

Российский Университет Дружбы Народов

Факультет физико-математических и естественных наук

Отчет по лабораторной работе №5

“ Основы работы с Midnight Commander (mc).

Структура программы на языке ассемблера NASM.

Системные вызовы в ОС GNU Linux.”

Студент: Богомолова Полина Петровна

Группа: НКАбд-01-25

Москва, 2025 год

Оглавление

| | |
|---|----|
| Цель работы..... | 2 |
| Теоретическое введение..... | 3 |
| Основы работы с Midnight Commander..... | 3 |
| Рис. 1..... | 4 |
| Рис. 2..... | 4 |
| Структура программы на языке ассемблера NASM..... | 5 |
| Элементы программирования..... | 6 |
| Рис. 3..... | 7 |
| Описание инструкции int..... | 8 |
| Системные вызовы для обеспечения диалога с пользователем..... | 8 |
| Порядок выполнения лабораторной работы..... | 11 |
| Рис. 4..... | 11 |
| Рис. 5..... | 11 |
| Рис. 6..... | 12 |
| Рис. 7..... | 12 |
| Рис. 8..... | 13 |
| Рис. 9..... | 13 |
| Рис. 10..... | 14 |
| Рис. 11..... | 14 |
| Рис. 12..... | 15 |
| Рис. 13..... | 15 |
| Рис. 14..... | 16 |
| Рис. 15..... | 16 |
| Рис. 16..... | 17 |
| Рис. 17..... | 17 |
| Рис. 18..... | 18 |
| Рис. 19..... | 18 |
| Рис. 20..... | 19 |
| Рис. 21..... | 19 |
| Задания для самостоятельной работы..... | 19 |
| Рис. 22..... | 20 |
| Рис. 23..... | 20 |
| Рис. 24..... | 21 |
| Рис. 25..... | 21 |
| Рис. 26..... | 22 |
| Рис. 27..... | 22 |
| Вывод..... | 23 |

| | |
|------------------------|----|
| Список литературы..... | 24 |
|------------------------|----|

Цель работы

Приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера mov и int.

Теоретическое введение

Основы работы с Midnight Commander

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. Для активации оболочки Midnight Commander достаточно ввести в командной строке mc и нажать клавишу Enter.

В Midnight Commander используются функциональные клавиши F1 - F10, к которым привязаны часто выполняемые операции. Основные клавиши и выполняемые с помощью них действия представлены на рисунках 1-2.

Рис. 1

| Функцио- нальные клавиши | Выполняемое действие |
|--------------------------------|---|
| F1 | вызов контекстно-зависимой подсказки |
| F2 | вызов меню, созданного пользователем |
| F3 | просмотр файла, на который указывает подсветка в активной панели |
| F4 | вызов встроенного редактора для файла, на который указывает подсветка в активной панели |

Рис. 2

| Функциональные клавиши | Выполняемое действие |
|------------------------|---|
| F5 | копирование файла или группы отмеченных файлов из каталога, отображаемого в активной панели, в каталог, отображаемый на второй панели |
| F6 | перенос файла или группы отмеченных файлов из каталога, отображаемого в активной панели, в каталог, отображаемый на второй панели |
| F7 | создание подкаталога в каталоге, отображаемом в активной панели |
| F8 | удаление файла (подкаталога) или группы отмеченных файлов |
| F9 | вызов основного меню программы |
| F10 | выход из программы |

Следующие комбинации клавиш облегчают работу с Midnight Commander:

- **Tab** используется для переключения между панелями;
- **↑** и **↓** используется для навигации, **Enter** для входа в каталог или открытия файла (если в файле расширений `mc.ext` заданы правила связи определённых расширений файлов с инструментами их запуска или обработки);
- **Ctrl + u** (или через меню Команда > Переставить панели) меняет местами содержимое правой и левой панелей;
- **Ctrl + o** (или через меню Команда > Отключить панели) скрывает или возвращает панели Midnight Commander, за которыми доступен для работы командный интерпретатор оболочки и выводимая туда информация.
- **Ctrl + x + d** (или через меню Команда > Сравнить каталоги) позволяет сравнить содержимое каталогов, отображаемых на левой и правой панелях.

Структура программы на языке ассемблера NASM

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 инициализированных 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 (четверное 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.

Синтаксис директив определения данных следующий:

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

Для объявления неинициализированных данных в секции `.bss` используются директивы `resb`,

`resw`, `resd` и другие, которые сообщают ассемблеру, что необходимо зарезервировать заданное количество ячеек памяти.

Элементы программирования

Инструкция языка ассемблера `mov` предназначена для дублирования данных 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). На рисунке 3 приведены примеры использования mov с различными операндами.

Рис. 3

| Тип операндов | Пример | Пояснение |
|-------------------|-----------------|---|
| mov <reg>,<reg> | mov eax,ebx | пересылает значение регистра ebx в регистр eax |
| mov <reg>,<mem> | mov cx,[eax] | пересылает в регистр cx значение из памяти, указанной в eax |
| mov <mem>,<reg> | mov rez,ebx | пересылает в переменную rez значение из регистра ebx |
| mov <reg>,<const> | mov eax,403045h | пишет в регистр eax значение 403045h |
| mov <mem>,<const> | mov byte[rez],0 | записывает в переменную 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 следующих примеров приведет к ошибке:

- mov al,1000h — ошибка, попытка записать 2-байтное число в 1-байтный регистр;
- mov eax,cx — ошибка, размеры операндов не совпадают.

Описание инструкции `int`

Инструкция языка ассемблера `int` предназначена для вызова прерывания с указанным номером. В общем виде она записывается в виде

`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). После вызова инструкции `int 80h` выполняется системный вызов какой-либо функции

ядра Linux. При этом происходит передача управления ядру операционной системы. Чтобы

узнать, какую именно системную функцию нужно выполнить, ядро извлекает номер системного вызова из регистра `eax`. Поэтому перед вызовом прерывания необходимо поместить в

этот регистр нужный номер. Кроме того, многим системным функциям требуется передавать

какие-либо параметры. По принятым в ОС Linux правилам эти параметры помещаются в порядке следования в остальные регистры процессора: `ebx`, `ecx`, `edx`. Если системная функция

должна вернуть значение, то она помещает его в регистр `eax`.

Системные вызовы для обеспечения диалога с пользователем

Простейший диалог с пользователем требует наличия двух функций — вывода текста на

экран и ввода текста с клавиатуры. Простейший способ вывести строку на экран — использовать системный вызов `write`. Этот системный вызов имеет номер 4, поэтому перед вызовом

инструкции `int` необходимо поместить значение 4 в регистр `eax`. Первым аргументом `write`,

помещаемым в регистр `ebx`, задаётся дескриптор файла. Для вывода на экран в качестве

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

Вторым аргументом задаётся адрес выводимой строки (помещаем его в регистр `ecx`, например, инструкцией `mov ecx, msg`). Строка может иметь любую длину. Последним аргументом

(т.е. в регистре `edx`) должна задаваться максимальная длина выводимой строки.

Для ввода строки с клавиатуры можно использовать аналогичный системный вызов `read`.

Его аргументы — такие же, как у вызова `write`, только для «чтения» с клавиатуры используется

файловый дескриптор 0 (стандартный ввод).

