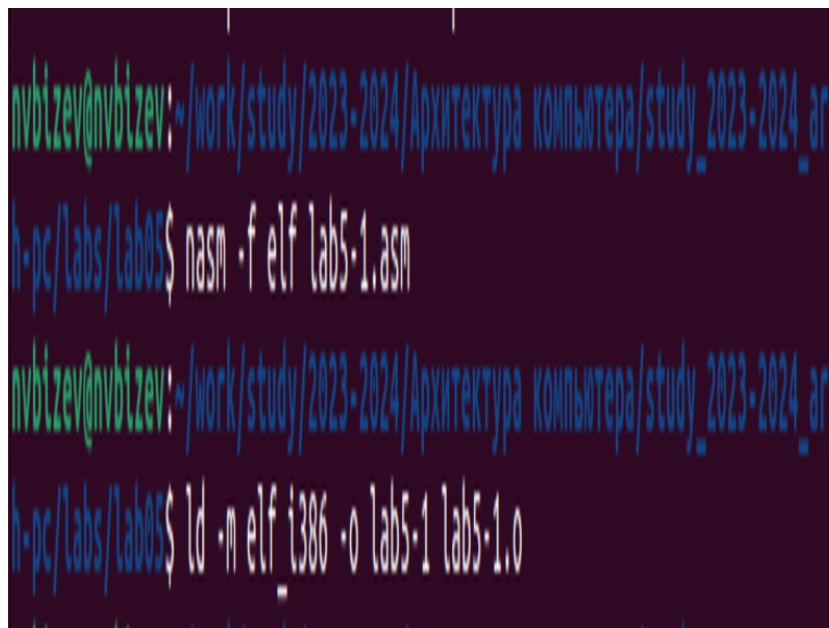
Открываю файл в текстовом редакторе для проверки. (рис. [4.7]).



```
1 SECTION .data ; Секция инициализированных данных
2 msg: DB 'Введите строку:',10 ; сообщение плюс
3 ; символ перевода строки
4 msgLen: EQU $-msg ; Длина переменной 'msg'
5 SECTION .bss ; Секция не инициализированных данных
6 buf1: RESB 80 ; Буфер размером 80 байт
7 52 Демидова А. В.
8 Архитектура ЭВМ
9 ;----- Текст программы -----
10 SECTION .text ; Код программы
11 GLOBAL _start ; Начало программы
12 _start: ; Точка входа в программу
13 ;----- Системный вызов 'write' -----
14 ; После вызова инструкции 'int 80h' на экран будет
15 ; выведено сообщение из переменной 'msg' длиной 'msgLen'
16 mov eax,4 ; Системный вызов для записи (sys_write)
17 mov ebx,1 ; Описатель файла 1 - стандартный вывод
18 mov ecx,msg ; Адрес строки 'msg' в 'ecx'
19 mov edx,msgLen ; Размер строки 'msg' в 'edx'
20 int 80h ; Вызов ядра
21 ;----- системный вызов 'read' -----
22 ; После вызова инструкции 'int 80h' программа будет ожидать ввода
23 ; строки, которая будет записана в переменную 'buf1' размером 80 байт
24 mov eax,3 ; Системный вызов для чтения (sys_read)
25 mov ebx,0 ; Дескриптор файла 0 - стандартный ввод
26 mov ecx,buf1 ; Адрес буфера под вводимую строку
27 mov edx,80 ; Длина вводимой строки
28 int 80h ; Вызов ядра
29 ;----- Системный вызов 'exit' -----
30 ; После вызова инструкции 'int 80h' программа завершит работу
31 mov eax,1 ; Системный вызов для выхода (sys_exit)
32 mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
33 int 80h ; Вызов ядра
```

Рис. 4.7: Открытие файла для просмотра

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm`. Создался объектный файл `lab5-1.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-1 lab5-1.o` (рис. [4.8]). Создался исполняемый файл `lab6-1`.

A terminal window with a dark background and light-colored text. The prompt is 'nrvbizev@nrvbizev: ~/work/study/2023-2024/Архитектура компьютера/study\_2023-2024\_ar'. The first command is 'h-pc/labs/lab05\$ nasm -f elf lab5-1.asm'. The second command is 'h-pc/labs/lab05\$ ld -m elf\_i386 -o lab5-1 lab5-1.o'.

```
nrvbizev@nrvbizev: ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar
h-pc/labs/lab05$ nasm -f elf lab5-1.asm
nrvbizev@nrvbizev: ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar
h-pc/labs/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
```

Рис. 4.8: Компиляция файла и передача на обработку компоновщику

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. [4.9]).

