РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

по лабораторной работе № 5

дисциплина: Архитектура компьютера

Студент: Бессонов Андрей Максимович

Группа: НКАбд - 01 - 25

МОСКВА

2025 г.

Содержание

1. Цель работы 4
2. Теоретическое введение 5
3. Выполнение лабораторной работы 8
4. Выполнение самостоятельной работы 16
5. Выводы 19

Список иллюстраций

Рис 3.1: 31 8
Рис 3.2: 32 9
Рис 3.3: 33 10
Рис 3.4: 34 11
Рис 3.5: 35 12
Рис 3.6: 36 12
Рис 3.7: 37 13
Рис 3.8: 38 14
Рис 3.9.1: 391 15
Рис 3.9.2: 392 15
Рис 3.10.1: 3101 16
Рис 3.10.2: 3102 16
Рис 4.1.1: 411 17
Рис 4.1.2: 412 17
Рис 4.1.3: 413 18
Рис 4.2.1: 421 18
Рис 4.2.2: 422 18

1. Цель работы

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

2. Теоретическое введение

2.1 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.

Следующие комбинации клавиш облегчают работу с Midnight Commander:
• Tab используется для переключениями 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.

Дополнительную информацию о Midnight Commander можно получить по команде man mc и на странице проекта .

2.2. 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 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 размеров 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 (учетверённое 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 неинициированных данных в секции .bss используются директивы resb, resw, resd и другие, которые сообщают ассемблеру, что необходимо зарезервировать заданное количество ячеек памяти.

2.3 Элементы программирования

2.3.1 Описание инструкции mov

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).

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

2.3.2. 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.

2.3.3. 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.

3. Выполнение лабораторной работы

3.1 Откроем Midnight Commander

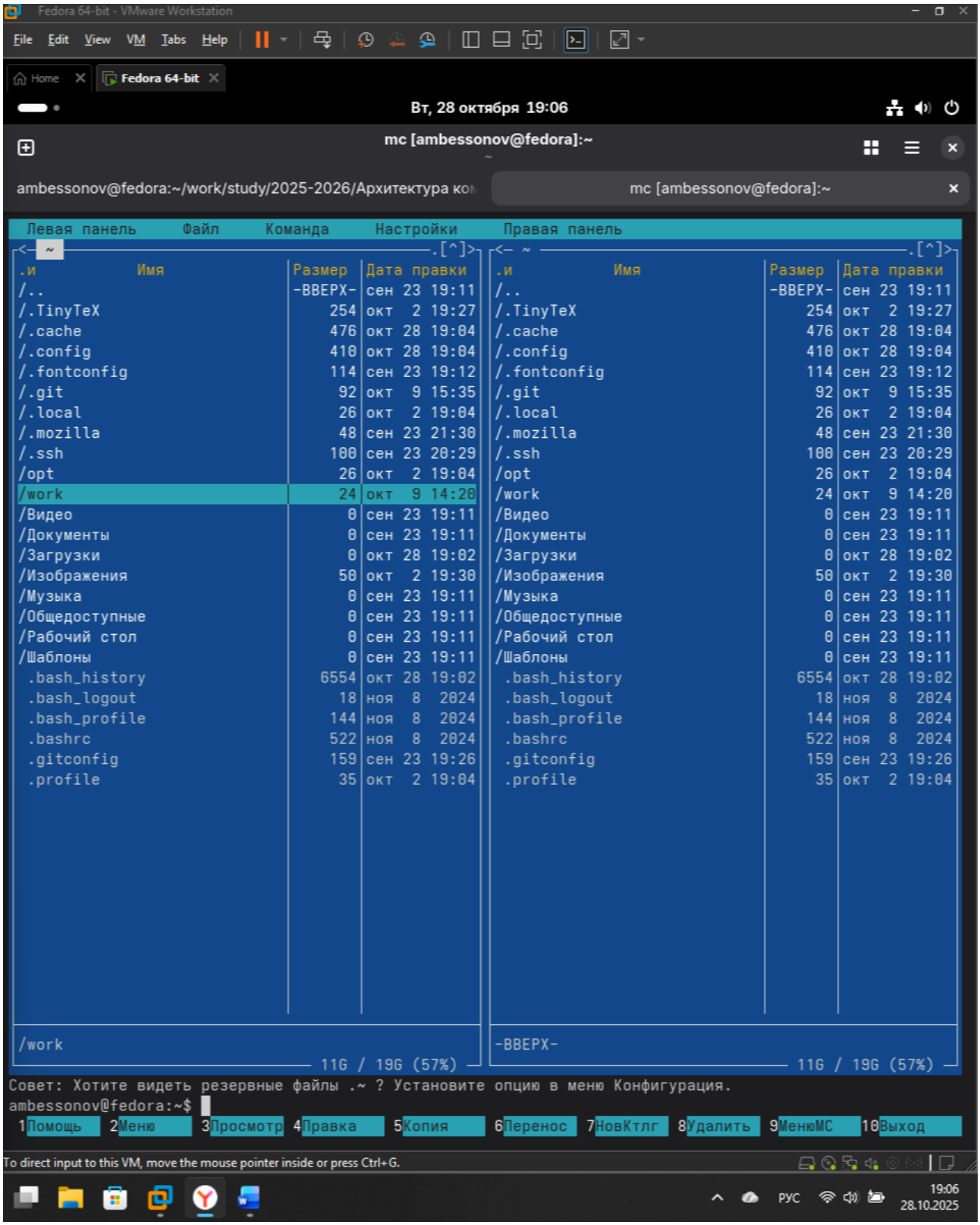


Рис 3.1: 31

3.2 Пользуясь клавишами ↑ , ↓ и Enter перейдем в каталог ~/work/arch-pc созданный при выполнении лабораторной работы №4

С помощью функциональной клавиши F7 создадим папку lab05 и перейдем в созданный каталог.

