**Отчёт по лабораторной работе №5**

***Дисциплина: Архитектура компьютера***

Байрамова Гюльсабах Акифовна НММбд-01-24

.

3.3.2 Описание инструкции int .

.

.

.

.

.

.

.

12

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

2

**26**

**Список литературы**

**25**

**5 Выводы**

22

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.1 Задания для самостоятельной работы.

**16**

**4 Выполнение лабораторной работы**

3.3.3 Системные вызовы для обеспечения диалога с пользователем 14

14

.

.

.

.

.

.

.

.

.

.

3.2 Cтруктура программы на языке ассемблера NASM .

7

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

3.1 Основы работы с Midnight Commander.

**7**

**3 Теоретическое введение**

**6**

**2 Задание**

**5**

**1 Цель работы**

**Содержание**

.

.

.

.

3.3.1 Описание инструкции mov.

12

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

3.3 Элементы программирования.

9

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.13 Текст программы ..

21

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.14 Создание и работа испоняемого файла

21

.

4.12 Создание копии файла lab5-1.asm .

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

20

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.10 Работа исполняемого файла.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

20

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.11 Перемещение файла in\_out.asm .

20

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.19 Измененнный текст программы ..

23

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.18 Создание и работа испоняемого файла

23

.

.

.

3

24

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.20 Создание и работа испоняемого файла

24

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.16 Создание копии файла lab5-1.asm .

22

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.15 Создание и работа испоняемого файла

22

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.17 Измененнный текст программы ..

23

.

.

.

.

.

.

.

.

.

.

.

.

.

4.4 Создание файла lab5-1.asm ..

17

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.3 Создание папки lab05 .

17

.

.

.

.

.

.

.

.

.

.

4.5 Выбор редактора .

18

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.1 Окно Midnight Commander ..

7

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

3.1 Окно Midnight Commander ..

**Список иллюстраций**

.

.

.

.

.

.

.

.

.

.

.

4.2 Каталог ~/work/arch-pc/ в Midnight Commander

16

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.8 Проверка .

19

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.9 Создание исполняемого файла .

20

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.6 Редактор mcedit.

18

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

4.7 Текст программы . .

19

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

10

.

.

.

.

.

.

.

3.3 Директивы для объявления неинициированных данных .

.

.

.

.

.

.

.

.

4

13

.

.

.

.

.

.

.

3.4 Варианты использования mov с разными операндами .

11

.

.

.

.

.

.

.

3.2 Примеры .

8

.

.

.

.

.

.

.

.

.

.

.

3.1 Функциональные клавиши Midnight Commander

**Список таблиц**

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

**1 Цель работы**

Приобретение практических навыков работы в *Midnight Commander*. Освоение

инструкций языка ассемблера

и

.

mov

int

5

**2 Задание**

1) Выполнение лабораторной работы

1) Подключение внешнего файла in\_out.asm

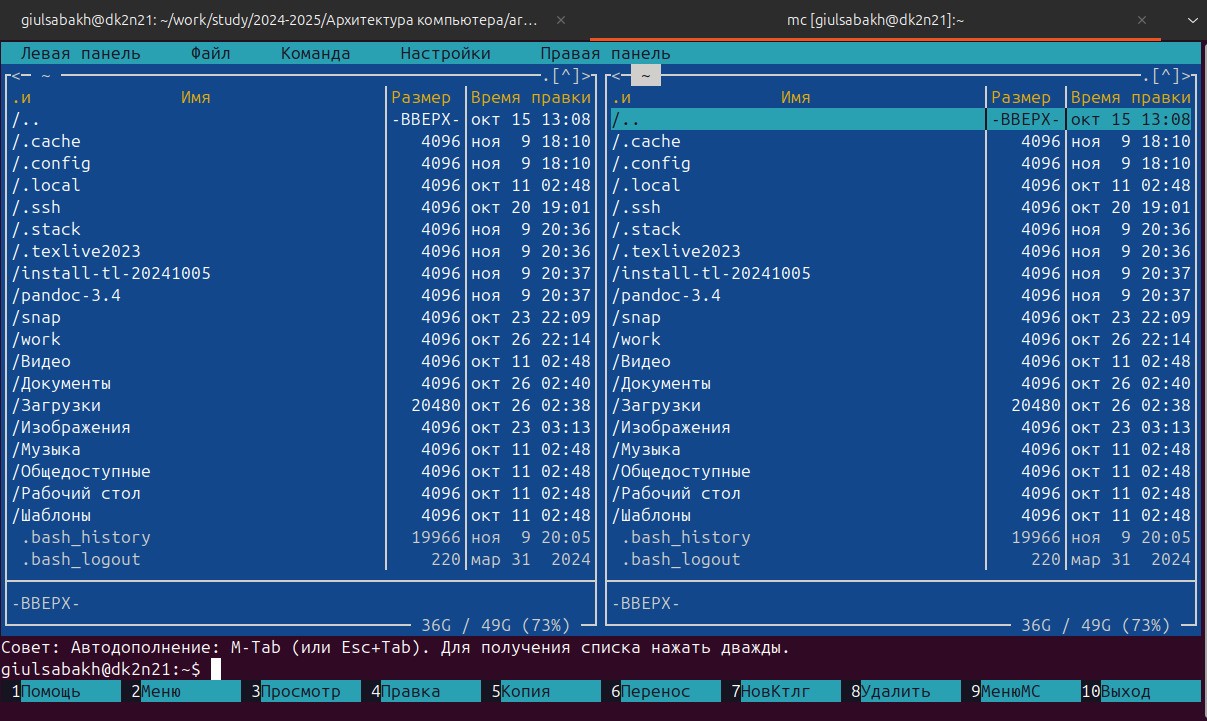
2) Задания для самостоятельной работы

6

Midnight Commander

7

Рис. 3.1: Окно Midnight Commander



рым привязаны часто выполняемые операции (табл. 3.1).

F10

F1

Midnight Commander

, к кото-

—

используются функциональные клавиши

В

Enter

(рис. 3.1).

строке mc и нажать клавишу

достаточно ввести в командной

Для активации оболочки

позволяет сделать работу с файлами более удобной и наглядной.

Midnight Commander

mc

является файловым менеджером.

файловой системой, т.е.