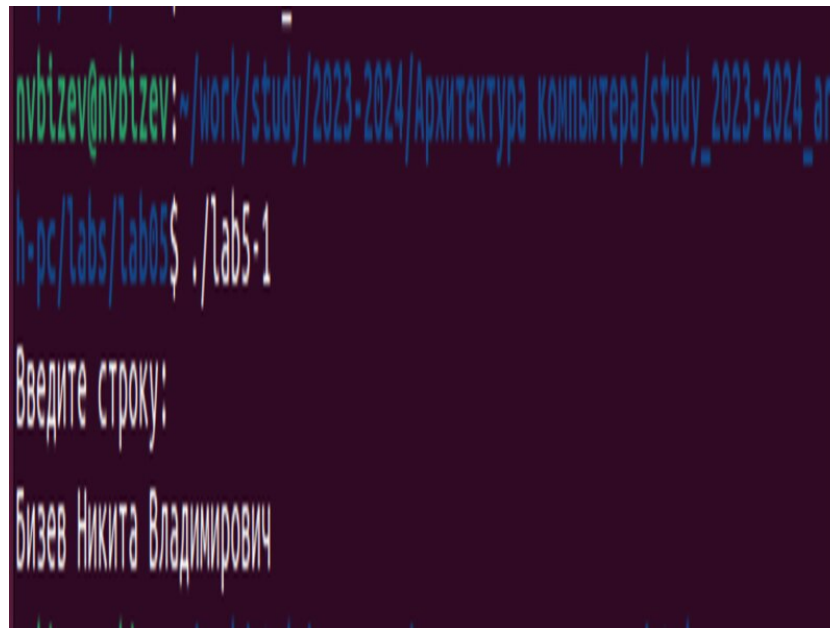


Рис. 4.9: Исполнение файла

### 4.3 Подключение внешнего файла

Скачиваю файл in\_out.asm со страницы курса в ТУИС. (рис. [4.10]).



Левая панель	Файл	Команда	Нас
<-	...23-2024_arh-pc/labs/lab05	-.[^]>	<-
.и	Имя	Размер	Время правки
	/..	-ВВЕРХ-	янв 29 09:58
	/presentation	4096	янв 25 23:46
	/report	4096	янв 25 23:46
	in_out.asm	3942	янв 29 09:58
	*lab5-1	8744	янв 29 09:57
	lab5-1.asm	2096	янв 29 09:57
	lab5-1.o	752	янв 29 09:57

Рис. 4.10: Скачанный файл

С помощью функциональной клавиши F5 копирую файл in\_out.asm в созданный каталог lab05 (рис. [4.11]).

...23-2024\_arh-pc/labs/lab05

-.[^]>

<- ~

-.[^]

Имя	Размер	Время правки	.и	Имя	Размер	Время пра
-----	--------	--------------	----	-----	--------	-----------

Копирование

Копировать файл "in\_out.asm" с исходным шаблоном:

\*

[^]

[x] Метасимволы shell

В:

/home/nvbizev/

[^]

[ ] Разыменовывать ссылки

[ ] Внутрь подкаталога, если есть

[x] Сохранять атрибуты

[ ] Изменять относительные ссылки

[< Дальше >]

[ В фоне ]

[ Прервать ]

Рис. 4.11: Копирование файла

С помощью функциональной клавиши F5 копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне tc прописываю имя для копии файла (рис. [4.12]).

