

Отчёт по лабораторной работе №4

Дисциплина: Архитектура Компьютера

Азарцова Вероника Валерьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Основы работы с Midnight Commander	7
3.2	Структура программы на языке ассемблера NASM	8
3.3	Элементы программирования	8
3.3.1	Инструкция mov	8
3.3.2	Инструкция int	9
3.3.3	Системные вызовы для обеспечения диалога с пользователем	9
4	Выполнение лабораторной работы	10
4.1	Подключение внешнего файла in_out.asm	17
5	Задания для самостоятельной работы	22
6	Выводы	28
	Список литературы	29

Список иллюстраций

4.1	Команда mc	10
4.2	Интерфейс Midnight Commander	10
4.3	Каталог arch-pc	11
4.4	Создание папки lab05	12
4.5	Создание файла lab5-1.asm	12
4.6	Файл lab5-1.asm в папке lab05	13
4.7	Редактирование lab5-1.asm в mcedit	14
4.8	Ввод текста программы в lab5-1.asm	15
4.9	Сохранение lab5-1.asm в mcedit	15
4.10	Содержимое lab5-1.asm в mcedit после сохранения	16
4.11	Транслирование и наличие lab5-1-exe	16
4.12	Компановка и наличие lab5-1-exe	17
4.13	Запуск lab5-1.asm	17
4.14	Файл in_out.asm в каталоге Загрузки	17
4.15	Копирование 1in_out.asm в Midnight Commander	18
4.16	Файл in_out.asm в каталоге lab05	18
4.17	Копирование файла lab5-1.asm	19
4.18	Исправленный текст программы	20
4.19	Запуск файла	20
4.20	Исправленный текст программы с sprint	21
4.21	Запуск файла с sprint	21
5.1	Копирование lab5-1.asm в lab5-3.asm	22
5.2	Измененный текст программы в lab5-3.asm	23
5.3	Создание и запуск исполняемого файла lab5-3-exe	24
5.4	Копирование lab5-2.asm в lab5-4.asm	25
5.5	Измененный текст программы в lab5-4-exe	26
5.6	Создание и запуск исполняемого файла lab5-4-exe	26

Список таблиц

3.1	Описание часто выполняемых операций ms	7
-----	--	---

1 Цель работы

Целью данной лабораторной работы является приобретение практических навыков работы в МС (Midnight Commander), изучение инструкций языка ассемблера `mov` и `int`, написание и запуск программ с системными вызовами для обеспечения диалога с пользователем, подключение и использование в программе внешнего файла.

2 Задание

1. Изучение теоретического введения.
2. Выполнение лабораторной работы.
3. Выполнение заданий для самостоятельной работы.

3 Теоретическое введение

3.1 Основы работы с Midnight Commander

Midnight Commander (mc) — это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системой.

В Midnight Commander используются функциональные клавиши F1 — F10 , к которым привязаны часто выполняемые операции (табл. 3.1) Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание часто выполняемых операций mc

F1-F10	Описание выполняемой операции
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
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

F1-F10	Описание выполняемой операции
--------	-------------------------------

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

3.2 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) — 1 байт; * DW (define word) — 2 байта; * DD (define double word) — 4 байта; * DQ (define quad word) — 8 байт; * DT (define ten bytes) — 10 байт.

3.3 Элементы программирования

3.3.1 Инструкция 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.

3.3.2 Инструкция `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.

3.3.3 Системные вызовы для обеспечения диалога с пользователем

Вывести строку на экран можно используя системный вызов `write` под номером 4, поместив значение 4 в регистр `eax`. Первым аргументом `write` задаётся дескриптор файла. Для вывода на экран в качестве дескриптора файла нужно указать 1 (стандартный вывод).

Вторым аргументом задаётся адрес выводимой строки.

Последним аргументом задаётся максимальная длина выводимой строки.