сматривать структуру каталогов и выполнять основные операции по управлению

mc

) — это программа, которая позволяет про-

Midnight Commander (или просто

**3.1 Основы работы с Midnight Commander**

**3 Теоретическое введение**

↑

(или через меню

•

ных расширений файлов с инструментами их запуска или обработки);

mc.ext

заданы правила связи определён-

файла (если в файле расширений

Enter

↓

>

для входа в каталог или открытия

используется для навигации,

и

•

Tab

используется для переключениями между панелями;

•

>

8

Midnight Commander

, за которыми доступен для работы ко-

вращает панели

Отключить панели

Команда

Ctrl + o

) скрывает или воз-

Следующие комбинации клавиш облегчают работу с Midnight Commander:

(или через меню

•

содержимое правой и левой панелей;

Переставить панели

Команда

Ctrl + u

) меняет местами

F1

копирование файла или группы отмеченных файлов из каталога,

подсветка в активной панели

F4

вызов встроенного редактора для файла, на который указывает

F3

просмотр файла, на который указывает подсветка в активной панели

F2

вызов меню, созданного пользователем

F5

вызов контекстно-зависимой подсказки

Назначение

виша

Кла-

Таблица 3.1: Функциональные клавиши Midnight Commander

отображаемого в активной панели, в каталог, отображаемый на второй

панели

перенос файла или группы отмеченных файлов из каталога,

F6

отображаемого в активной панели, в каталог, отображаемый на второй

панели

создание подкаталога в каталоге, отображаемом в активной панели

F7

удаление файла (подкаталога) или группы отмеченных файлов

F8

вызов основного меню программы

F9

выход из программы

F10

тивы

**SECTION** .bss *; Секция содержит переменные, для*

... *; которых не задано начальное значение*

**SECTION** .text *; Секция содержит код программы*

**GLOBAL** \_start

\_start: *; Точка входа в программу*

... *; Текст программы*

**mov eax**,1 *; Системный вызов для выхода (sys\_exit)*

**mov ebx**,0 *; Выход с кодом возврата 0 (без ошибок)*

**int** 80h *; Вызов ядра*

Для объявления инициированных данных в секции

используются дирек-

.data

... *; которых задано начальное значение*

,

,

,

и

, которые резервируют память и указывают, какие значения

DB

DW

DD

DQ

DT

должны храниться в этой памяти:

9

NASM

мандный интерпретатор оболочки и выводимая туда информация.

•

(или через меню

>

) позволяет срав-

Ctrl + x + d

Команда

Сравнить каталоги

нить содержимое каталогов, отображаемых на левой и правой панелях.

**3.2 Cтруктура программы на языке ассемблера NASM**

Программа на языке ассемблера

, как правило, состоит из трёх секций:

секция кода программы (

), секция инициированных (известных

SECTION .text

во время компиляции) данных (

) и секция неинициализирован-

SECTION .data

ных данных (тех, под которые во время компиляции только отводится память, а

значение присваивается в ходе выполнения программы) (

).

SECTION .bss

Таким образом, общая структура программы имеет следующий вид:

**SECTION** .data *; Секция содержит переменные, для*

10011001b

b db ‘!’

конце числа) подсказки

binary

b

) в

(

двоичную систему счисления указывает также буква

значением, заданным в двоичной системе счисления (на

определяем переменную

a

размером 1 байт с начальным

определяем переменную

a db

Пояснение

Пример

размером 4 байта с начальным

10

decimal

d

) в конце числа)

(

десятичную систему указывает буква

значением, заданным в десятичной системе счисления (на

d

определяем переменную

d dd -345d

определяем строку из 5 байт

c db “Hello”

!

b

в 1 байт, инициализируемую символом

) — определяет переменную размеров в 2 байта (слово);

ное слово);

define double word

DD

) — определяет переменную размером в 4 байта (двой-

(

•

define word

DW

•

(

•

define byte

DB

) — определяет переменную размером в 1 байт;

(

•

) — определяет переменную размером в 10 байт.

Таблица 3.2: Примеры

<имя> DB <операнд> [, <операнд>] [, <операнд>]

Синтаксис директив определения данных следующий:

связи с особенностями хранения данных в оперативной памяти.

ления массивов. Для определения строк принято использовать директиву DB в

Директивы используются для объявления простых переменных и для объяв-

define ten bytes

DT

(

•

рённое слово);

define quad word

DQ

) — определяет переменную размером в 8 байт (учетве-

(

Резервирование

По адресу с

resb

string resb 20

заданного числа

меткой

string

однобайтовых

будет расположен

ячеек

массив из 20

однобайтовых

ячеек (хранение

строки символов)

Резервирование

По адресу с

resw

count resw 256

заданного числа

меткой

count

двухбайтовых

будет расположен

ячеек (слов)

массив из 256

двухбайтовых

слов

11

Пример

Пояснение

h dd 0f1ah

определяем переменную

размером 4 байта с начальным

h

значением, заданным в шестнадцатеричной системе счисления (

h

-

)

hexadecimal

Для объявления неинициированных данных в секции .bss используются ди-

рективы resb, resw, resd и другие, которые сообщают ассемблеру, что необходимо

зарезервировать заданное количество ячеек памяти. Примеры их использования

приведены в табл. 3.3.

Таблица 3.3: Директивы для объявления неинициированных данных

Назначение

Назначение

Директика

директивы

Аргумент

аргумента

Инструкция языка ассемблера mov предназначена для дублирования данных

ячеек (двойных

двойное слово (т.е.

слов)

4 байта для

хранения

большого числа)

**3.3 Элементы программирования**

**3.3.1 Описание инструкции mov**

расположено одно

источника в приёмнике. В общем виде эта инструкция записывается в виде

**mov** dst,src

Здесь операнд dst — приёмник, а src — источник.

В качестве операнда могут выступать регистры (register),

ячейки памяти

(memory) и непосредственные значения (const). В табл. 3.4 приведены варианты

использования mov с разными операндами.

12

Резервирование

Назначение

Назначение

Директика

директивы

Аргумент

аргумента

По адресу с

resd

x resd 1

заданного числа

меткой

будет

x

четырёхбайтовых

переменную

значение из

регистра

ebx

пишет в регистр

mov

mov eax,403045h

значение

<reg>,<const>

eax

403045h

записывает в

mov

mov byte[rez],0

rez

<mem>,<const>

rez

значение 0

ВАЖНО! Переслать значение из одной ячейки памяти в другую нельзя, для

этого необходимо использовать две инструкции

:

mov

**mov eax**, x

**mov** y, **eax**

Также необходимо учитывать то, что размер операндов приемника и источника

должны совпадать. Использование слудующих примеров приведет к ошибке:

13

регистр

Таблица 3.4: Варианты использования mov с разными операндами

Тип операндов

Примеры

Пояснение

пересылает

mov <reg>,<reg>

mov eax,ebx

значение

регистра

в

ebx

eax

пересылает в

mov <reg>,<mem> mov cx,[eax]

регистр

cx

значение из

памяти,

указанной в

eax

пересылает в

mov <mem>,<reg> mov rez,ebx

переменную

должна вернуть значение, то она помещает его в регистр

. Поэтому перед

eax

вызовом прерывания необходимо поместить в этот регистр нужный номер. Кроме

того, многим системным функциям требуется передавать какие-либо параметры.