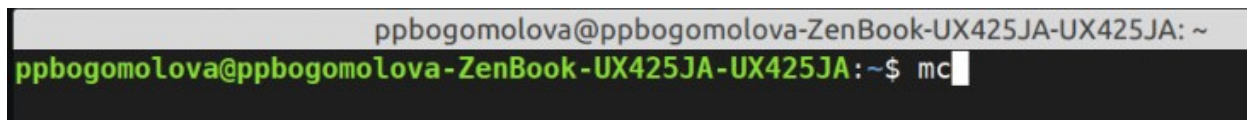
Системный вызов `exit` является обязательным в конце любой программы на языке ассемблер. Для обозначения конца программы перед вызовом инструкции `int 80h` необходимо

поместить в регистр `eax` значение 1, а в регистр `ebx` код завершения 0.

Порядок выполнения лабораторной работы

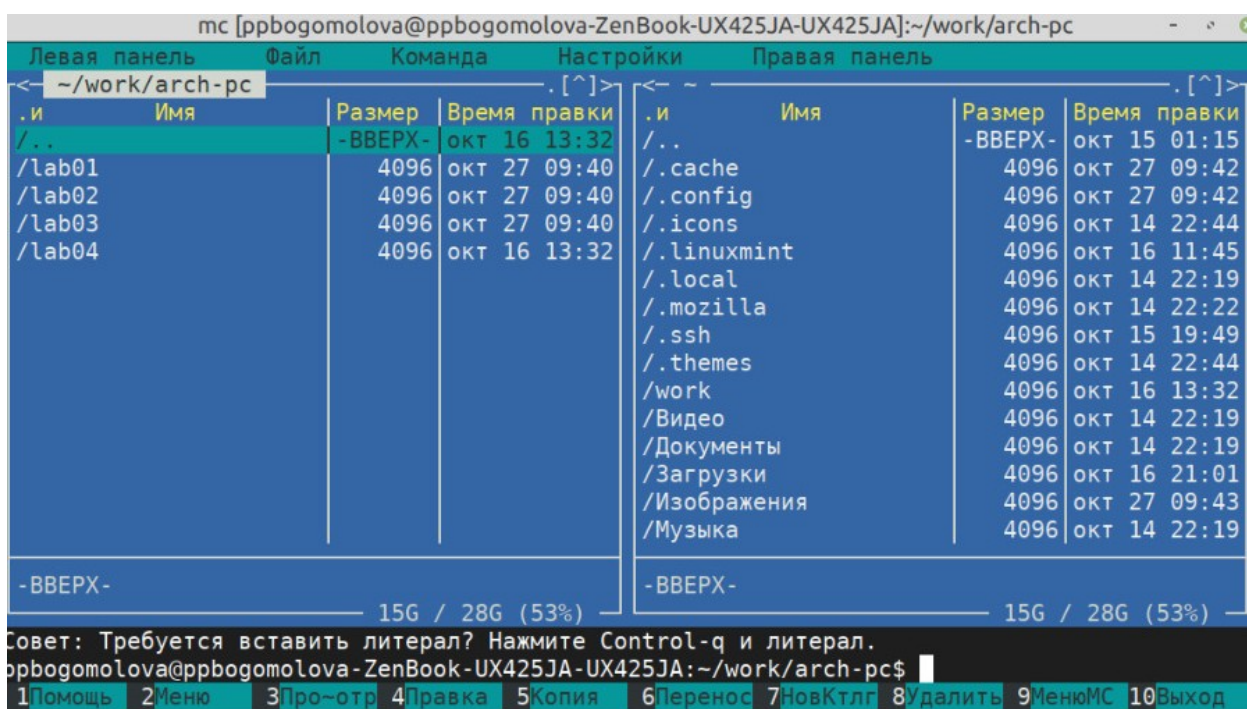
1. Откроем Midnight Commander с помощью команды `mc`. Результат представлен на рисунке 4.

Рис. 4



2. Пользуясь клавишами `↑`, `↓` и `Enter` перейдем в каталог `~/work/arch-pc`, созданный при выполнении лабораторной работы №4. Результат представлен на рисунке 5.

Рис. 5



3. С помощью функциональной клавиши `F7` создадим папку `lab05` и перейдем в созданный каталог. Результат представлен на рисунках 6-8.