Для ввода строки с клавиатуры можно использовать аналогичный системный вызов `read` с такими же аргументами, как у вызова `write`, но дескриптором файла 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.

4 Выполнение лабораторной работы

1. Ввожу команду mc. (рис. 4.1).

```
vvazarcova@fedora:~$ mc
```

Рис. 4.1: Команда mc

Открывается Midnight Commander (рис. 4.2).

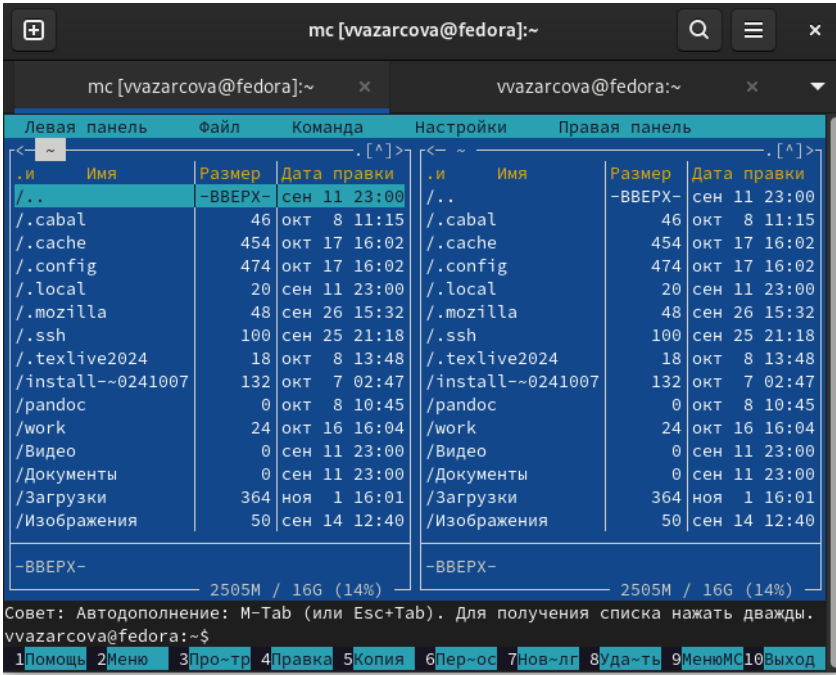


Рис. 4.2: Интерфейс Midnight Commander

2. Используя клавиши “вверх”, “вниз” и “Enter”, перехожу в каталог ~/work/arch-рс, созданный при выполнении лабораторной работы

№4 (рис. 4.3).