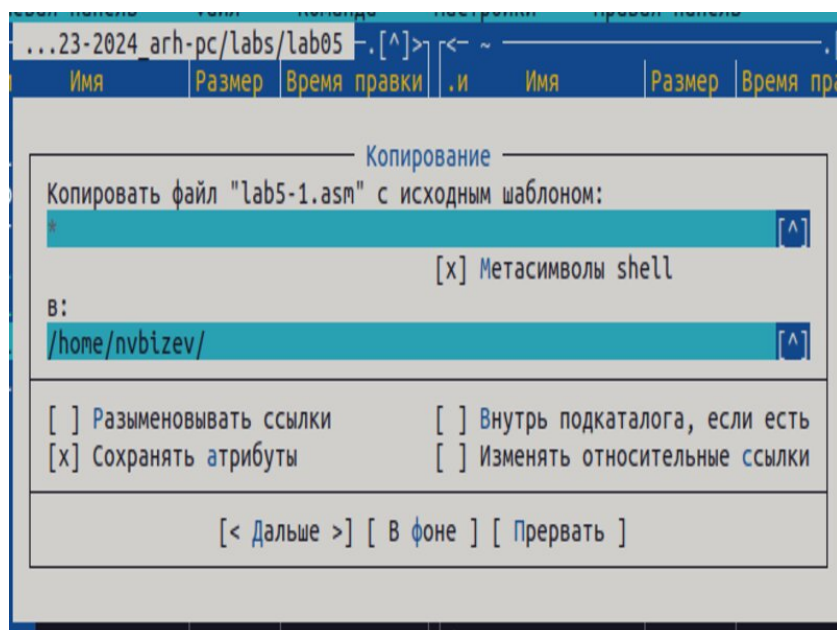
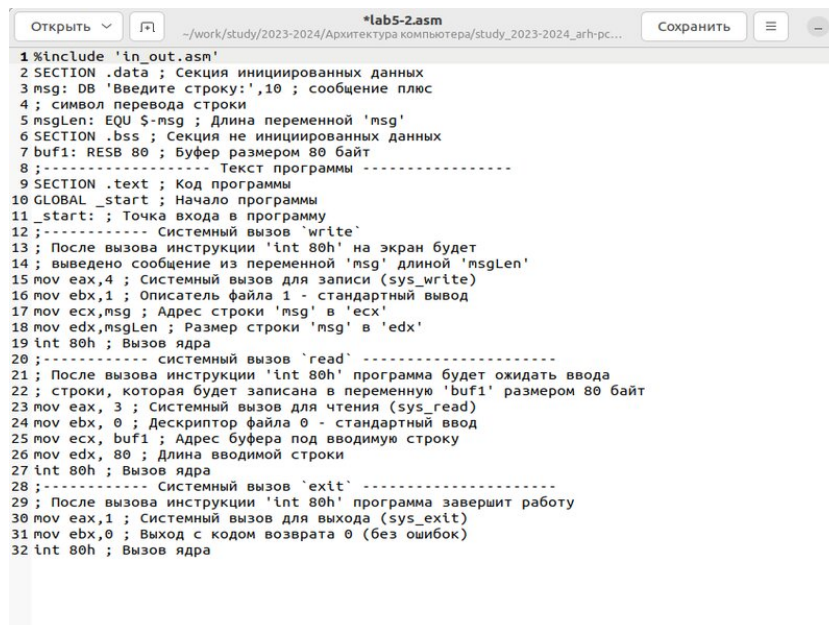


Рис. 4.12: Копирование файла

Изменяю содержимое файла lab5-2.asm во встроенном редакторе. (рис. [4.13]), чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm.