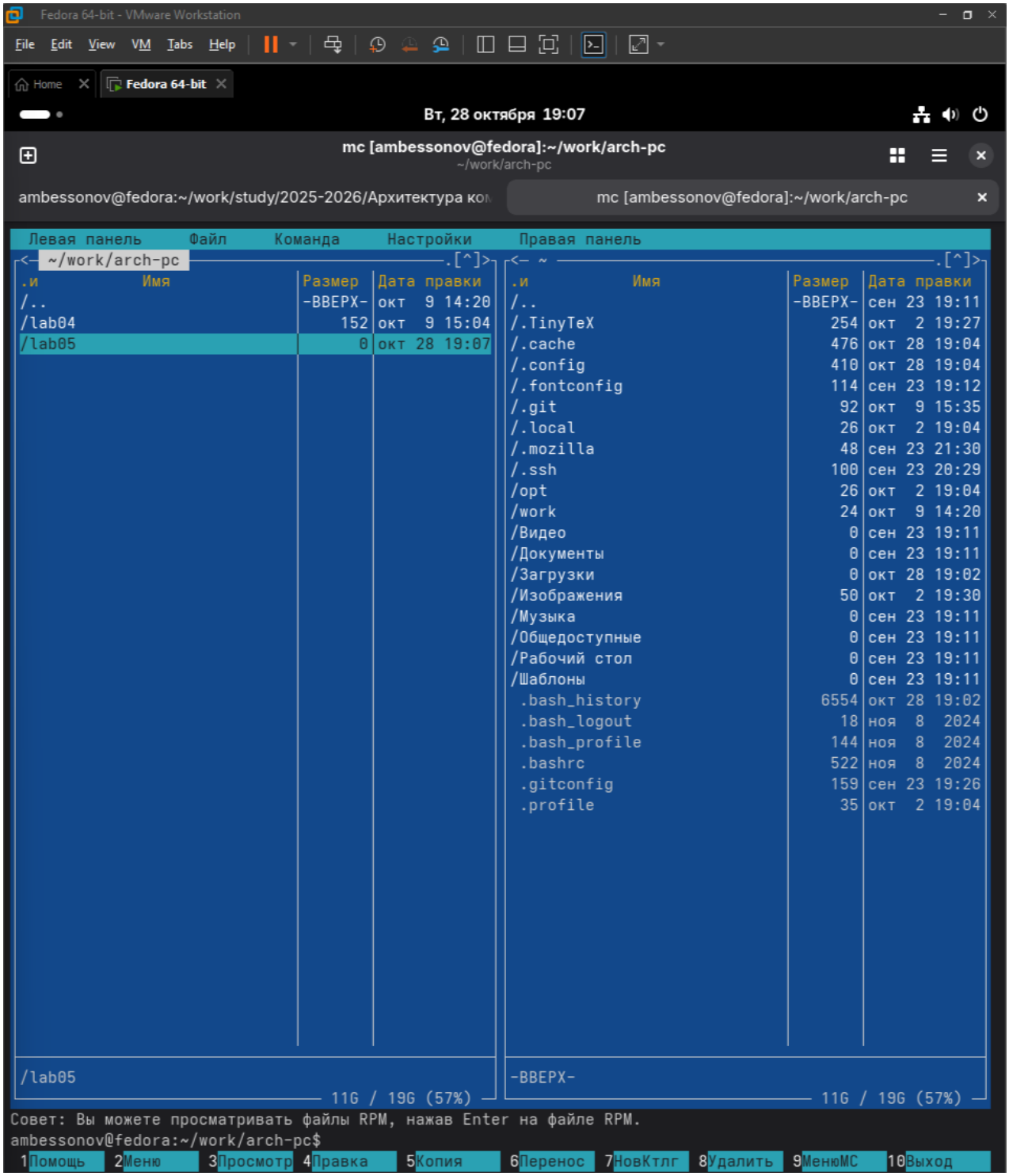


Рис 3.2: 32

3.3 Пользуясь строкой ввода и командой touch создадим файл lab5-1.asm

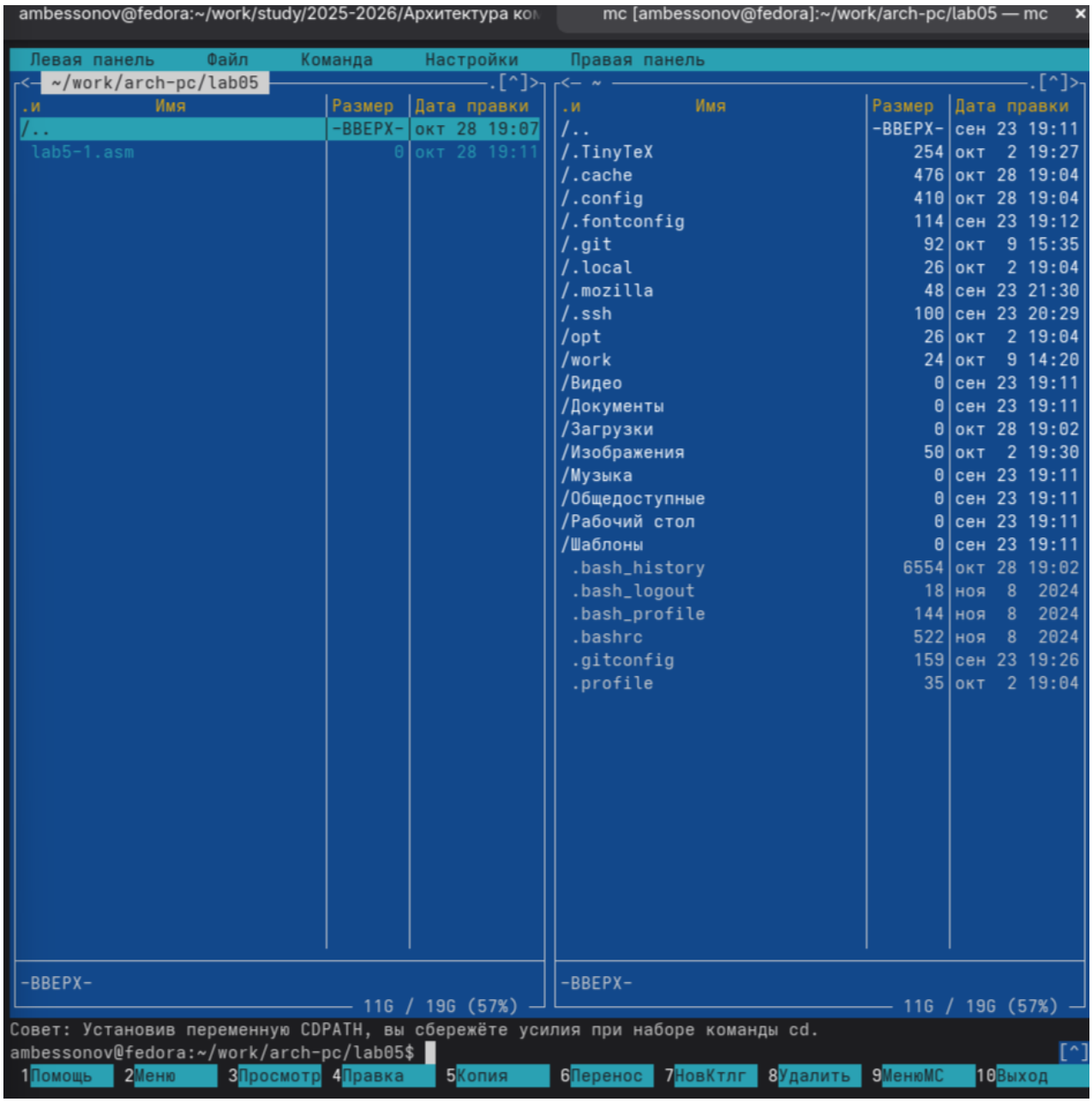


Рис 3.3: 33

3.4 С помощью функциональной клавиши F4 откроем файл lab5-1.asm для редактирования во встроенном редакторе.

Введем текст программы из листинга 5.1, сохраним изменения и закроем файл.