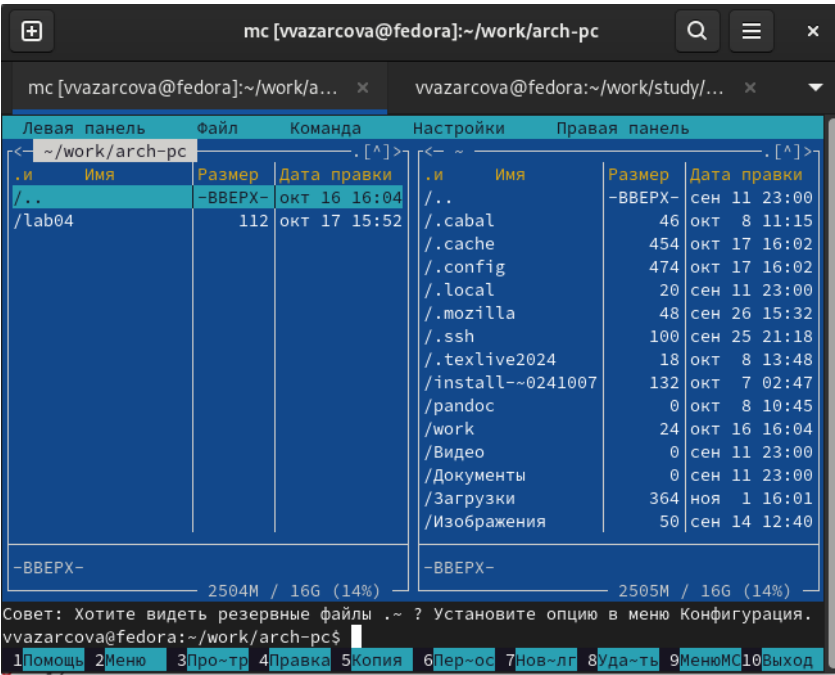


Рис. 4.3: Каталог arch-pc

3. С помощью функциональной клавиши F7 создаю папку lab05 и перехожу в созданный каталог (рис. 4.4).

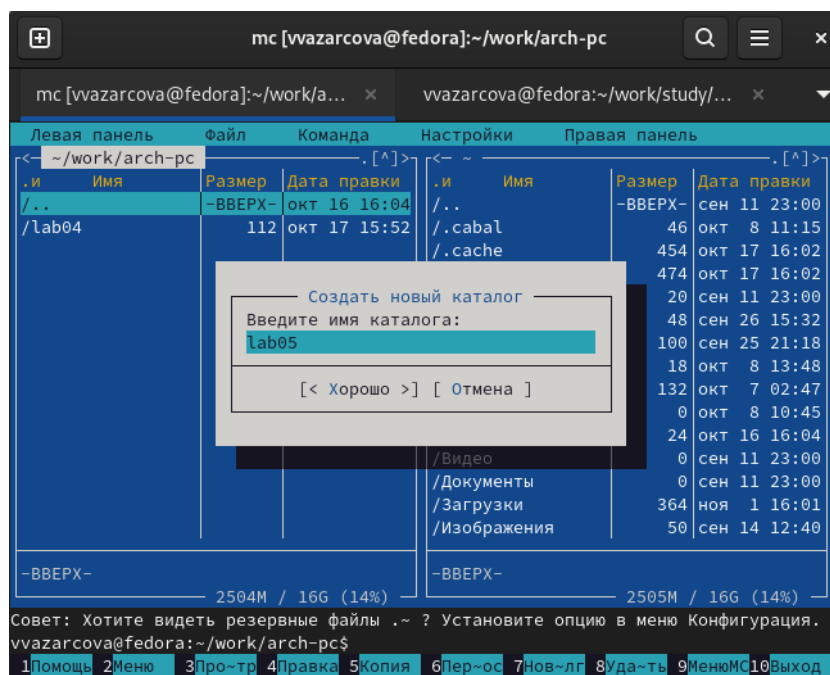


Рис. 4.4: Создание папки lab05

4. Пользуясь строкой ввода, создаю файл lab5-1.asm (рис. 4.5).

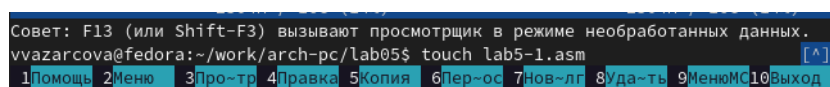


Рис. 4.5: Создание файла lab5-1.asm

Проверяю, что файл создан. (рис. 4.6).