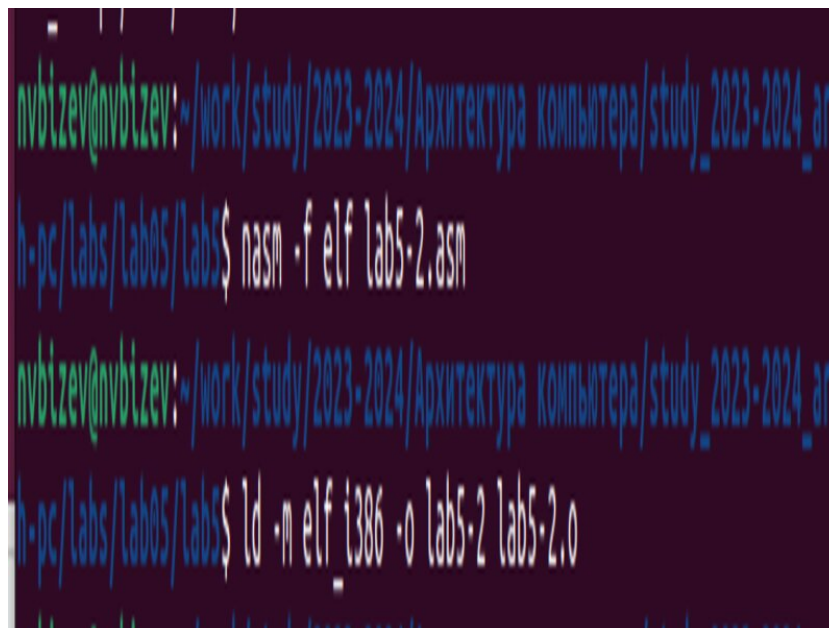




```
1 %include 'in_out.asm'
2 SECTION .data ; Секция иницированных данных
3 msg: DB 'Введите строку:',10 ; сообщение плюс
4 ; символ перевода строки
5 msgLen: EQU $-msg ; Длина переменной 'msg'
6 SECTION .bss ; Секция не иницированных данных
7 buf1: RESB 80 ; Буфер размером 80 байт
8 ;----- Текст программы -----
9 SECTION .text ; Код программы
10 GLOBAL _start ; Начало программы
11 _start: ; Точка входа в программу
12 ;----- Системный вызов 'write' -----
13 ; После вызова инструкции 'int 80h' на экран будет
14 ; выведено сообщение из переменной 'msg' длиной 'msgLen'
15 mov eax,4 ; Системный вызов для записи (sys_write)
16 mov ebx,1 ; Описатель файла 1 - стандартный вывод
17 mov ecx,msg ; Адрес строки 'msg' в 'ecx'
18 mov edx,msgLen ; Размер строки 'msg' в 'edx'
19 int 80h ; Вызов ядра
20 ;----- системный вызов 'read' -----
21 ; После вызова инструкции 'int 80h' программа будет ожидать ввода
22 ; строки, которая будет записана в переменную 'buf1' размером 80 байт
23 mov eax,3 ; Системный вызов для чтения (sys_read)
24 mov ebx,0 ; Дескриптор файла 0 - стандартный ввод
25 mov ecx,buf1 ; Адрес буфера под вводимую строку
26 mov edx,80 ; Длина вводимой строки
27 int 80h ; Вызов ядра
28 ;----- Системный вызов 'exit' -----
29 ; После вызова инструкции 'int 80h' программа завершит работу
30 mov eax,1 ; Системный вызов для выхода (sys_exit)
31 mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
32 int 80h ; Вызов ядра
```

Рис. 4.13: Редактирование файла

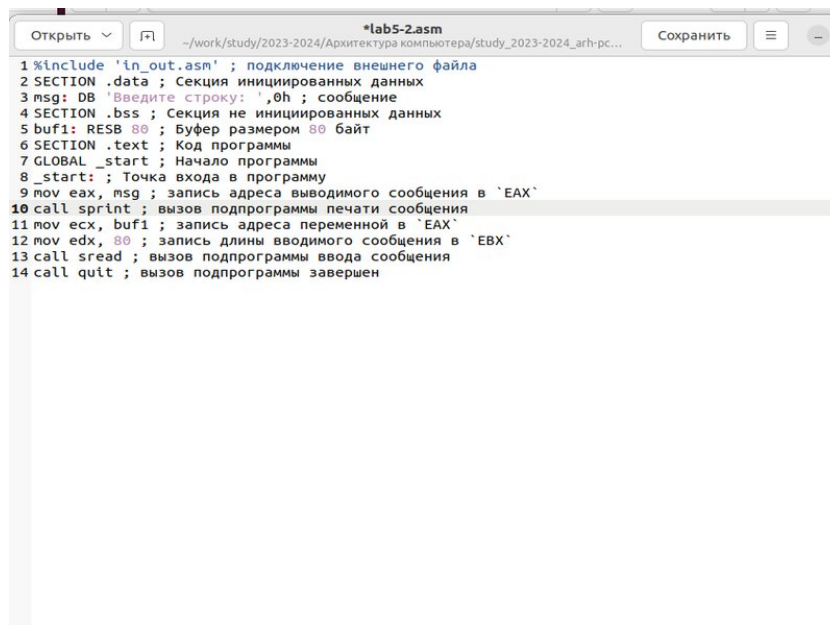
Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-2.asm`. Создался объектный файл `lab5-2.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o` Создался исполняемый файл `lab5-2`. Запускаю исполняемый файл. (рис. [4.14])