По принятым в ОС

правилам эти параметры помещаются в порядке следо-

Linux

вания в остальные регистры процессора:

,

,

. Если системная функция

ebx

ecx

edx

нить, ядро извлекает номер системного вызова из регистра

.

eax

**3.3.3 Системные вызовы для обеспечения диалога с**

**пользователем**

Простейший диалог с пользователем требует наличия двух функций — выво-

да текста на экран и ввода текста с клавиатуры. Простейший способ вывести

строку на экран — использовать системный вызов

. Этот системный вызов

write

имеет номер 4, поэтому перед вызовом инструкции

необходимо поместить

int

14

n

•

— ошибка, попытка записать 2-байтное число в 1-байтный

mov al,1000h

регистр;

•

— ошибка, размеры операндов не совпадают.

mov eax,cx

**3.3.2 Описание инструкции int**

Инструкция языка ассемблера intпредназначена для вызова прерывания с

указанным номером. В общем виде она записывается в виде

**int** n

Здесь

— номер прерывания, принадлежащий диапазону 0–255.

При программировании в

с использованием вызовов ядра

Linux

sys\_calls

(принято задавать в шестнадцатеричной системе счисления).

n=80h

После вызова инструкции

выполняется системный вызов какой-либо

int 80h

функции ядра

. При этом происходит передача управления ядру операци-

Linux

онной системы. Чтобы узнать, какую именно системную функцию нужно выпол-

является обязательным в конце любой программы на

Для ввода строки с клавиатуры можно использовать аналогичный системный

вызов

. Его аргументы – такие же, как у вызова

, только для «чтения» с

read

write

клавиатуры используется файловый дескриптор 0 (стандартный ввод).

Системный вызов

длина выводимой строки.

exit

языке ассемблер. Для обозначения конца программы перед вызовом инструкции

необходимо поместить в регистр еах значение 1, а в регистр

код

int 80h

ebx

завершения 0.

15

Вторым аргументом задаётся адрес выводимой строки (помещаем его в регистр

значение 4 в регистр

. Первым аргументом

, помещаемым в регистр

eax

write

, задаётся дескриптор файла. Для вывода на экран в качестведескриптора

ebx

файла нужно указать 1 (это означает «стандартный вывод», т. е. вывод на экран).

, например, инструкцией

). Строка может иметь любую длину.

ecx

mov ecx, msg

Последним аргументом (т.е. в регистре

) должна задаваться максимальная

edx

**4 Выполнение лабораторной работы**

Откроем

и с помощью клавиш

и

перейдем в каталог

Midnught Commander

↑

↓

‘~/work/arch-pc/, созданный при выполненнии лабораторной работы №4(рис. 4.1,

4.2)