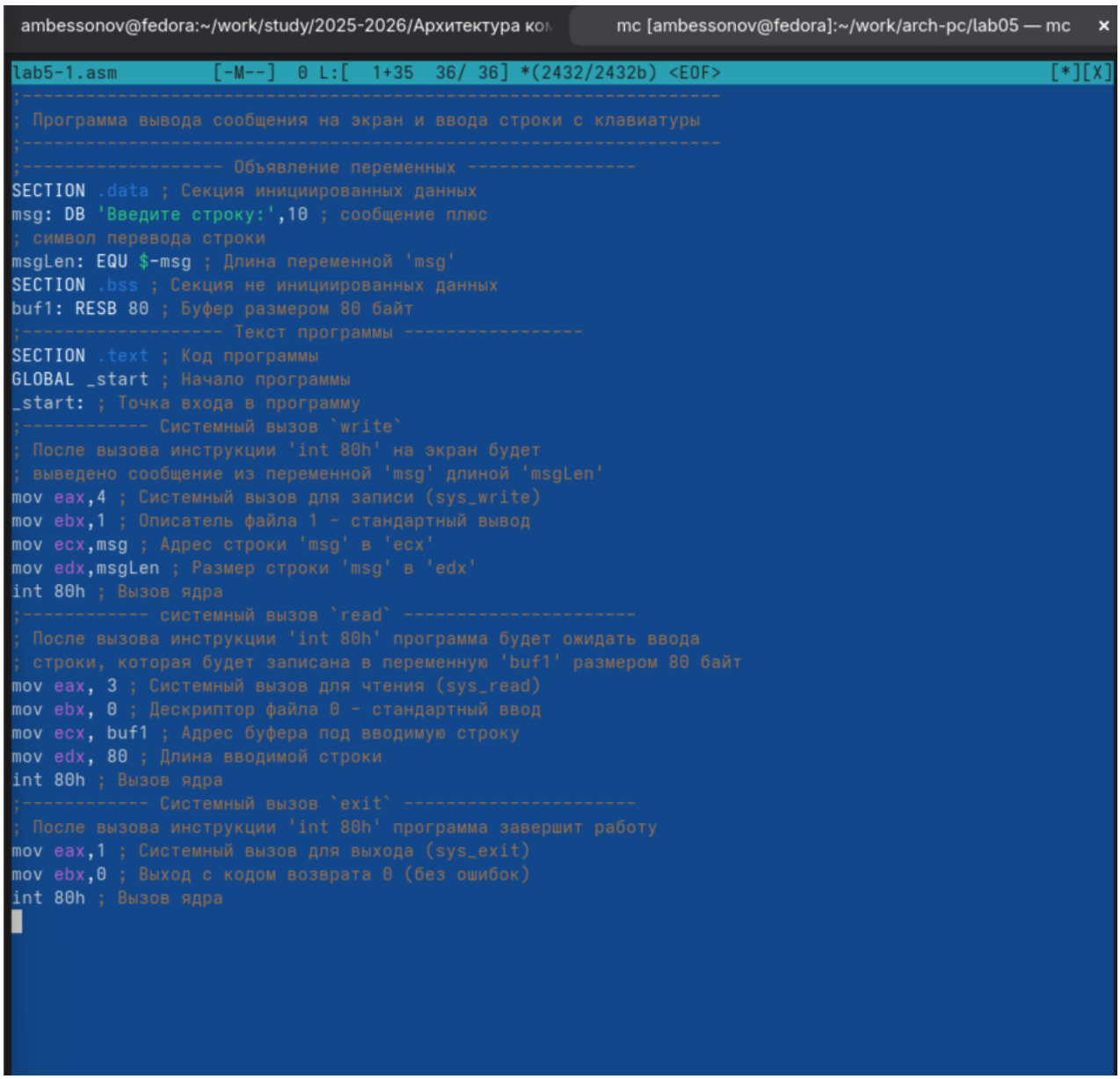


Рис 3.4: 34

3.5 С помощью функциональной клавиши F3 откроем файл lab5-1.asm для просмотра. Убедимся, что файл содержит текст программы.

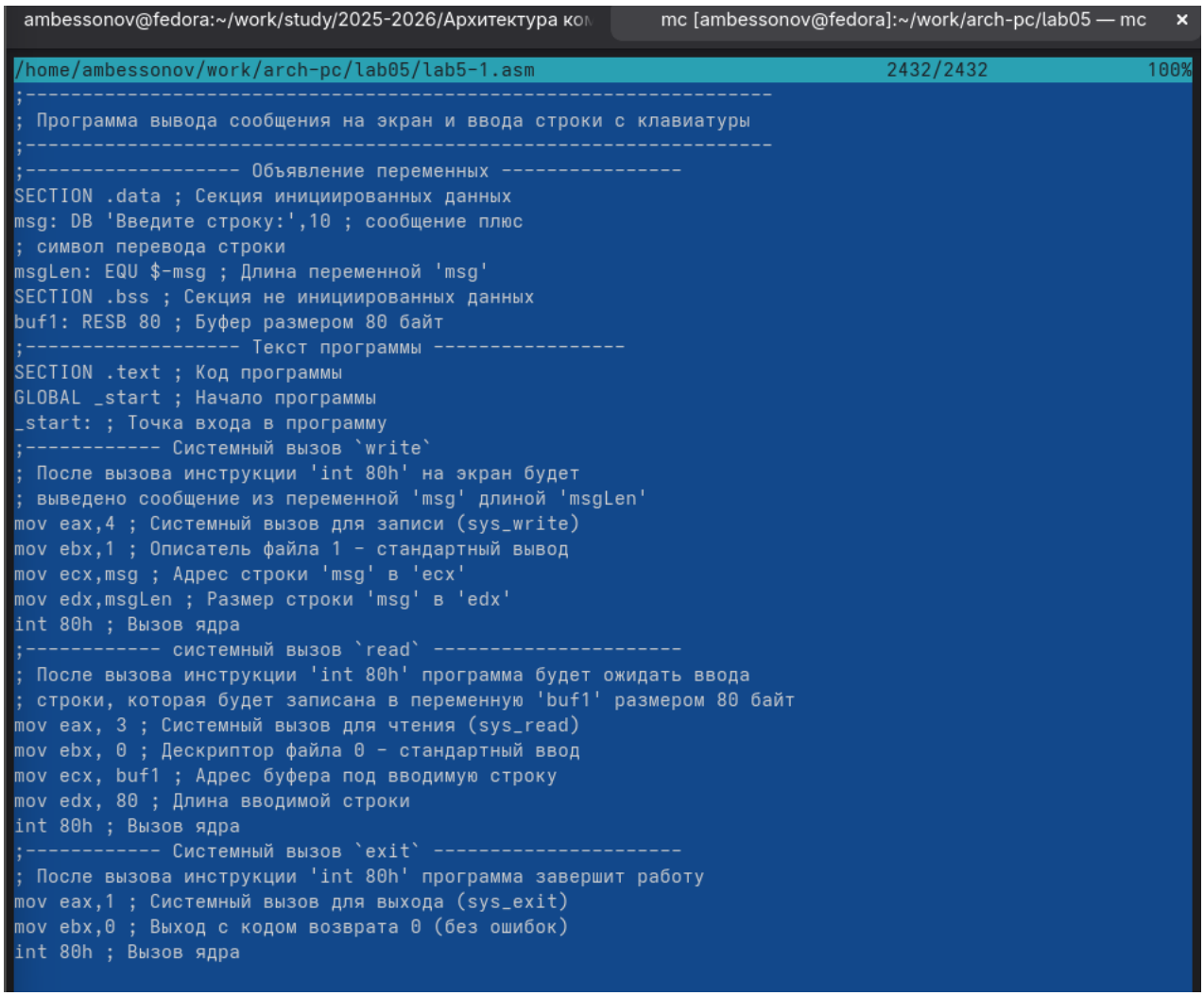


Рис 3.5: 35

3.6 Оттранслируем текст программы lab5-1.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл. Программа выводит строку 'Введите строку:' и ожидает ввода с клавиатуры. На запрос введем ФИО.