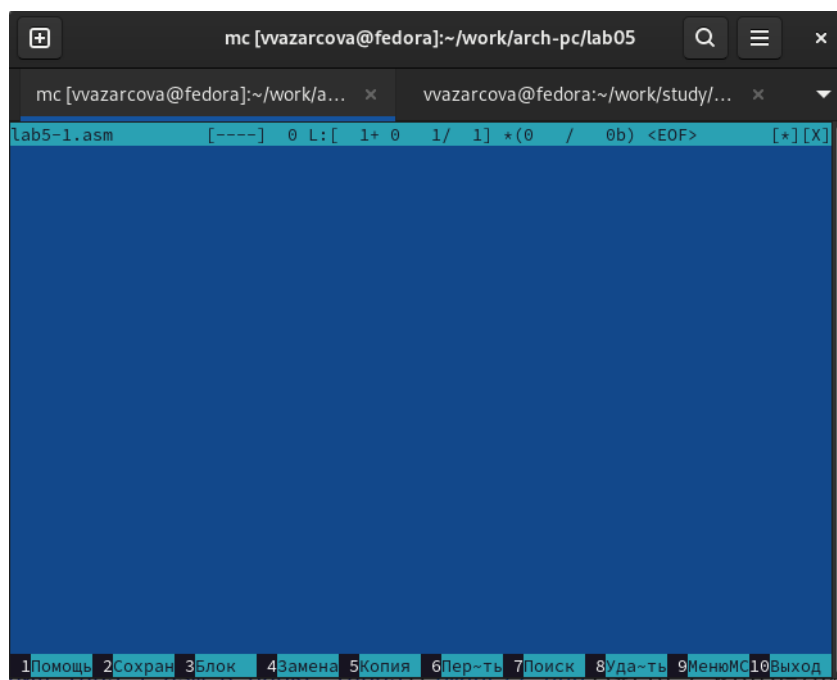


Рис. 4.6: Файл lab5-1.asm в папке lab05

5. С помощью функциональной клавиши “F4” открываю файл lab5-1.asm для редактирование во встроенном редакторе mcedit. (рис. 4.7).