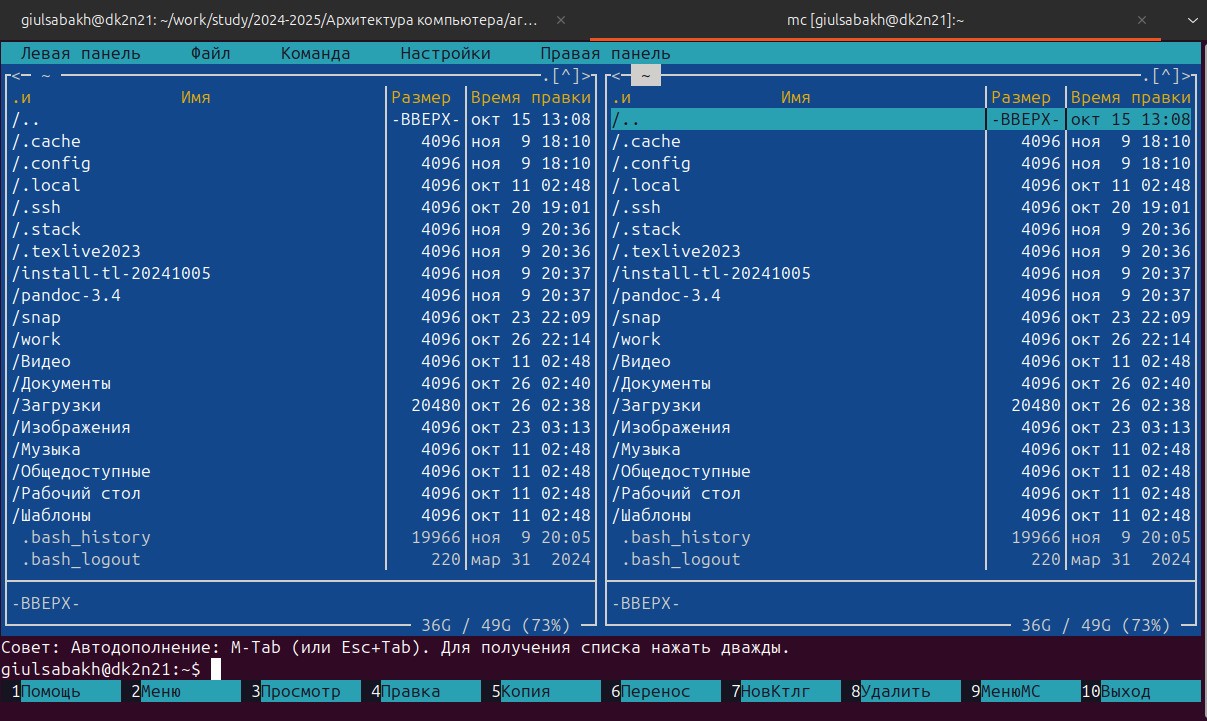


Рис. 4.1: Окно Midnight Commander

16

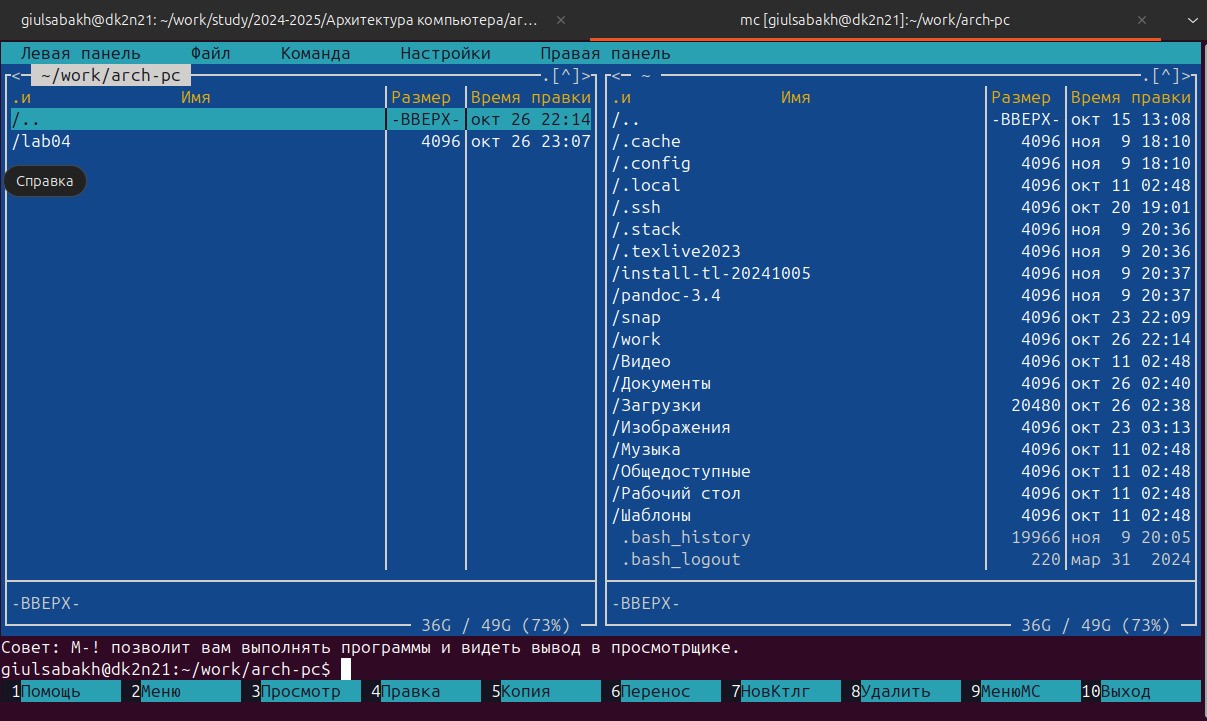


Рис. 4.2: Каталог ~/work/arch-pc/ в Midnight Commander

С помощью функциональной клавиши

создадим папку

и, переёдя в

F7

lab05

неё, с помощью команды

создадим файл

(рис. 4.3, 4.4)

touch

lab5-1.asm

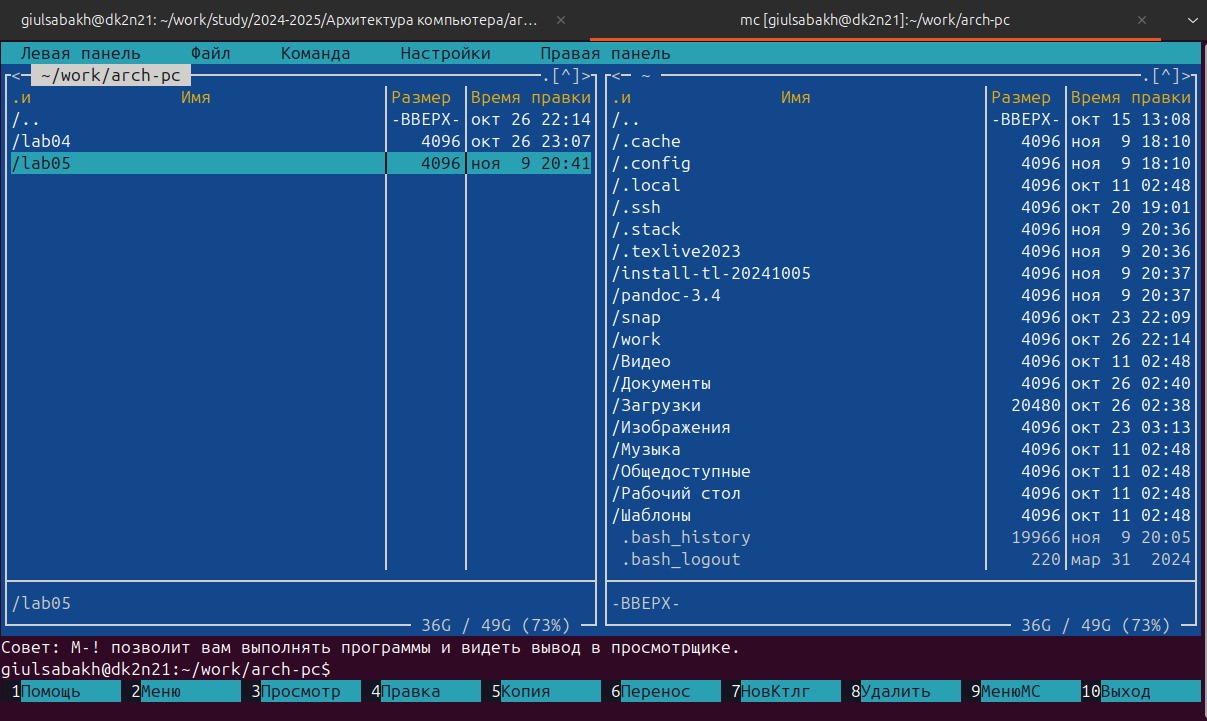


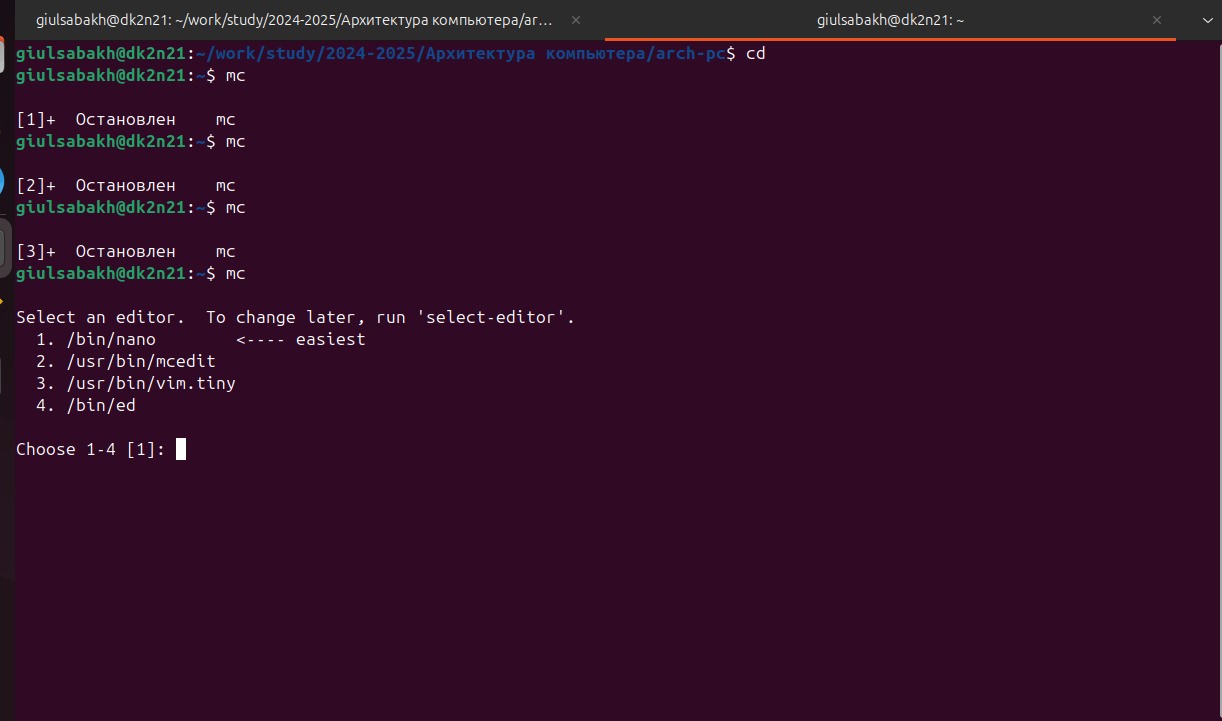
Рис. 4.3: Создание папки lab05

17

используются редакторы

18

Рис. 4.5: Выбор редактора



4.6).