Рис. 6

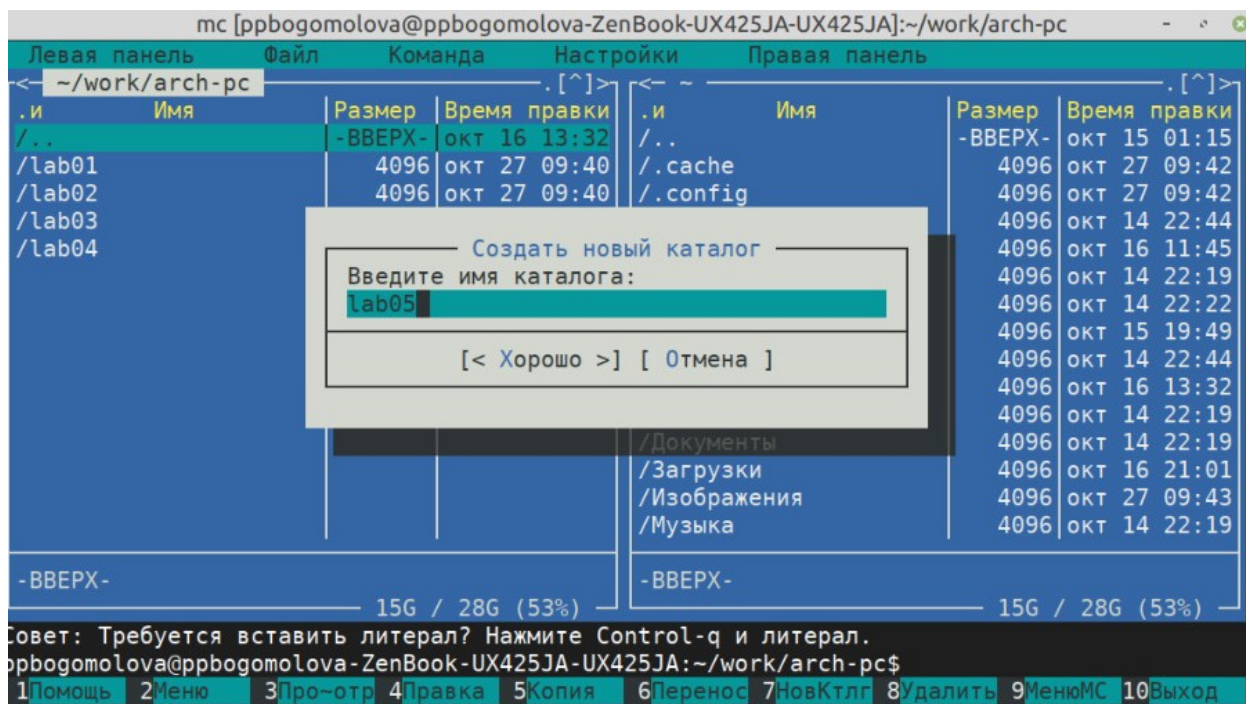


Рис. 7

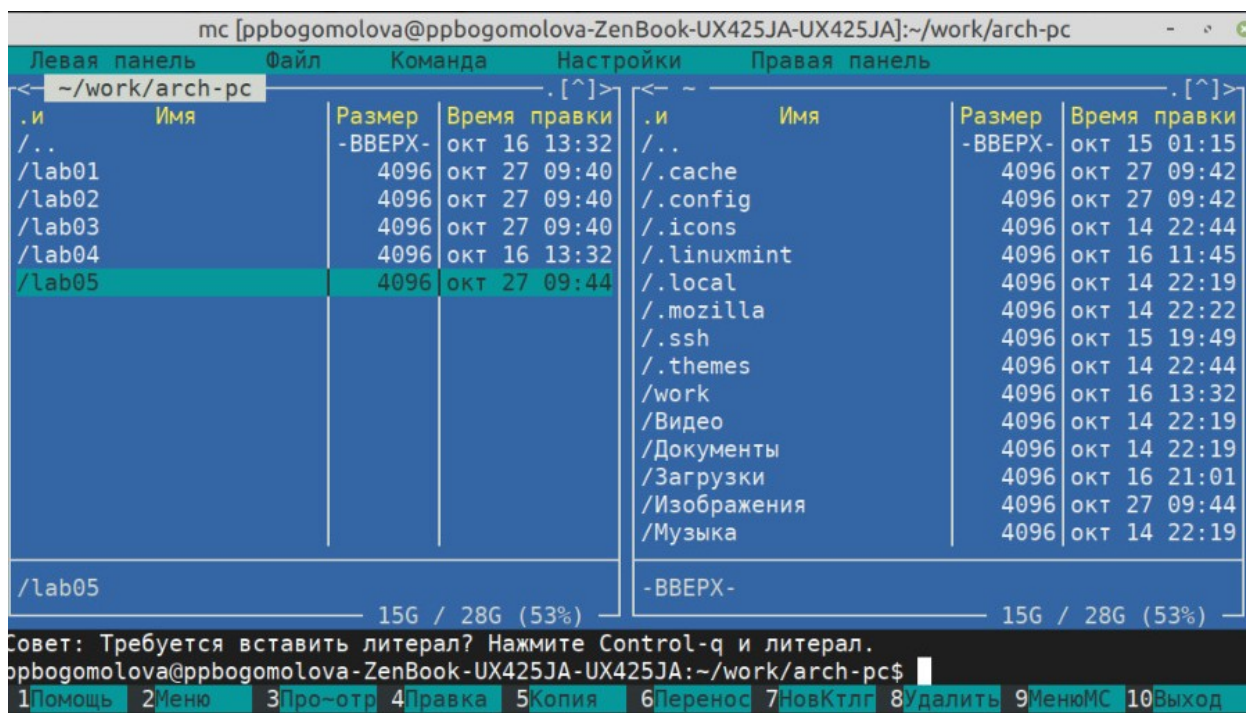
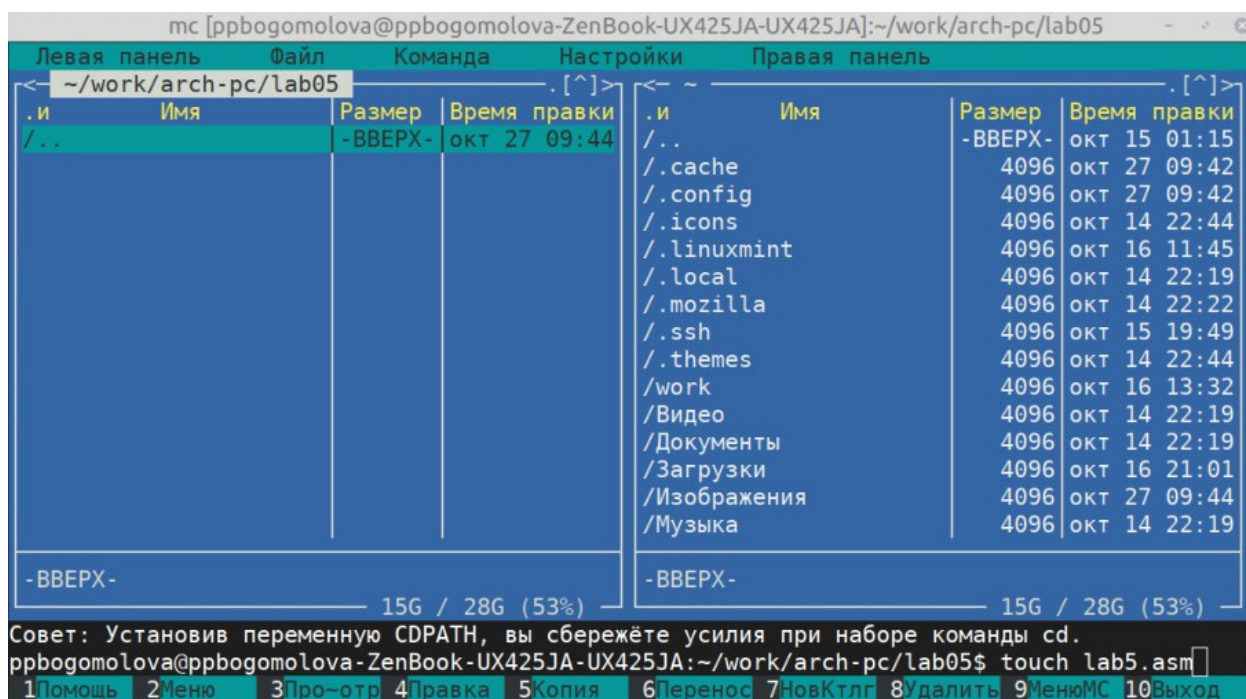


Рис. 8



4. Пользуясь строкой ввода и командой touch создадим файл lab5-1.asm. Результат представлен на рисунках 9-10.