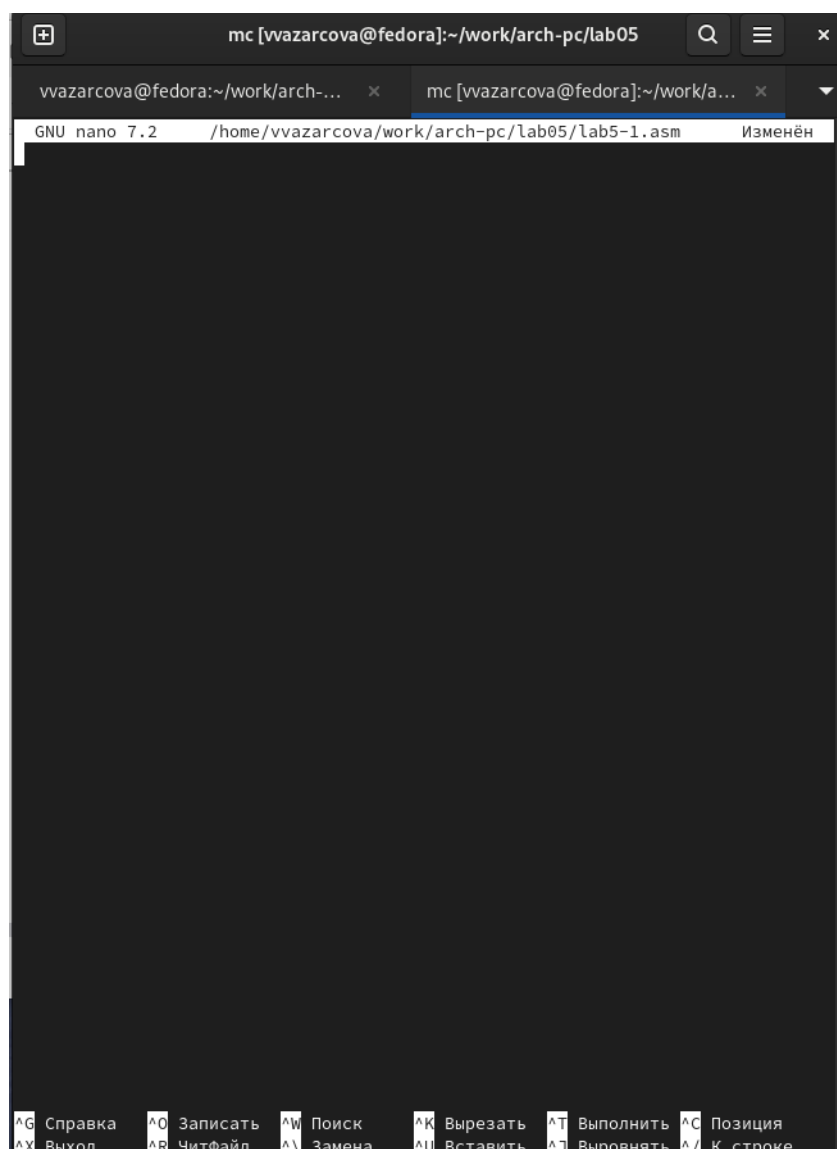
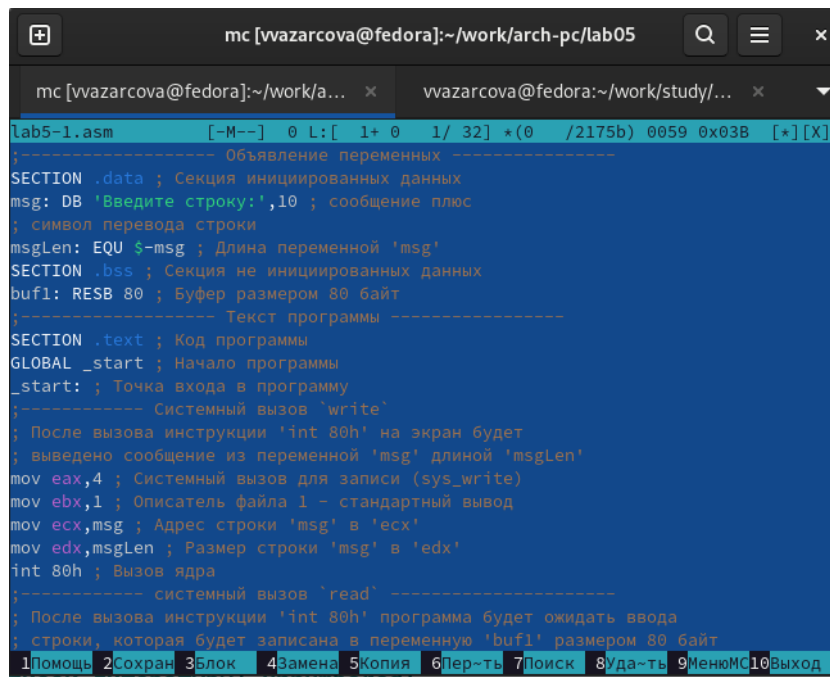


Рис. 4.7: Редактирование lab5-1.asm в mcedit

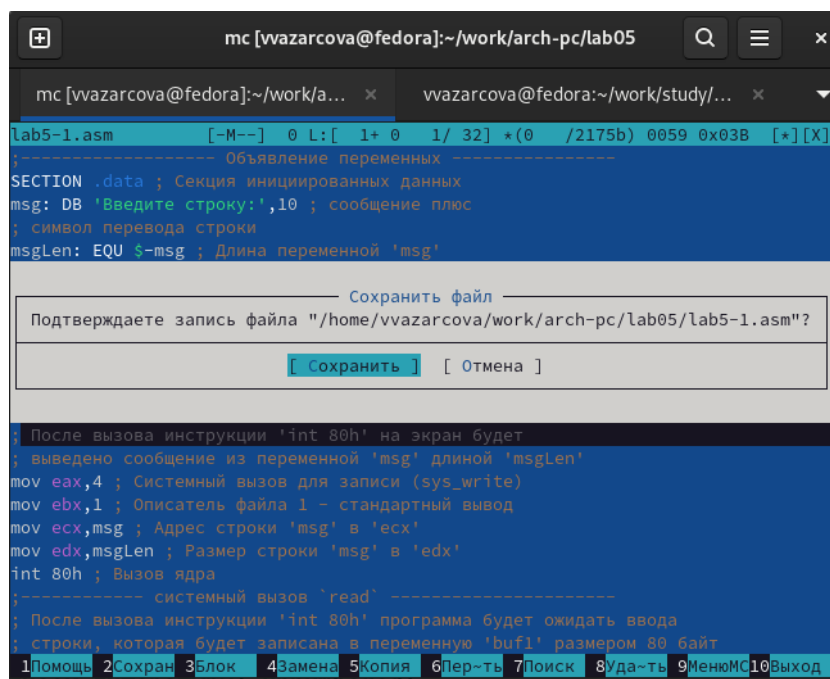
6. Ввожу текст программы. (рис. 4.8).



```
mc [vvazarcova@fedora]:~/work/arch-pc/lab05
lab5-1.asm [-M--] 0 L: [ 1+ 0 1/ 32] *(0 /2175b) 0059 0x03B [*][X]
;----- Объявление переменных -----
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных 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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 4.8: Ввод текста программы в lab5-1.asm

Сохраняю изменения и закрываю файл (рис. 4.9).