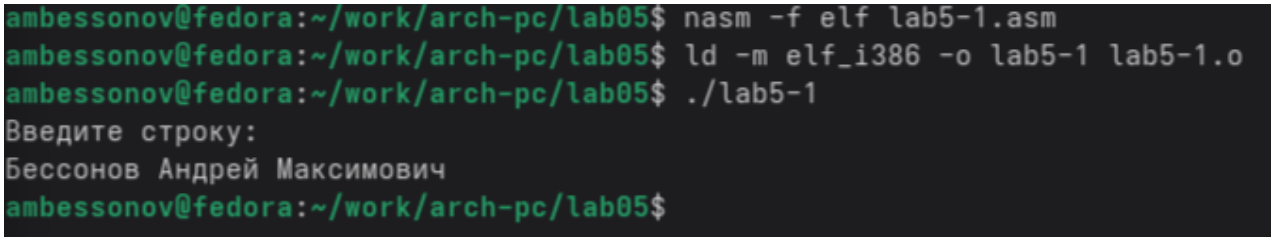


Рис 3.6: 36

3.7 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 (скриншот был выполнен после копирования)

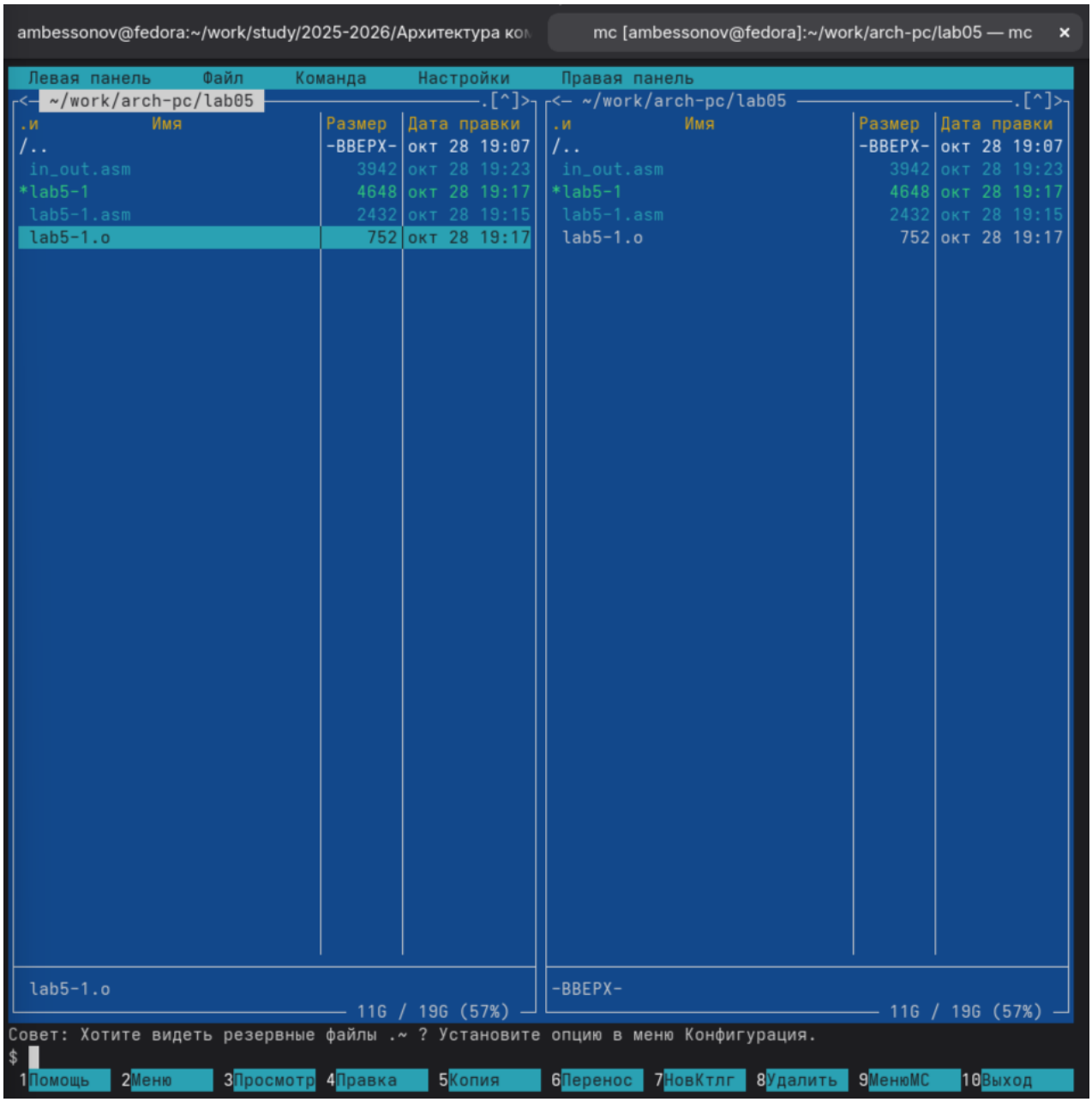


Рис 3.7: 37

3.8 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.