Рис. 9

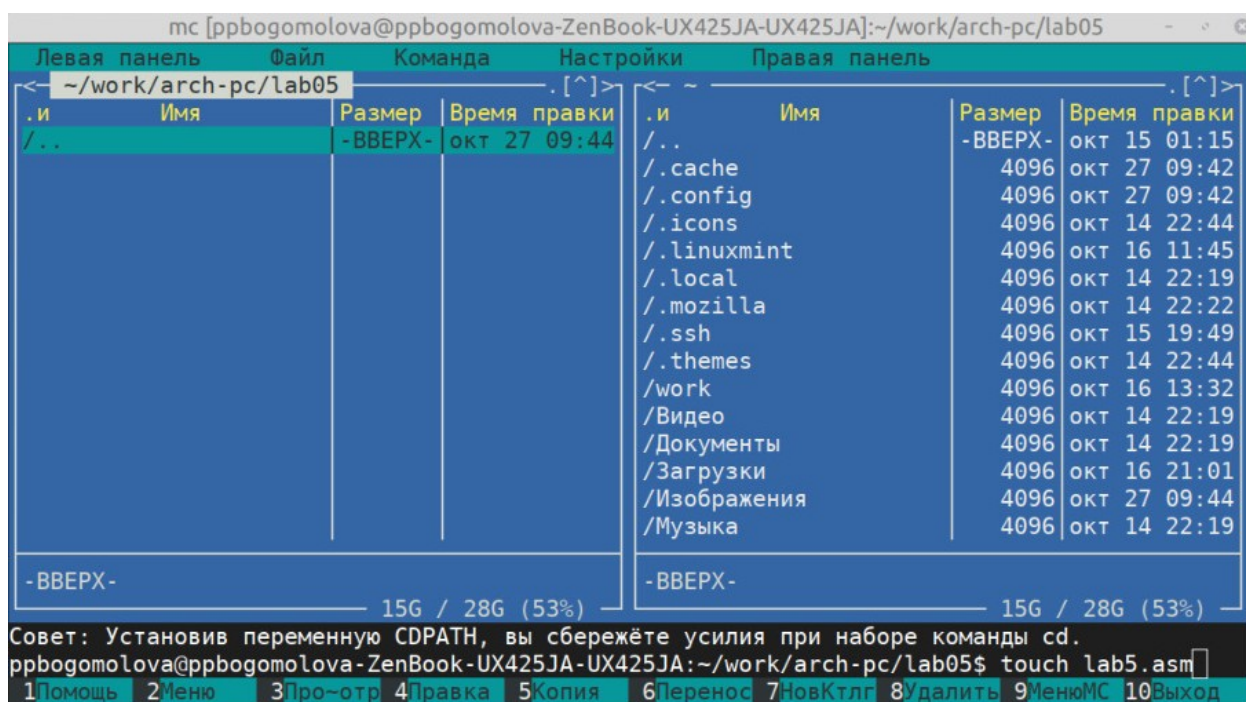
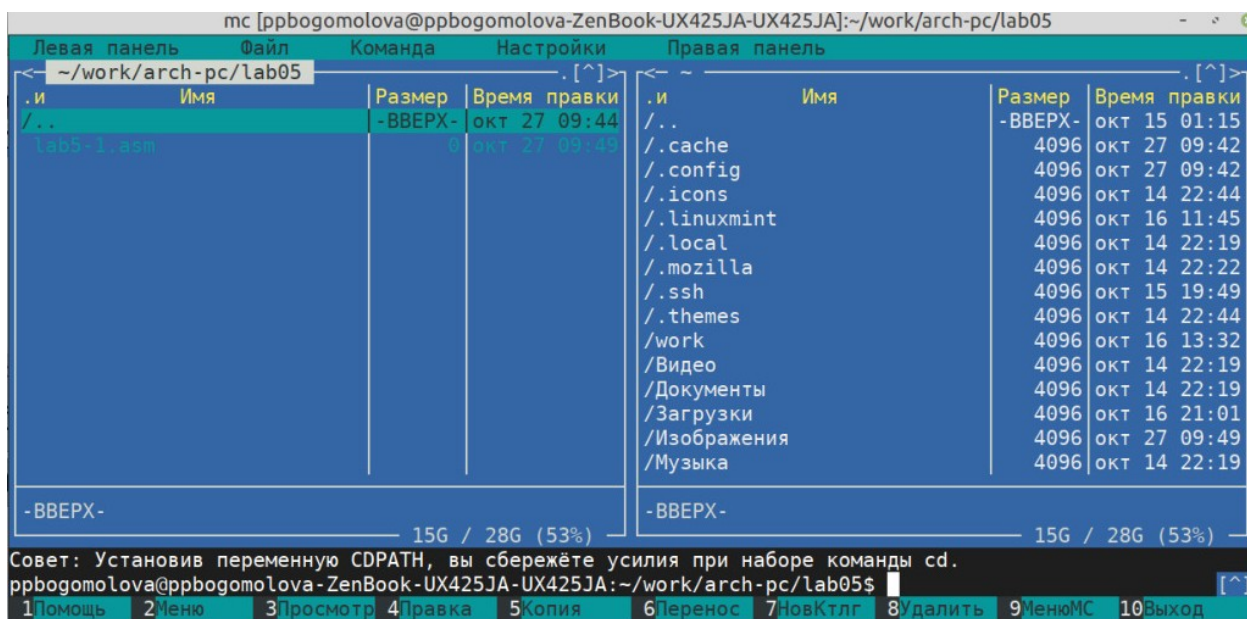
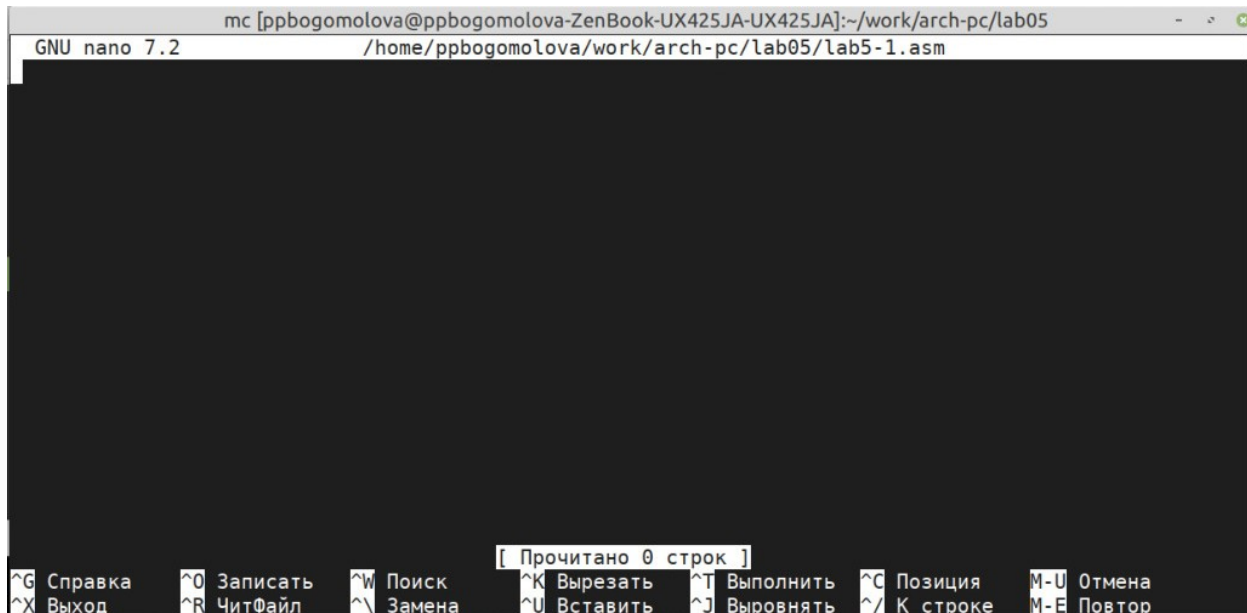


Рис. 10



5. С помощью функциональной клавиши F4 откройте файл lab5-1.asm для редактирования во встроенном редакторе nano. Результат представлен на рисунке 11.

Рис. 11



6. Введем текст программы, сохраним изменения и закроем файл. При этих действиях используем клавиши: ctrl + x, y, enter. Результат представлен на рисунке 12.

Рис. 12

```

GNU nano 7.2
mc [ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA]~/work/arch-pc/lab05
/home/ppbogomolova/work/arch-pc/lab05/lab5-1.asm *

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

```

Справка Выход Записать ЧитФайл Поиск Замена Вырезать Вставить Выполнить Выровнять Позиция К строке Отмена Повтор Установить метку На скобку Копировать Обр. поиск

7. С помощью функциональной клавиши F3 откроем файл lab5-1.asm для просмотра. Убедимся, что файл содержит текст программы. Результат представлен на рисунке 13.