```
mc [vvazarcova@fedora]:~/work/arch-pc/lab05
lab5-1.asm [-M--] 0 L: [ 1+ 0 1/ 32] *(0 /2175b) 0059 0x03B [*][X]
;----- Объявление переменных -----
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
;----- 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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Сохранить файл

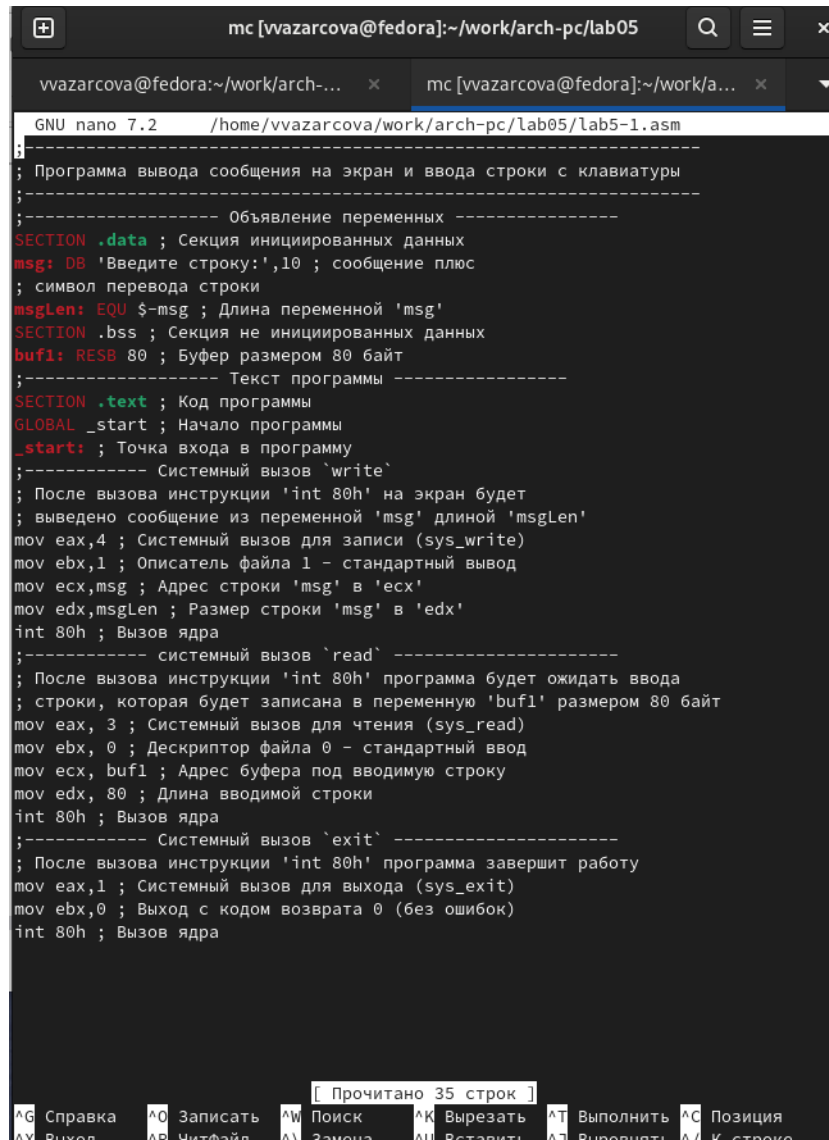
Подтверждаете запись файла "/home/vvazarcova/work/arch-pc/lab05/lab5-1.asm"?