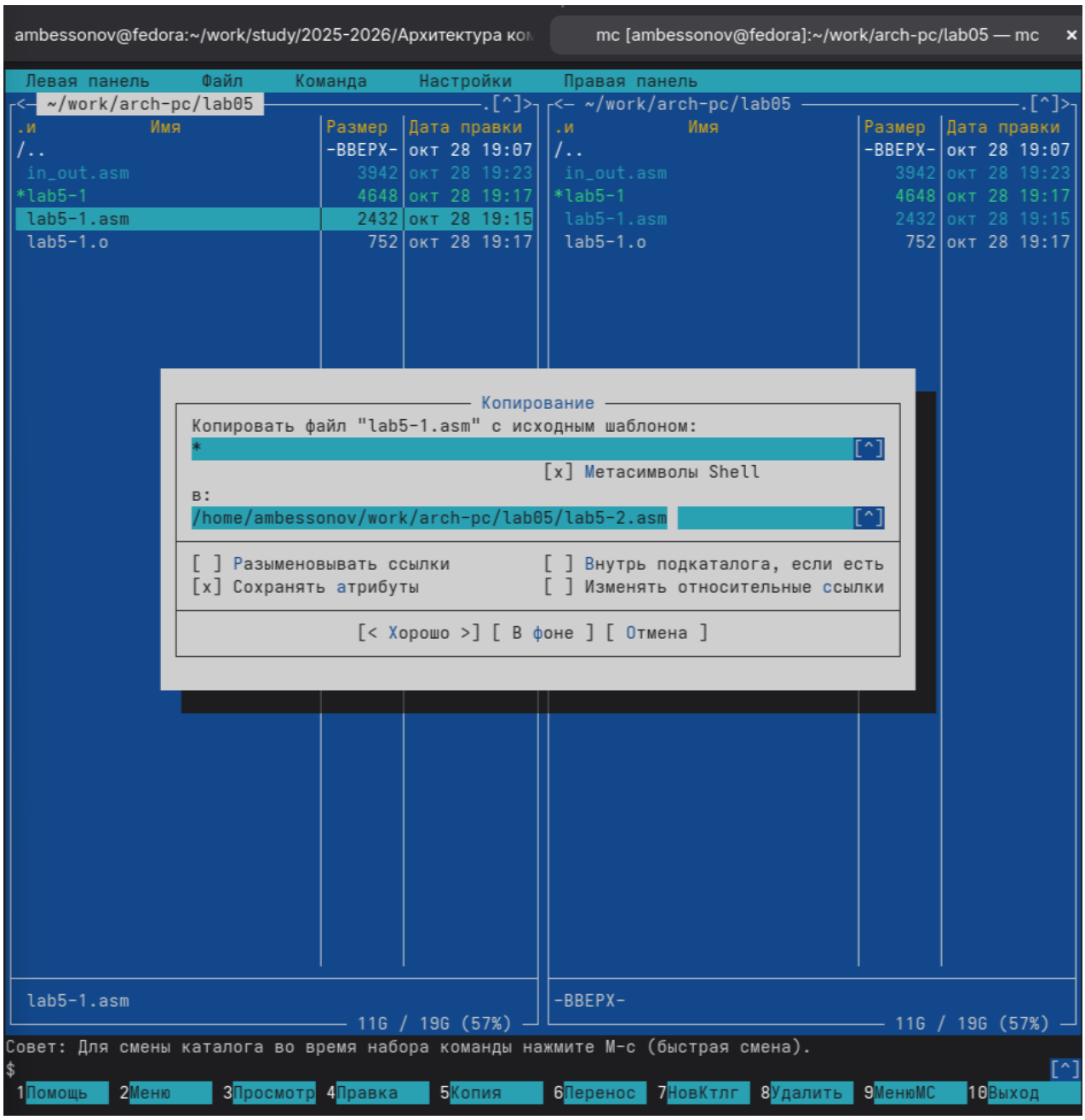


Рис 3.8: 38

3.9 Исправим текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in_out.asm (используем подпрограммы sprintLF, sread и quit) в соответствии с листингом 5.2. Создадим исполняемый файл и проверим его работу.