Рис. 13

```

/home/ppbogomolova/work/arch-pc/lab05/lab5-1.asm 285/285 100%
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

```

1Помощь 2Разверн 3Выход 4Чех 5Перейти 6 7Поиск 8Исходный 9Формат 10Выход

8. Оттранслируем текст программы lab5-1.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл.

При этом используем команды `nasm -f elf, ld -m elf_i386 -o, ./` . Программа выводит строку 'Введите строку:' и ожидает ввода с клавиатуры. На запрос введем ФИО. Результат представлен на рисунке 14.

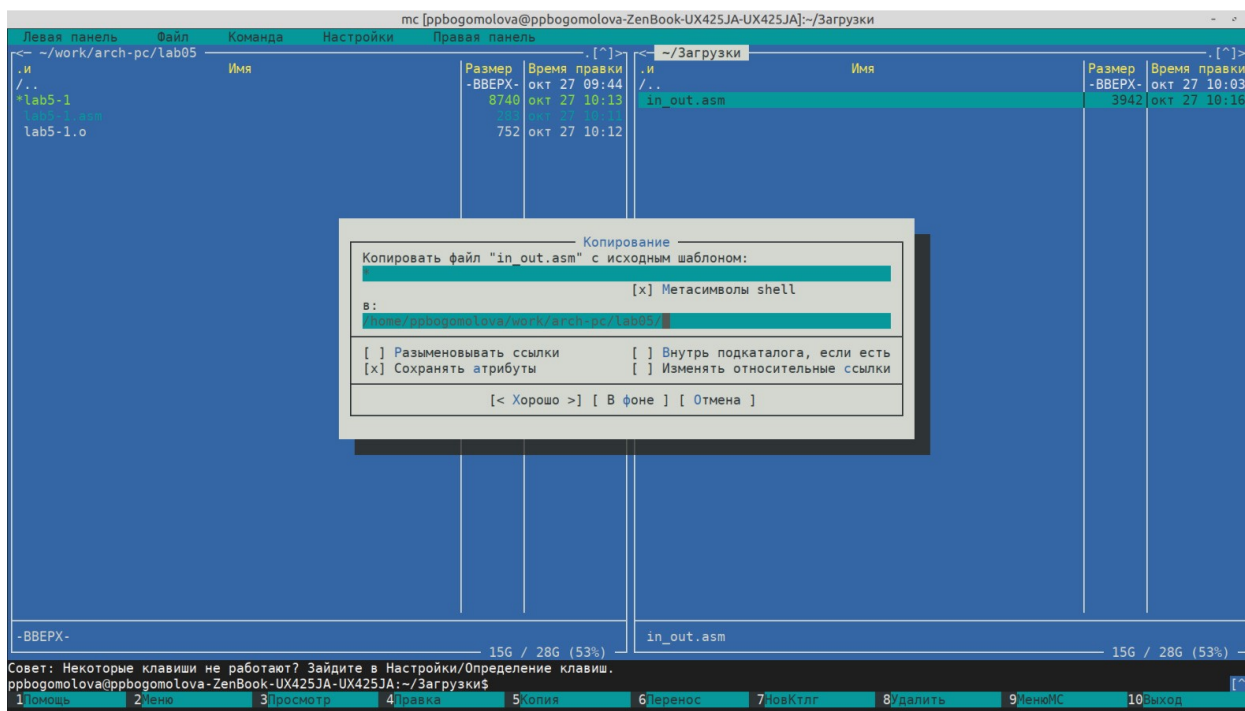
Рис. 14

```
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA: ~/work/arch-pc/lab05
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$ ./lab5-1
Введите строку
Богомолова Полина Петровна
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$
```

9. Скачаем файл `in_out.asm` со страницы в ТУИС.

10. Подключаемый файл `in_out.asm` должен лежать в том же каталоге, что и файл с программой, в которой он используется. В одной из панелей `mc` откроем каталог с файлом `lab5-1.asm`. В другой панели каталог со скаченным файлом `in_out.asm` (для перемещения между панелями используем `Tab`). Скопируем файл `in_out.asm` в каталог с файлом `lab5-1.asm` с помощью функциональной клавиши `F5` . Результат представлен на рисунке 15.

Рис. 15



11. С помощью функциональной клавиши `F6` создадим копию файла `lab5-1.asm` с именем `lab5-2.asm`. Выделим файл `lab5-1.asm`, нажмем клавишу `F6` , введем

имя файла lab5-2.asm и нажмем клавишу Enter. Результат представлен на рисунках 16-17.