mcedit

nano

Midnight Commander

(рис. 4.5,

или

дактора

тирования во встроенном редакторе. Как правило в качестве встроенного ре-

lab5-1.asm

F4

для редак-

откроем файл

С помощью функциональной клавиши

Рис. 4.4: Создание файла lab5-1.asm

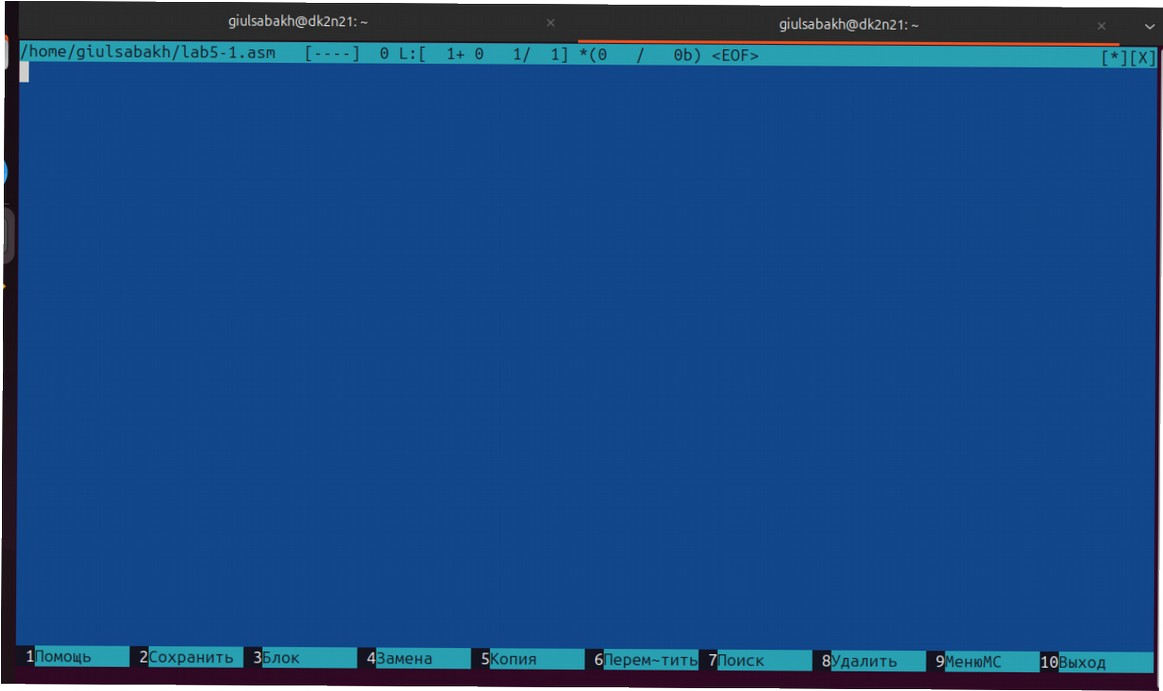
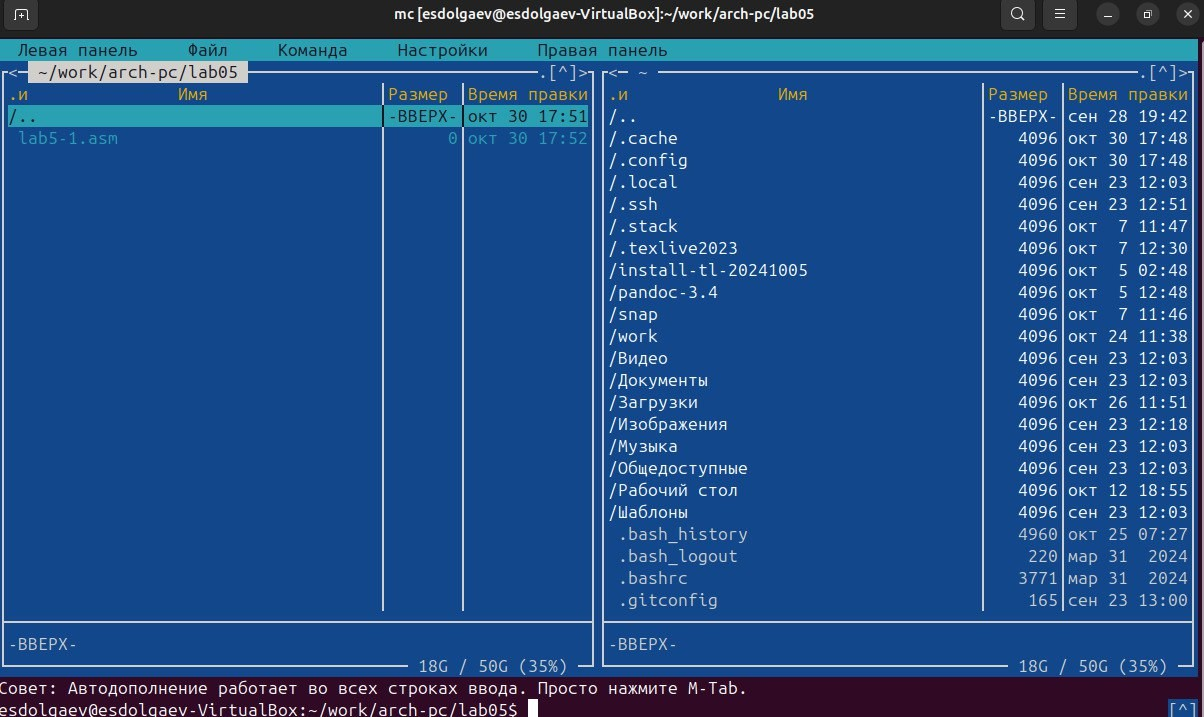


Рис. 4.6: Редактор mcedit

Введём текст программы, сохраним изменения и закроем файл(рис. 4.7).