A terminal window with a dark background and light-colored text. The prompt is 'nvbitzev@nvbitzev: ~/work/study/2023-2024/Архитектура компьютера/study\_2023-2024\_ar'. The first command is 'h-pc/labs/labs5/labs\$ nasm -f elf lab5-2.asm'. The second command is 'h-pc/labs/labs5/labs\$ ld -m elf\_i386 -o lab5-2 lab5-2.o'.

```
nvbitzev@nvbitzev: ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar
h-pc/labs/labs5/labs$ nasm -f elf lab5-2.asm
nvbitzev@nvbitzev: ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar
h-pc/labs/labs5/labs$ ld -m elf_i386 -o lab5-2 lab5-2.o
```

Рис. 4.14: Исполнение файла

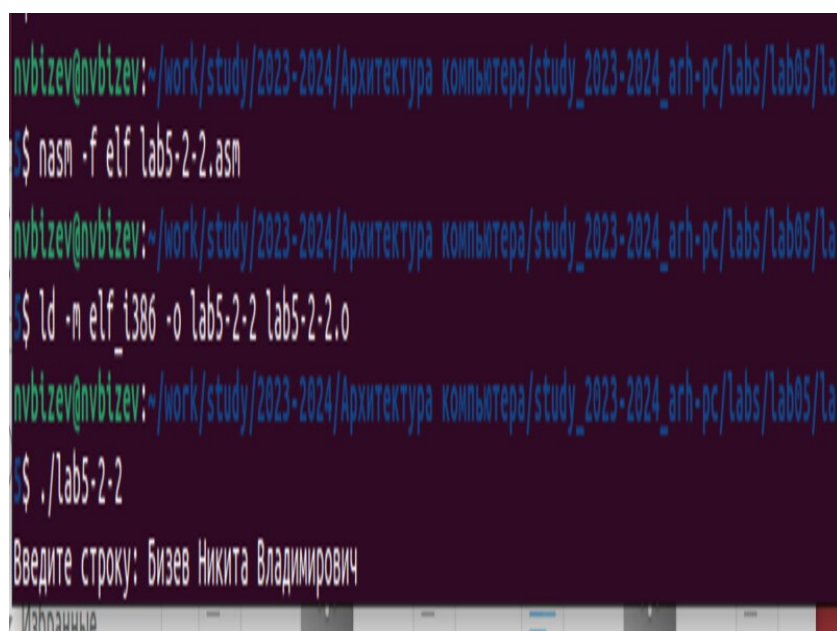
Открываю файл lab5-2.asm для редактирования в текстовом редакторе функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. [4.15]).



```
*lab5-2.asm
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data ; Секция иницированных данных
3 msg: DB "Введите строку: ",0h ; сообщение
4 SECTION .bss ; Секция не иницированных данных
5 buf1: RESB 80 ; Буфер размером 80 байт
6 SECTION .text ; Код программы
7 GLOBAL _start ; Начало программы
8 _start: ; Точка входа в программу
9 mov eax, msg ; запись адреса выводимого сообщения в `EAX`
10 call sprint ; вызов подпрограммы печати сообщения
11 mov ecx, buf1 ; запись адреса переменной в `EAX`
12 mov edx, 80 ; запись длины вводимого сообщения в `EBX`
13 call sread ; вызов подпрограммы ввода сообщения
14 call quit ; вызов подпрограммы завершения
```

Рис. 4.15: Отредактированный файл

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. [4.16]).



```
nvbizev@nvbizev: ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab05/lab
: $ nasm -f elf lab5-2.asm
nvbizev@nvbizev: ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab05/lab
: $ ld -m elf_i386 -o lab5-2-2 lab5-2-2.o
nvbizev@nvbizev: ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab05/lab
: $ ./lab5-2-2
Введите строку: Бизев Никита Владимирович
```

Рис. 4.16: Исполнение файла

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая выполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами `sprintLF` и `sprint`.

## 4.4 Задания для самостоятельной работы

1. Создаю копию файла lab5-1.asm. (рис. 4.17).