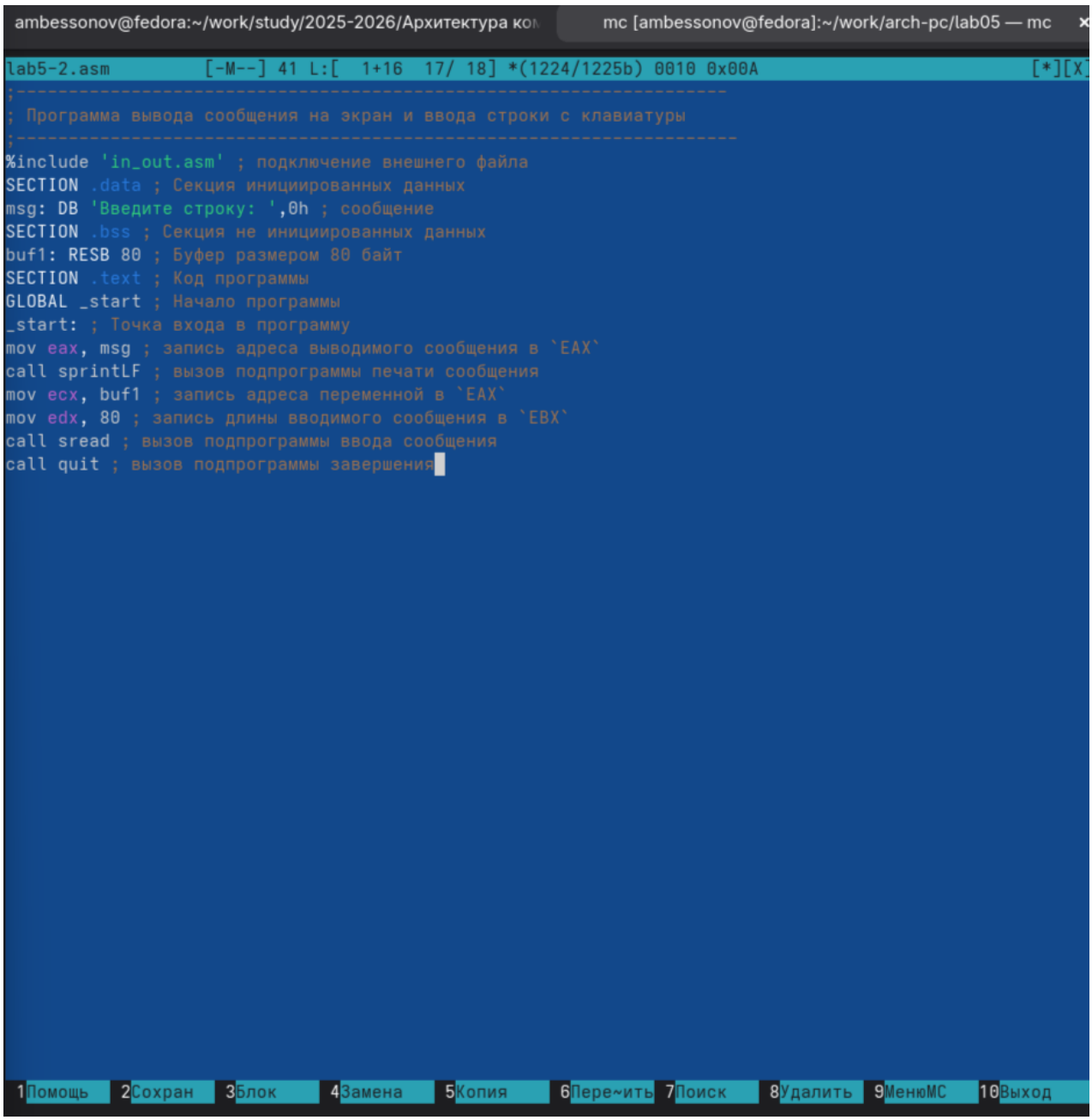


Рис 3.9.1: 391

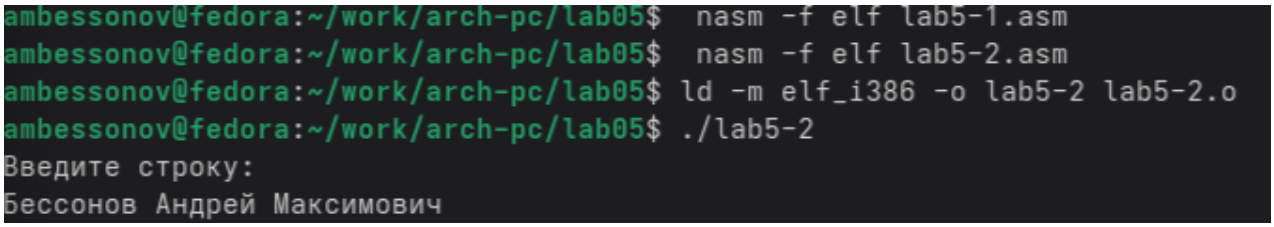


Рис 3.9.2: 392

3.10 В файле lab5-2.asm заменим подпрограмму sprintLF на sprint. Создадим исполняемый файл и проверим его работу.