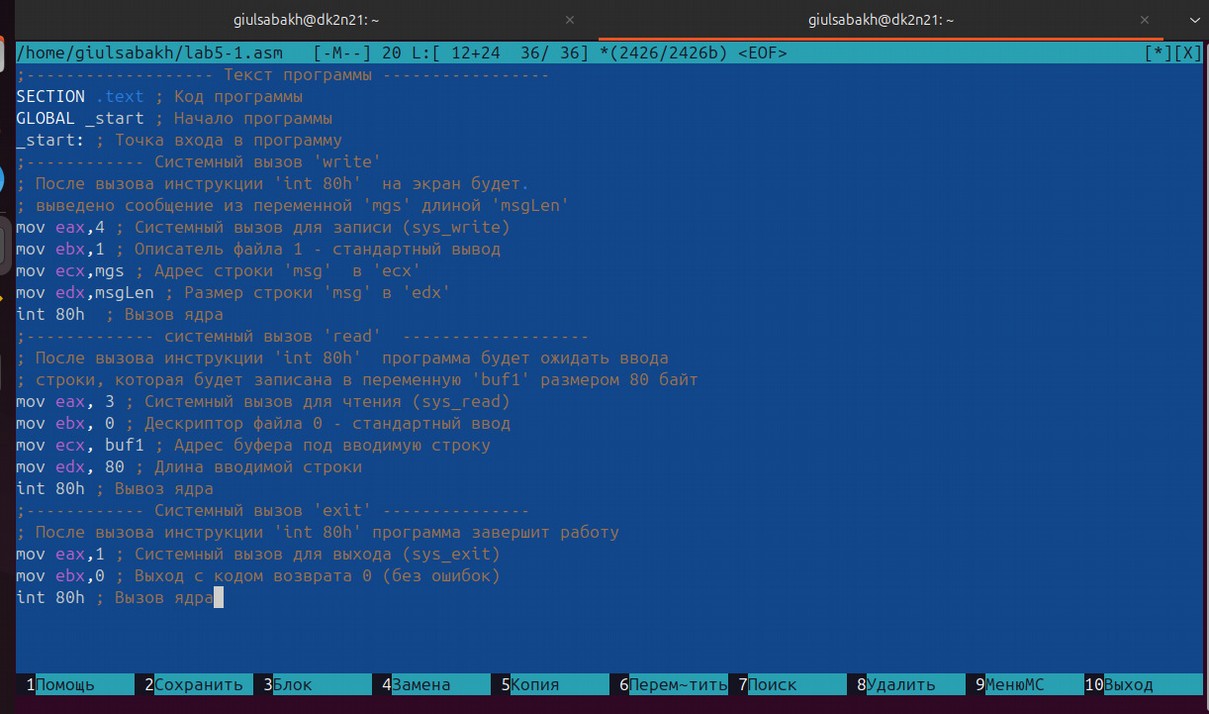


Рис. 4.7: Текст программы

С помощью функциональной клавиши

откроем файл

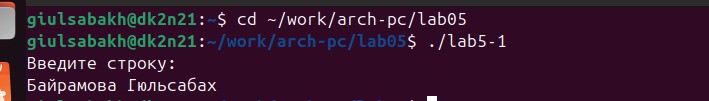
для про-

F3

lab5-1.asm

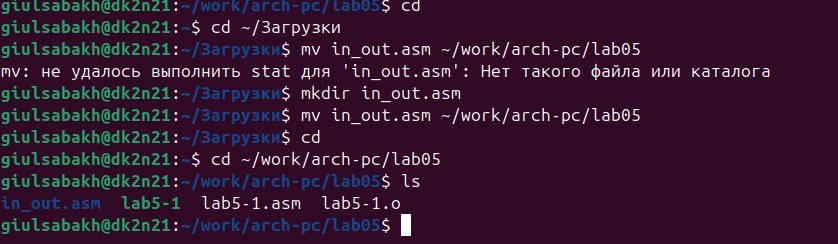
смотра. Убедимся, что файл содержит текст программы(рис. 4.8).

19



20

Рис. 4.11: Перемещение файла in\_out.asm



папку, где находится файл с программой(рис. 4.11).

in\_out.asm

со страницы курса в ТУИС и переместим его в

Далее, скачаем файл

Рис. 4.10: Работа исполняемого файла

Рис. 4.9: Создание исполняемого файла



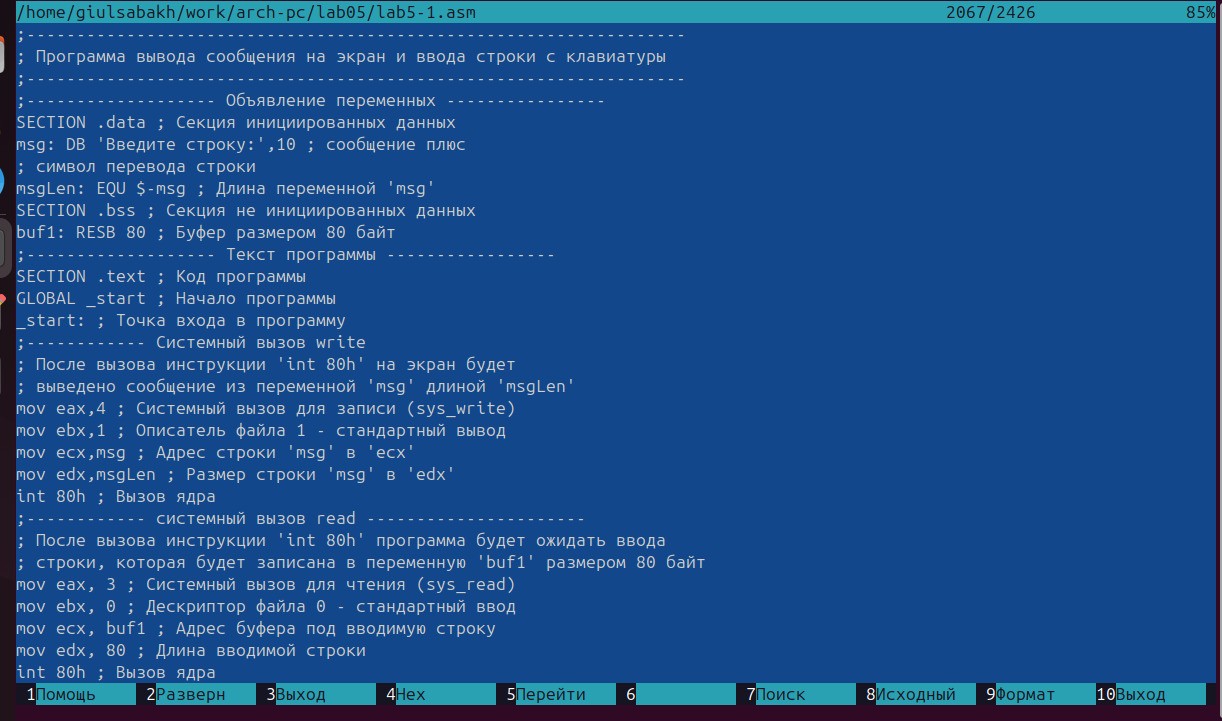
запрос введём ФИО(рис. 4.9, 4.10).

Программа выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. На

компоновку объектного файла и запустим получившийся исполняемый файл.

Оттранслируем текст программы lab5-1.asm в объектный файл. Выполним

Рис. 4.8: Проверка



С помощью функциональной клавиши F6 создим копию файла lab5-1.asm с

именем lab5-2.asm(рис. 4.12).