Рис. 16

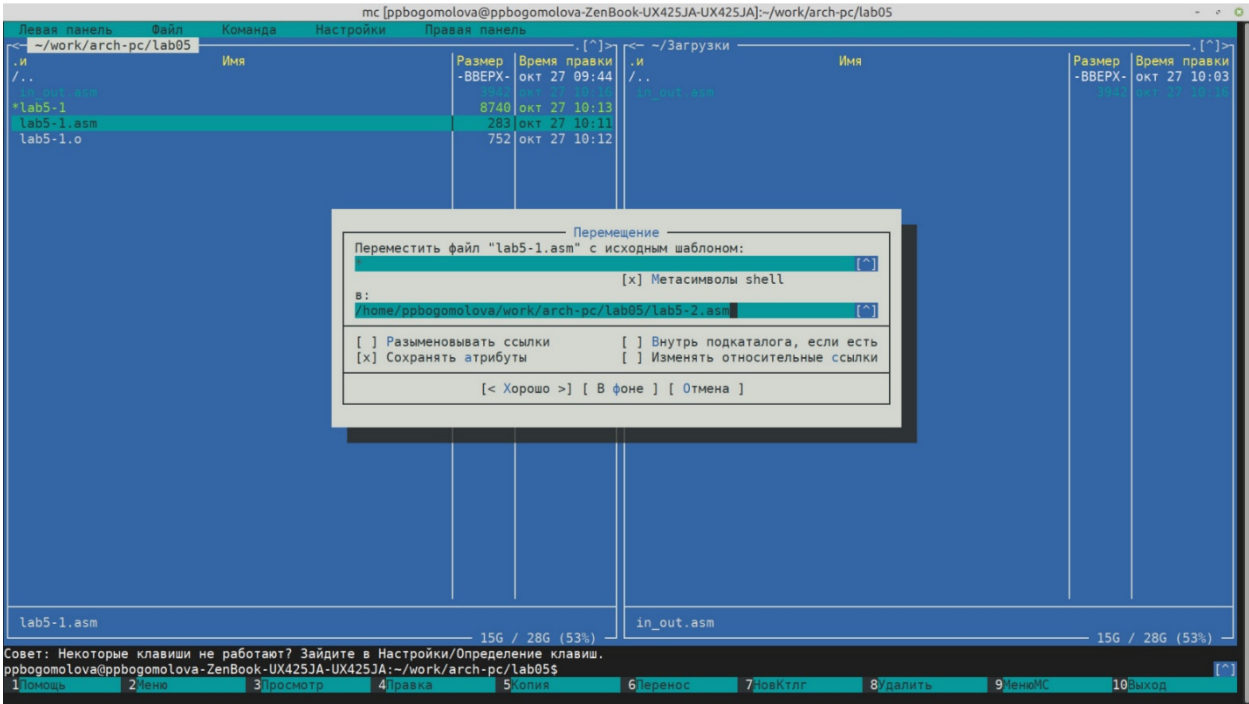
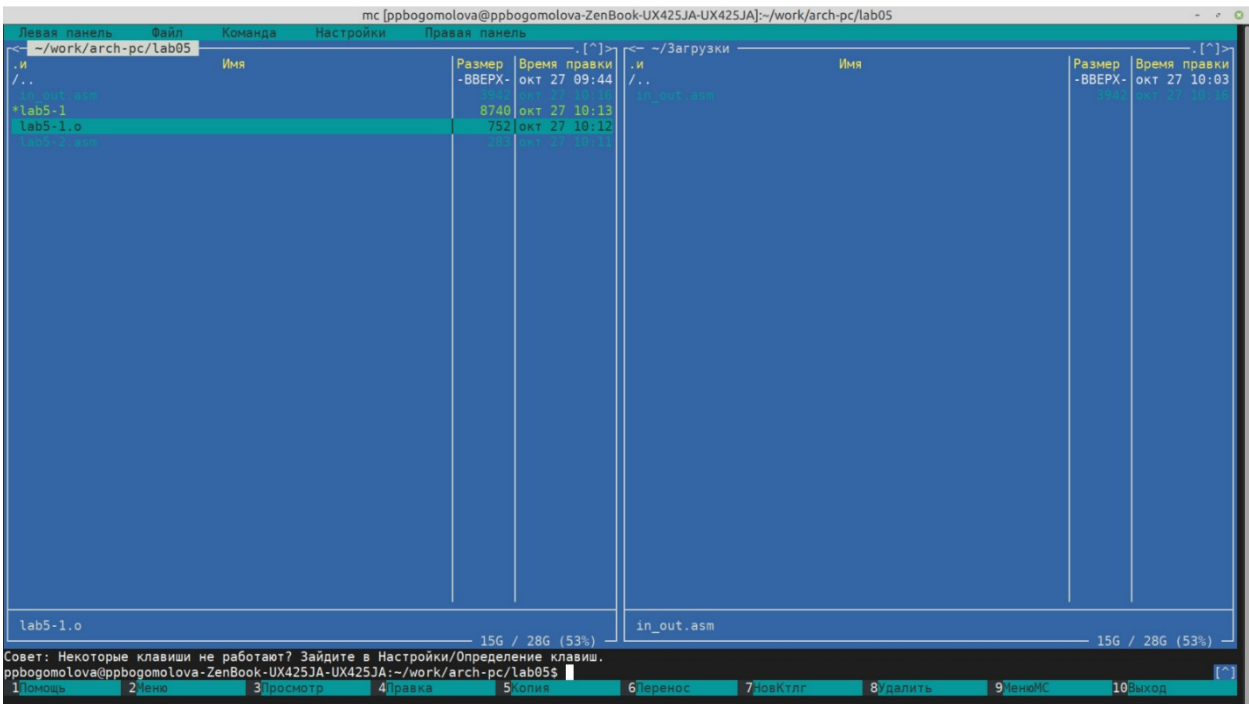


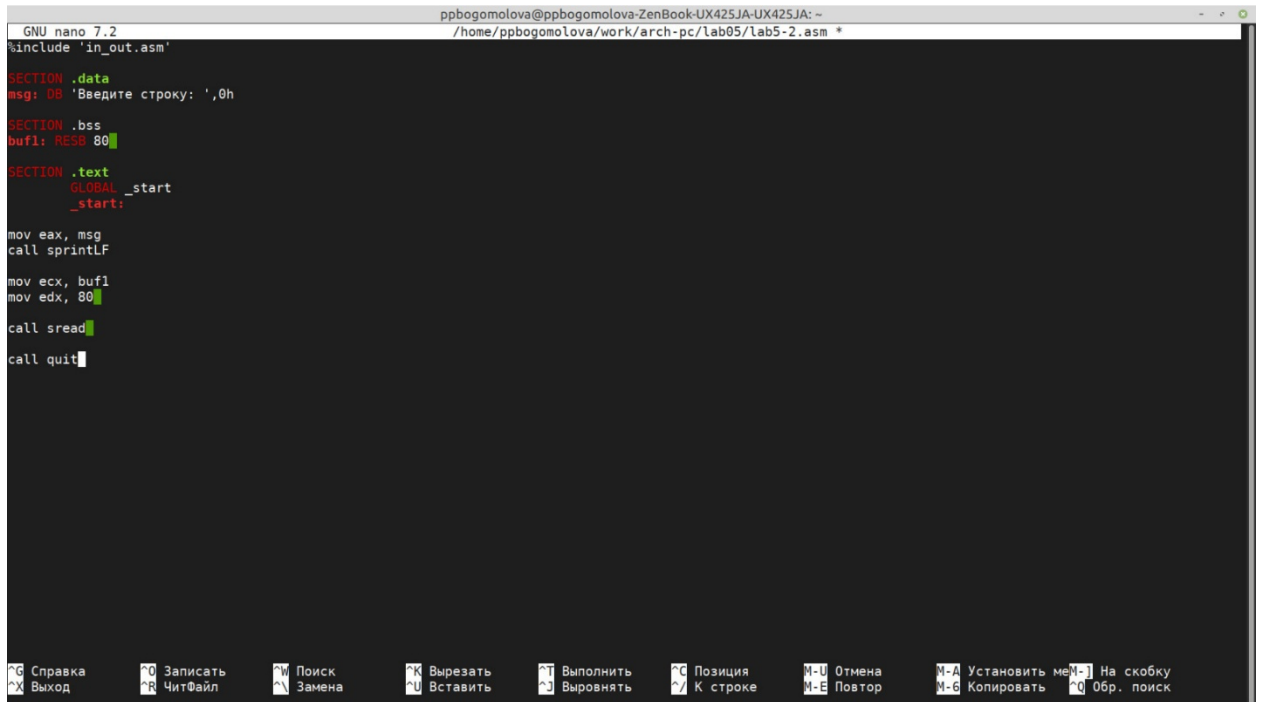
Рис. 17



12. Изменим текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in_out.asm (используем подпрограммы sprintLF, sread и quit). Создадим исполняемый файл и проверим его работу. При этом будем

использовать команды `nasm -f elf`, `ld -m elf_i386 -o, ./`. Результат представлен на рисунках 18-19.