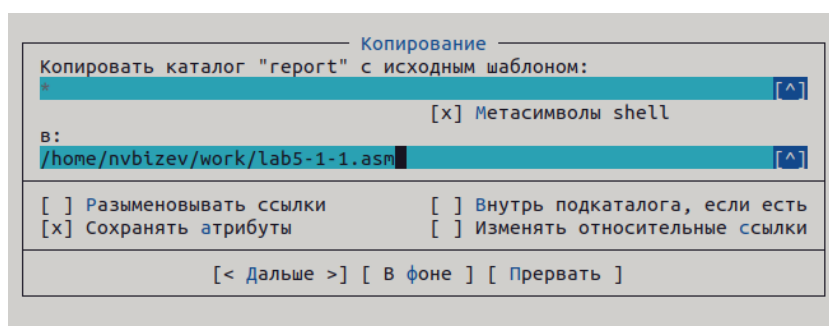
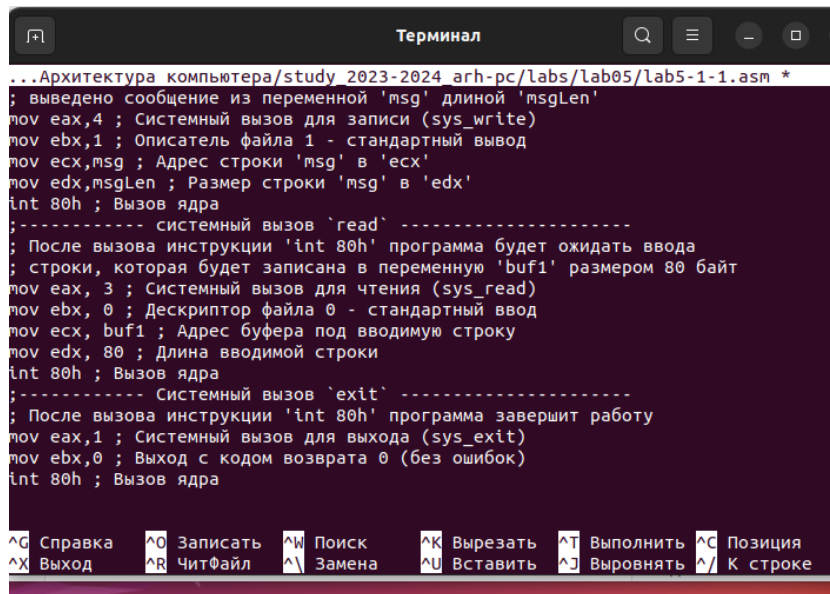


Рис. 4.17: Создание копии файла

Вношу изменения в программу так, чтобы она выводила введённую строку на экран. (рис. 4.18).



```
...Архитектура компьютера/study_2023-2024_arh-pc/labs/lab05/lab5-1-1.asm *
; выведено сообщение из переменной 'msg' длиной 'msgLen'
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки 'msg' в 'ecx'
mov edx,msgLen ; Размер строки 'msg' в 'edx'
int 80h ; Вызов ядра
;----- системный вызов 'read' -----
; После вызова инструкции 'int 80h' программа будет ожидать ввода
; строки, которая будет записана в переменную 'buf1' размером 80 байт
mov eax, 3 ; Системный вызов для чтения (sys_read)
mov ebx, 0 ; Дескриптор файла 0 - стандартный ввод
mov ecx, buf1 ; Адрес буфера под вводимую строку
mov edx, 80 ; Длина вводимой строки
int 80h ; Вызов ядра
;----- Системный вызов 'exit' -----
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра

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

Рис. 4.18: Внесение изменений в программу

Сам код:

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

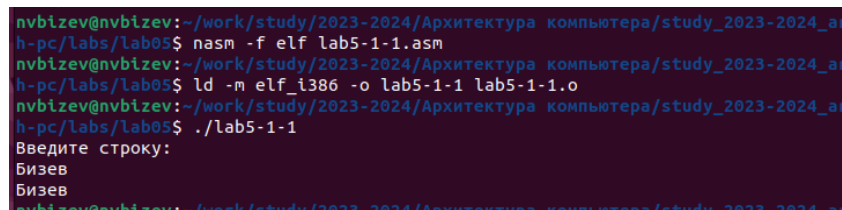
mov ebx, 0

```

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

```

2. Получаю исполняемый файл и проверяю его работу. На приглашение ввести строку ввожу свою фамилию. (рис. 4.19).