[Сохранить] [Отмена]

Рис. 4.9: Сохранение lab5-1.asm в mcedit

7. С помощью функциональной клавиши “F4” открываю файл lab5-1.asm, что-

бы проверить, что он содержит файл программы (рис. 4.10).

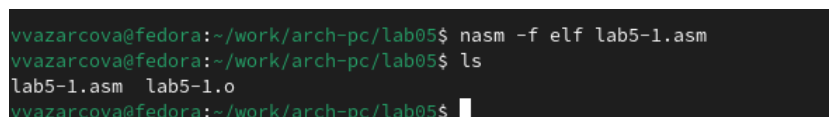


```
GNU nano 7.2 /home/vvazarcova/work/arch-pc/lab05/lab5-1.asm
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных 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

[ Прочитано 35 строк ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^V Вставить ^J Выровнять ^_/ К строке
```

Рис. 4.10: Содержимое lab5-1.asm в mcedit после сохранения

8. Транслирую текст программы lab5-1.asm в объектный файл lab5-1.o и проверяю результат с помощью ls. (рис. 4.11).



```
vvazarcova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
vvazarcova@fedora:~/work/arch-pc/lab05$ ls
lab5-1.asm lab5-1.o
vvazarcova@fedora:~/work/arch-pc/lab05$
```

Рис. 4.11: Транслирование и наличие lab5-1-exe

Выполняю компоновку объектного файла lab5-1.o в исполняемый файл lab5-1_exe и проверяю результат с помощью ls. (рис. 4.12).

```
vvezarcova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-1.o -o lab5-1-exe
vvezarcova@fedora:~/work/arch-pc/lab05$ ls
lab5-1.asm  lab5-1-exe  lab5-1.o
vvezarcova@fedora:~/work/arch-pc/lab05$
```

Рис. 4.12: Компоновка и наличие lab5-1-exe

Запускаю получившийся файл. (рис. 4.13).

```
vvezarcova@fedora:~/work/arch-pc/lab05$ ./lab5-1-exe
Введите строку:
Азарцова Вероника Валерьевна
vvezarcova@fedora:~/work/arch-pc/lab05$
```

Рис. 4.13: Запуск lab5-1.asm

Файл работает корректно: выводит приглашение и запрашивает ввод.

4.1 Подключение внешнего файла in_out.asm

9. Скачиваю файл in_out.asm с курса в ТУИС в каталог Загрузки (рис. 4.14).

```
vvezarcova@fedora:~$ ls Загрузки
in_out.asm          Л02_Азарцова_отчет.pdf
Л01_Азарцова_отчет.pdf  Л03_Азарцова_отчет.pdf
vvezarcova@fedora:~$
```

Рис. 4.14: Файл in_out.asm в каталоге Загрузки

10. В одной панели mc открываю каталог с файлом lab5-1.asm, в другой каталог со скаченным файлом in_out.asm. Копирую файл in_out.asm в каталог с файлом lab5-1.asm с помощью функциональной клавиши F5 (рис. 4.15).