Рис. 18



```
GNU nano 7.2 ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA: ~
/home/ppbogomolova/work/arch-pc/lab05/lab5-2.asm *
#include 'in_out.asm'

SECTION .data
msg: DB 'Введите строку: ',0h

SECTION .bss
buf1: RESB 80

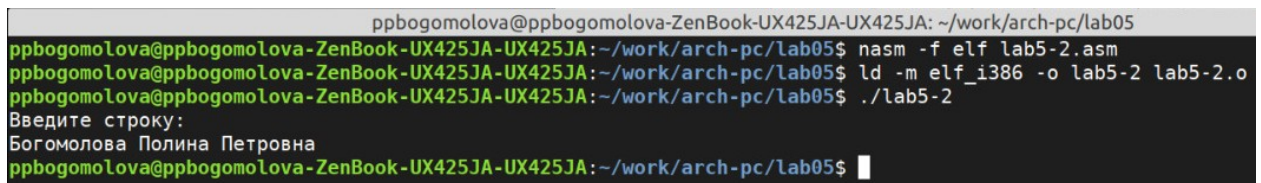
SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprintf

mov ecx, buf1
mov edx, 80

call read
call quit
```

Рис. 19



```
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA: ~/work/arch-pc/lab05
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Богомолова Полина Петровна
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$
```

13.В файле `lab5-2.asm` заменим подпрограмму `sprintf` на `print`. Создадим исполняемый файл и проверим его работу. Отличие программы с использованием `print` от программы с использованием `sprintf` заключается в том, что `sprintf` при выводе на экран добавляет к сообщению символ перевода строки. То есть `print` выводит сообщение на экран на одной строке с 'Введите строку', а `sprintf` выводит сообщение на экран, добавляя символ перевода строки, то есть не на одной строке с 'Введите строку:'. Результат представлен на рисунках 20-21.

Рис. 20



```
GNU nano 7.2 ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA: ~/work/arch-pc/lab05
/home/ppbogomolova/work/arch-pc/lab05/lab5-2.asm *
#include 'lab_out.asm'

SECTION .data
msg: DB 'Введите строку: ',0h

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

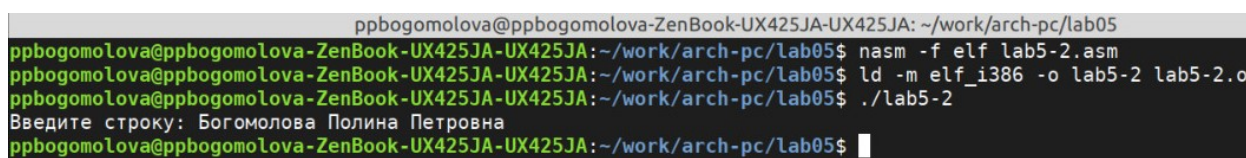
mov eax, msg
call sprint

mov ecx, buf1
mov edx, 80

call sread

call quit
```

Рис. 21



```
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA: ~/work/arch-pc/lab05
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Богомолова Полина Петровна
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$
```

Ссылка на мой репозиторий в github:

https://github.com/bogomolova-pp/study_2025-2026_arh-pc

Задания для самостоятельной работы

1-2. Создайте копию файла lab5-1.asm. Внесите изменения в программу (без использования внешнего файла in_out.asm), так чтобы она работала по следующему алгоритму:

- вывести приглашение типа “Введите строку:”;
- ввести строку с клавиатуры;
- вывести введенную строку на экран.

Получите исполняемый файл и проверьте его работу. На приглашение ввести строку

введите свою фамилию

Скопируем файл lab5-1.asm, назовем его lab5-1-copy.asm. Изменим программу. Создадим исполняемый файл и проверим его работу. При этом будем использовать команды `nasm -f elf, ld -m elf_i386 -o, ./`

Решение представлено на рисунке 22-24.

Рис. 22

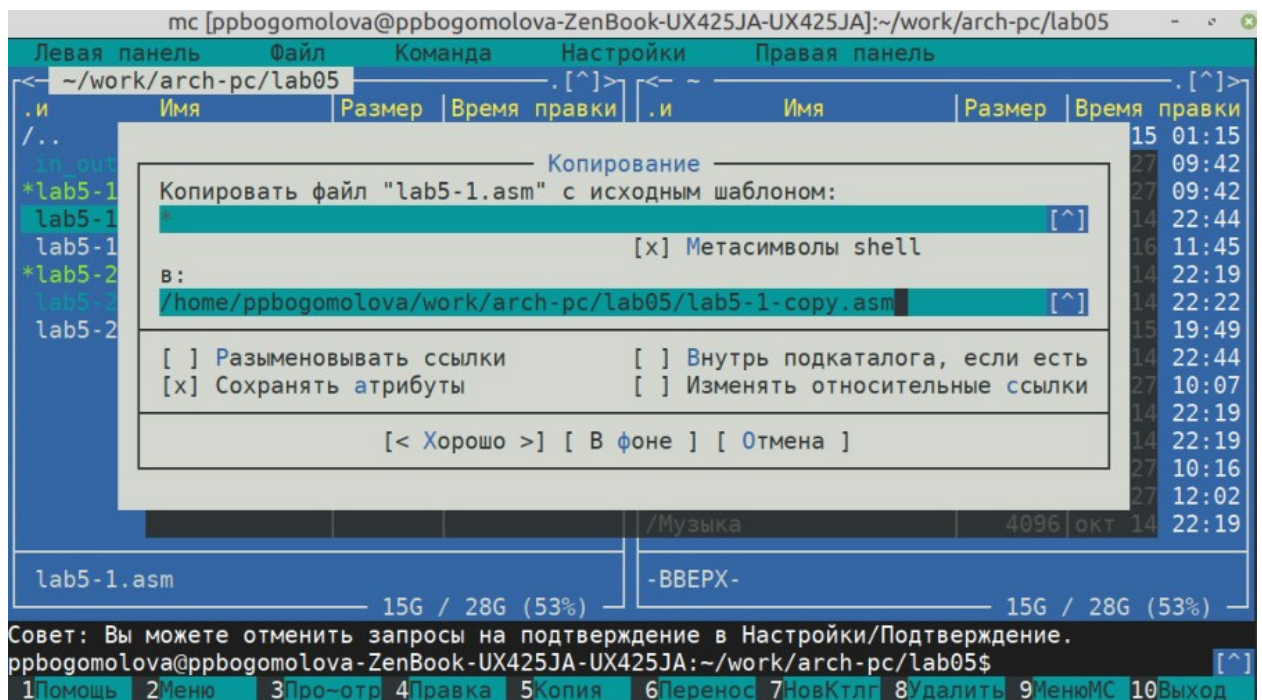


Рис. 23

```

GNU nano 7.2 ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA: ~/work/arch-pc/lab05
/home/ppbogomolova/work/arch-pc/lab05/lab5-1-copy.asm
SECTION .data
msg db 'Введите строку:', 10
msgLen equ $ - msg

SECTION .bss
buf 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, buf
mov edx, 80
int 80h

mov edx, eax
mov eax, 4
mov ebx, 1
mov ecx, buf
int 80h

mov eax, 1
mov ebx, 0
int 80h

```

Рис. 24

```

ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA: ~/work/arch-pc/lab05
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$ nasm -f elf lab5-1-copy.asm
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-copy lab5-1-copy.o
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$ ./lab5-1-copy
Введите строку:
Богомолова Полина Петровна
Богомолова Полина Петровна
ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA:~/work/arch-pc/lab05$

```

3-4. Создайте копию файла lab5-2.asm. Исправьте текст программы с использованием подпрограмм из внешнего файла in_out.asm, так чтобы она работала по следующему алгоритму:

- вывести приглашение типа “Введите строку:”;
- ввести строку с клавиатуры;
- вывести введенную строку на экран.