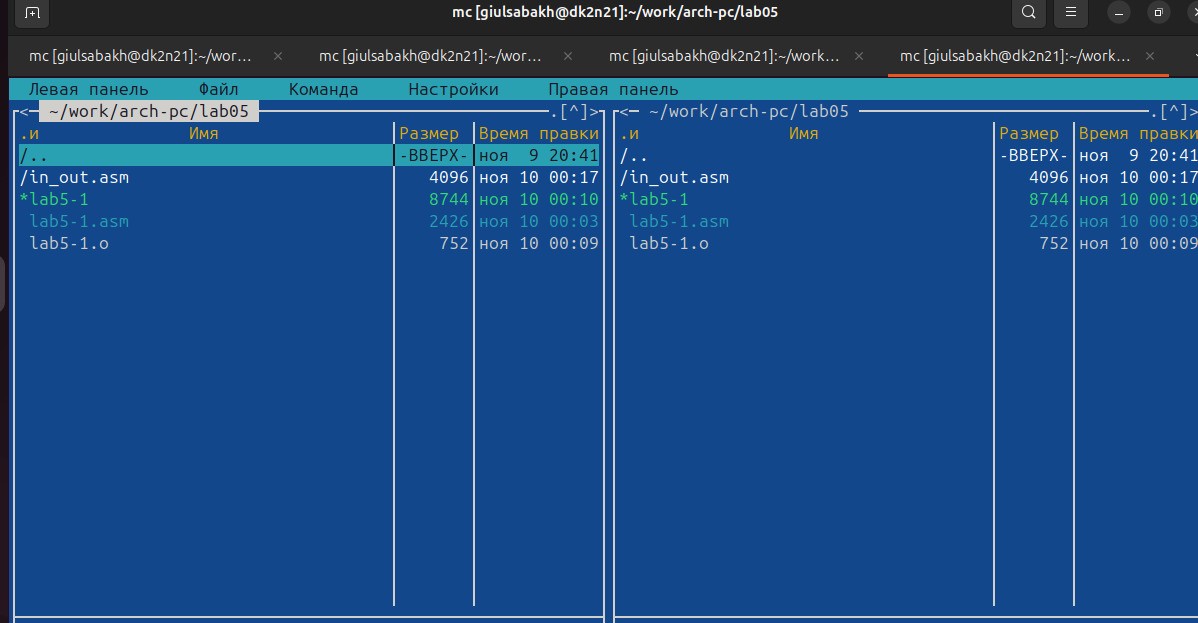


Рис. 4.12: Создание копии файла lab5-1.asm

Исправим текст программы в файле

с использование подпрограмм

lab5-2.asm

из внешнего файла

. Создадим исполняемый файл и проверим его

in\_out.asm

работу(рис. 4.13).

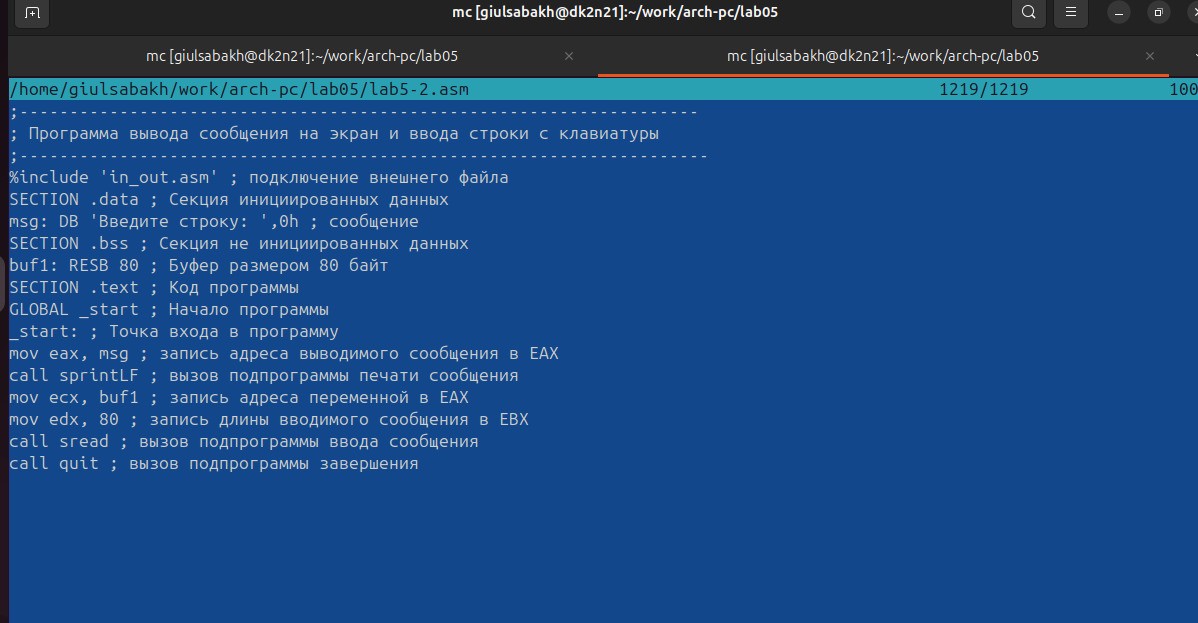


Рис. 4.13: Текст программы

21

зования внешнего файла

22

строку введите свою фамилию(рис. 4.18).

Получим исполняемый файл и проверим его работу. На приглашение ввести

• вывести введённую строку на экран.

• ввести строку с клавиатуры;

• вывести приглашение типа “Введите строку:”;

алгоритму(рис. 4.16, 4.17):

in\_out.asm

), так чтобы она работала по следующему

lab5-1.asm

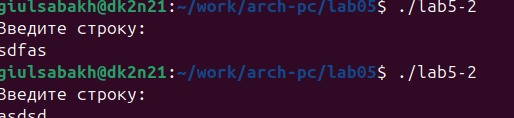
. Внесём изменения в программу (без исполь-

Создадим копию файла

**4.1 Задания для самостоятельной работы**

Теперь вводимая строка выводится ниже строки ‘Введите строку:’.

Рис. 4.15: Создание и работа испоняемого файла



емый файл и проверим его работу(рис. 4.15).