```

nvbizhev@nvbizhev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_a
h-pc/labs/lab05$ nasm -f elf lab5-1-1.asm
nvbizhev@nvbizhev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_a
h-pc/labs/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
nvbizhev@nvbizhev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_a
h-pc/labs/lab05$ ./lab5-1-1
Введите строку:
Бизев
Бизев
nvbizhev@nvbizhev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_a

```

Рис. 4.19: Запуск исполняющего файла

Программа работает.

3. Создаю копию файла lab5-2.asm. (рис. 4.20).

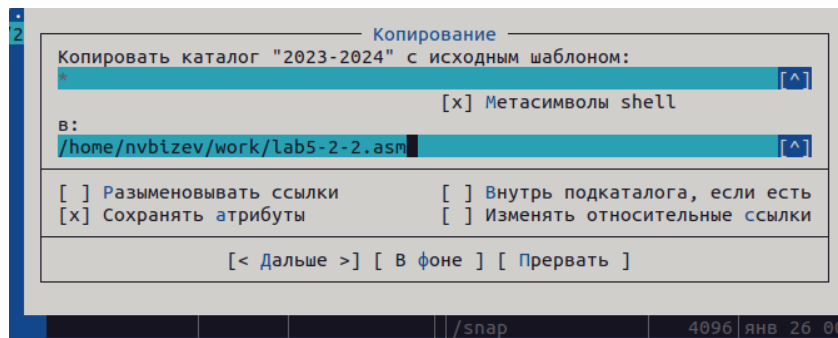


Рис. 4.20: Создание копии файла

Вношу изменения в программу с использование подпрограмм из внешнего файла `in_out.asm` так, чтобы она выводила введённую строку на экран. (рис. 4.21).

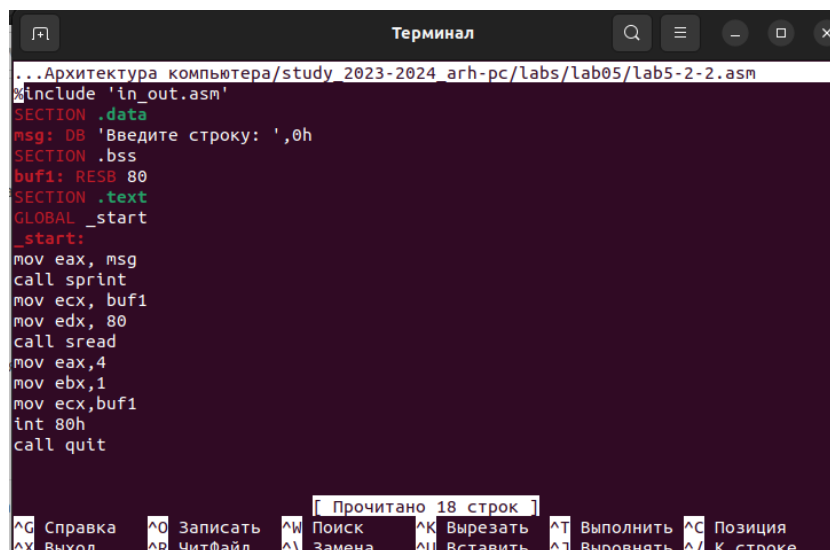


Рис. 4.21: Внесение изменений в программу

Сам код:

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите строку:',0h
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

```

4. Создаю исполняемый файл и проверяю его работу. (рис. 4.22).



```
nvbizhev@nvbizhev:~/work/study/2023-2024/Архитектура компьютера/s  
h-pc/labs/lab05$ nasm -f elf lab5-2-2.asm  
nvbizhev@nvbizhev:~/work/study/2023-2024/Архитектура компьютера/s  
h-pc/labs/lab05$ ld -m elf_i386 -o lab5-2-2 lab5-2-2.o  
nvbizhev@nvbizhev:~/work/study/2023-2024/Архитектура компьютера/s  
h-pc/labs/lab05$  
nvbizhev@nvbizhev:~/work/study/2023-2024/Архитектура компьютера/s  
h-pc/labs/lab05$ ./lab5-2-2  
Введите строку: Бизев  
Бизев
```

Рис. 4.22: Запуск исполняющего файла

Программа работает.

## 5 Выводы

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

## 6 Список литературы

- [illegible]