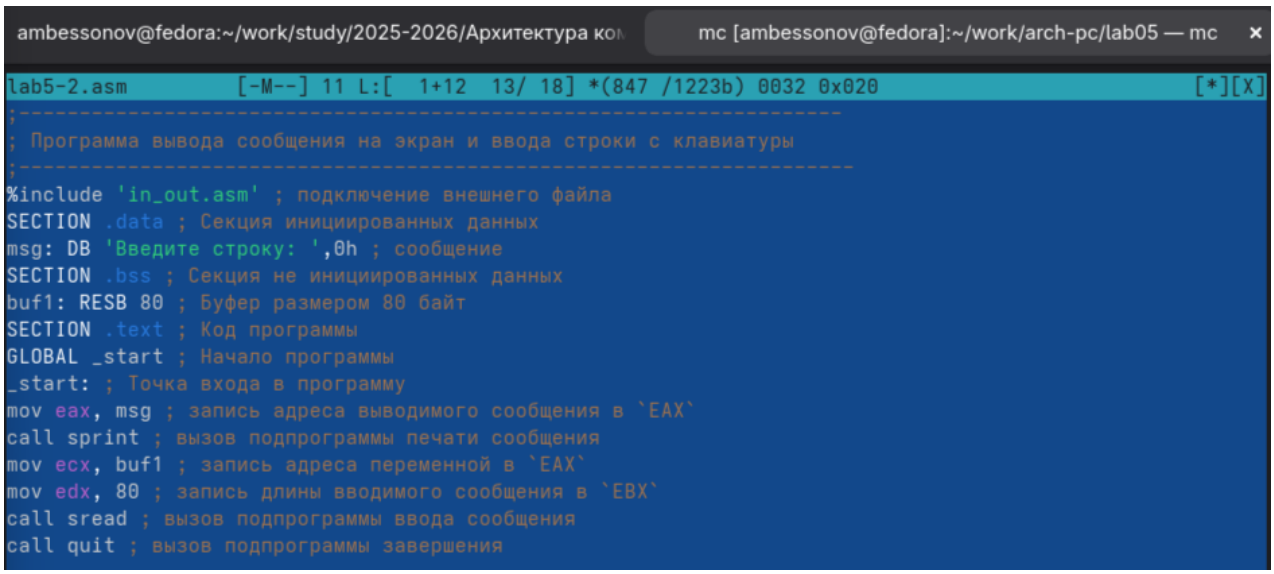


Рис 3.10.1: 3101

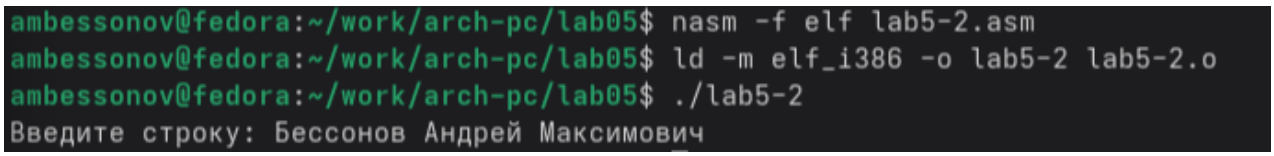


Рис 3.10.2: 3102

Заметим, что в программе с sprintLF она выполняется с новой строки, а в текущей программе с sprint она не переходит на новую строку для ввода.

Вывод: выполнили задания лабораторной работы, научились работать с новыми командами.

4. Выполнение самостоятельной работы

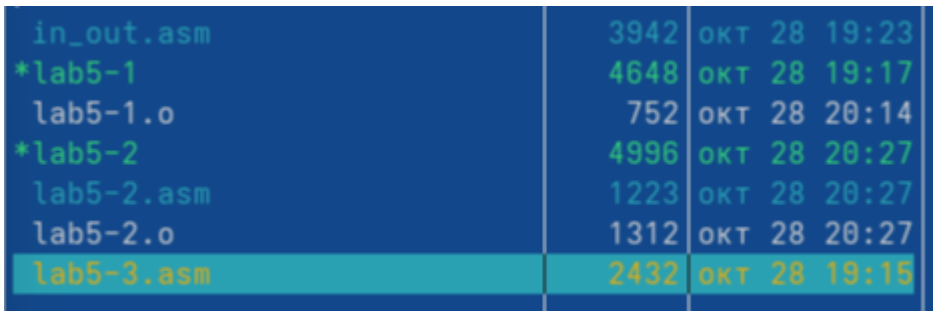
4.1 Создадим копию файла lab5-1.asm. Назовем ее lab5-3.asm. Внесем изменения в программу (без использования внешнего файла in_out.asm), так чтобы она работала по следующему алгоритму:

• вывести приглашение типа “Введите строку:”;

• ввести строку с клавиатуры;

• вывести введённую строку на экран

Получим исполняемый файл и проверим его работу. На приглашение ввести строку введем свою фамилию.



in_out.asm	3942	окт 28 19:23
*lab5-1	4648	окт 28 19:17
lab5-1.o	752	окт 28 20:14
*lab5-2	4996	окт 28 20:27
lab5-2.asm	1223	окт 28 20:27
lab5-2.o	1312	окт 28 20:27
lab5-3.asm	2432	окт 28 19:15

Рис 4.1.1: 411

```
/home/ambessonov/work/arch-pc/lab05/lab5-3.asm
;-----------------------------------------------------------------
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----------------------------------------------------------------
;------------------- Объявление переменных ----------------
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт
;------------------- Текст программы -----------------
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
;------------ Системный вызов `write`
; После вызова инструкции 'int 80h' на экран будет
; 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
```

Рис 4.1.2: 412

```
ambessonov@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-3.asm
ambessonov@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-3 lab5-3.o
ambessonov@fedora:~/work/arch-pc/lab05$ ./lab5-3
Введите строку:
Bessonov
```

Рис 4.1.3: 413

Таким образом, программа при вводе фамилии выводит ее на экран.

4.2 Создадим копию файла lab5-2.asm. Исправим 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 введённую строку на экран.

Создадим исполняемый файл и назовем его lab5-4, проверим его работу.

```
 in_out.asm          3942 окт 28 19:23
*lab5-1              4648 окт 28 19:17
 lab5-1.o             752 окт 28 20:14
*lab5-2              4996 окт 28 20:27
 lab5-2.asm          1223 окт 28 20:27
 lab5-2.o            1312 окт 28 20:27
*lab5-3              4648 окт 28 20:34
 lab5-3.asm          2432 окт 28 19:15
 lab5-3.o             752 окт 28 20:34
 lab5-4.asm          1223 окт 28 20:27
```

Рис 4.2.1: 421

```
ambessonov@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-4.asm
ambessonov@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-4 lab5-4.o
ambessonov@fedora:~/work/arch-pc/lab05$ ./lab5-4
Введите строку: Bessonov
```

Рис 4.2.2: 422

Таким образом, программа выводит на экран введенную строку.

Вывод: выполнили задания самостоятельной работы, отработали навыки полученные в ходе лабораторной работы.

5. Выводы

Приобрели практические навыки работы в Midnight Commander. Освоили инструкций языка ассемблера mov и int.