В файле lab5-2.asm заменим подпрограмму sprintLF на sprint. Создим исполня-

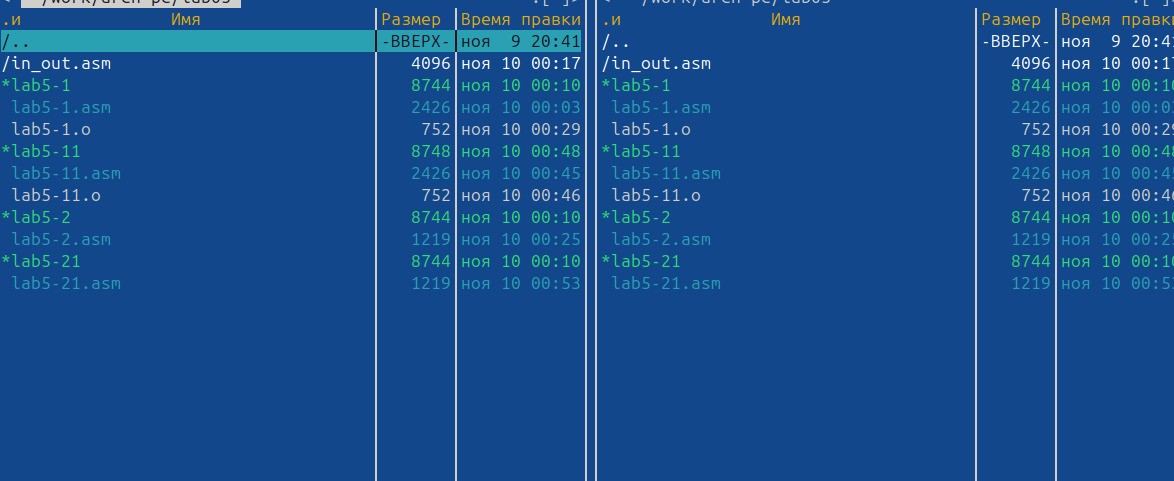


Рис. 4.16: Создание копии файла lab5-1.asm

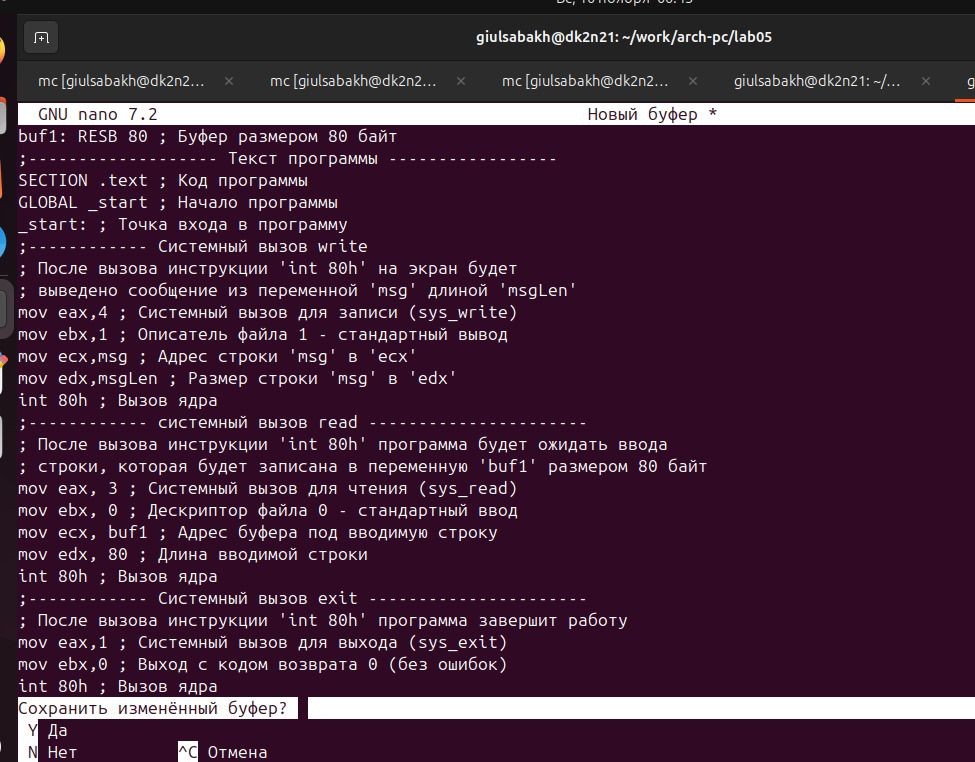


Рис. 4.17: Измененнный текст программы

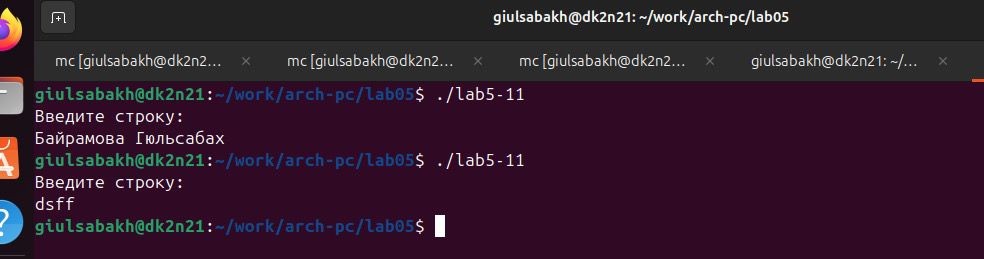


Рис. 4.18: Создание и работа испоняемого файла

Создадим копию файла

. Исправим текст программы с использо-

lab5-2.asm

вание подпрограмм из внешнего файла

, так чтобы она работала по

in\_out.asm

следующему алгоритму(рис. 4.19):

23

• вывести приглашение типа “Введите строку:”;

• ввести строку с клавиатуры;

• вывести введённую строку на экран.

Создадим исполняемый файл и проверьте его работу(рис. 4.19).

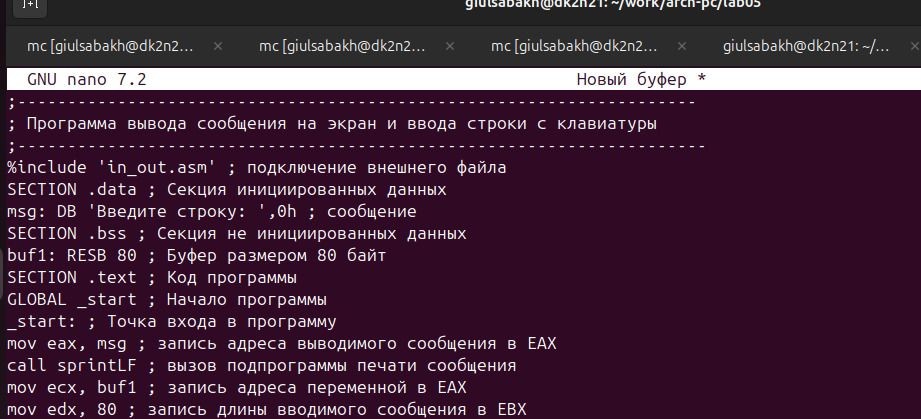


Рис. 4.19: Измененнный текст программы

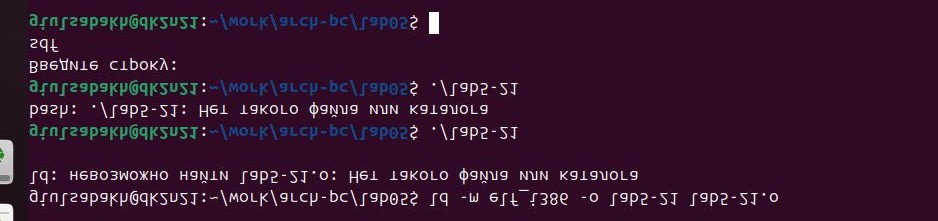


Рис. 4.20: Создание и работа испоняемого файла

24

**5 Выводы**

В ходе выполнения данной лабораторной работы я приобрёл практические

навыки работы в *Midnight Commander* и освоил инструкций языка ассемблера

mov

и

.

int

25

**Список литературы**

26