Создайте исполняемый файл и проверьте его работу

Скопируем файл lab5-2.asm, назовем его lab5-2-copy.asm. Изменим программу, использующую внешний файл in_out.asm. Создадим исполняемый файл и проверим его работу. При этом будем использовать команды `nasm -f elf`, `ld -m elf_i386 -o, ./`.

Решение представлено на рисунках

Рис. 25

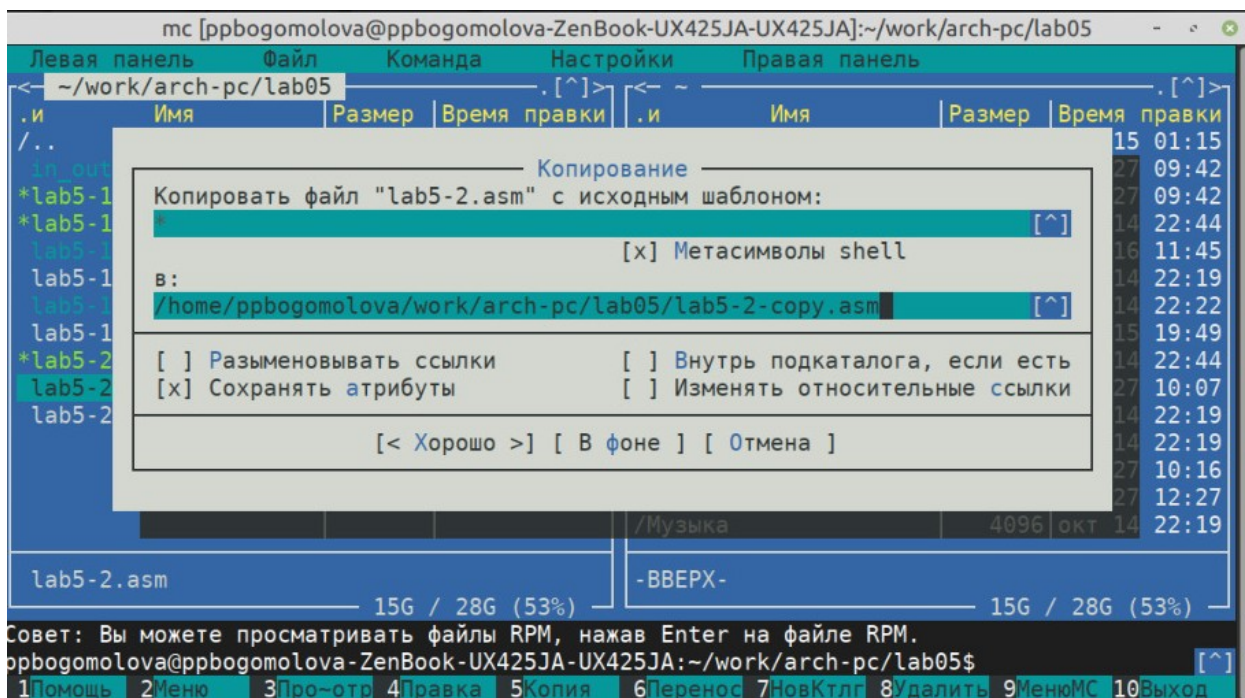


Рис. 26

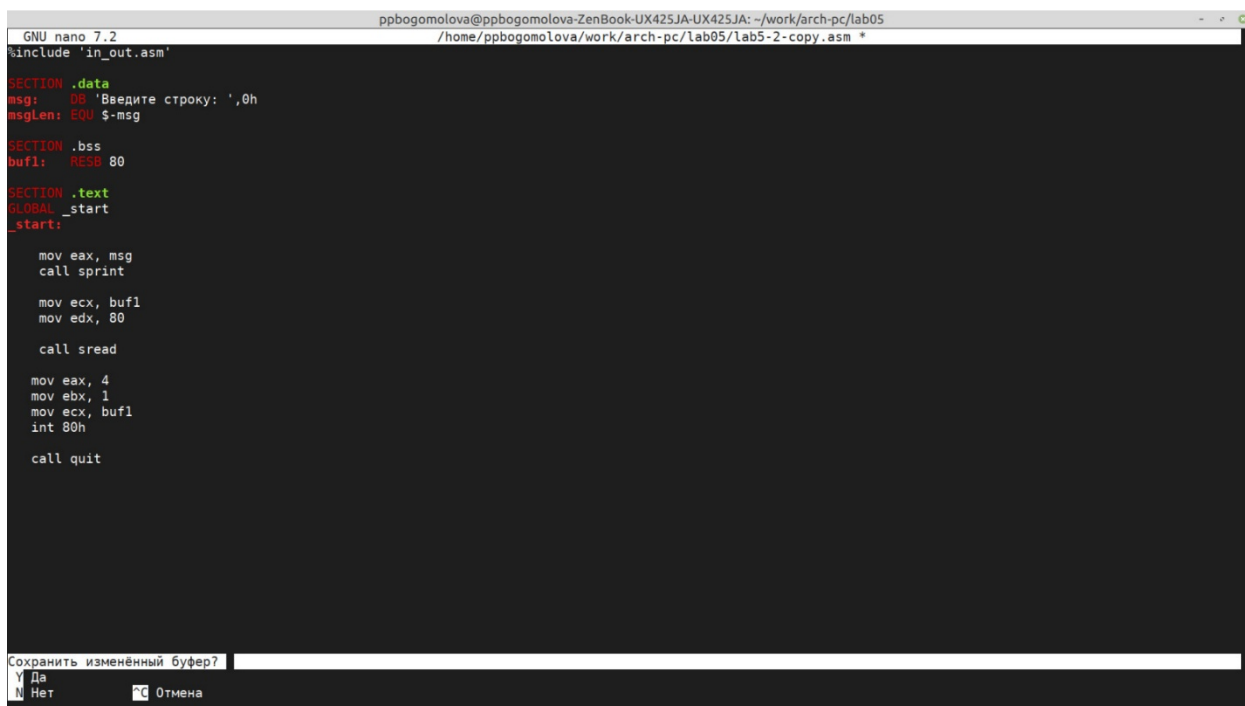
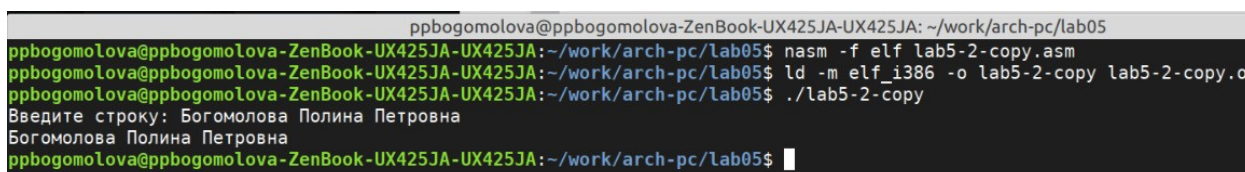


Рис. 27



Вывод

В результате выполнения лабораторной работы я приобрела практические навыки работы в Midnight Commander и освоила инструкции языка ассемблера `mov` и `int`.

Список литературы

1. Демидова А.В - Лабораторная работа №5. Основы работы с Midnight Commander (mc). Структура программы на языке ассемблера NASM. Системные вызовы в ОС GNU Linux