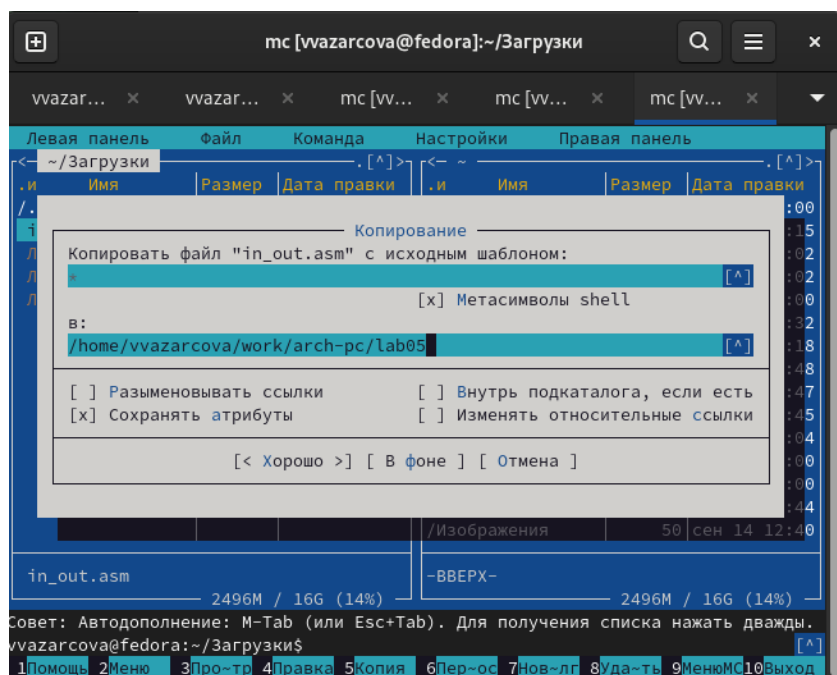


Рис. 4.15: Копирование in_out.asm в Midnight Commander

Проверяю наличие файла в нужном каталоге (рис. 4.16).

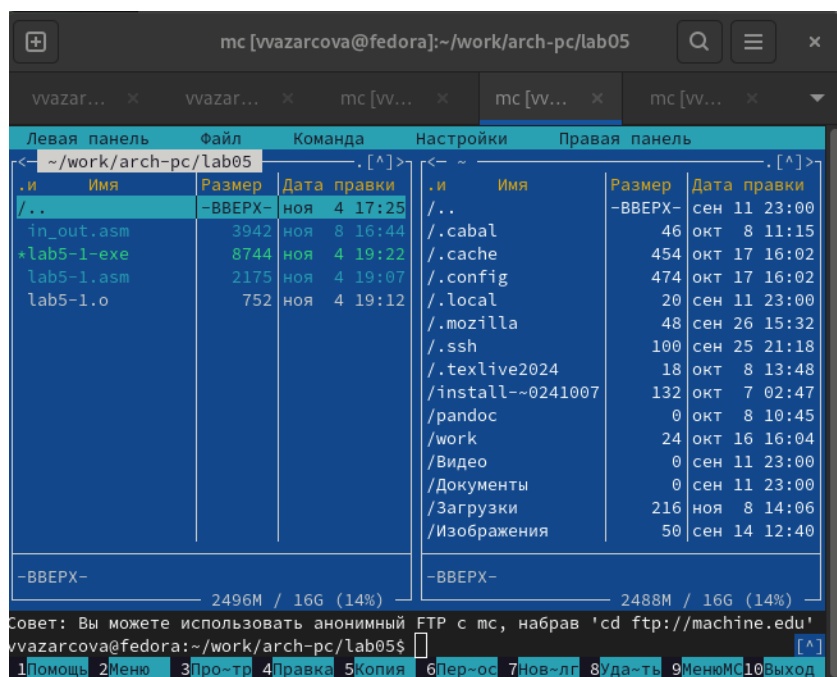


Рис. 4.16: Файл in_out.asm в каталоге lab05

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

именем lab5-2.asm (рис. 4.17).

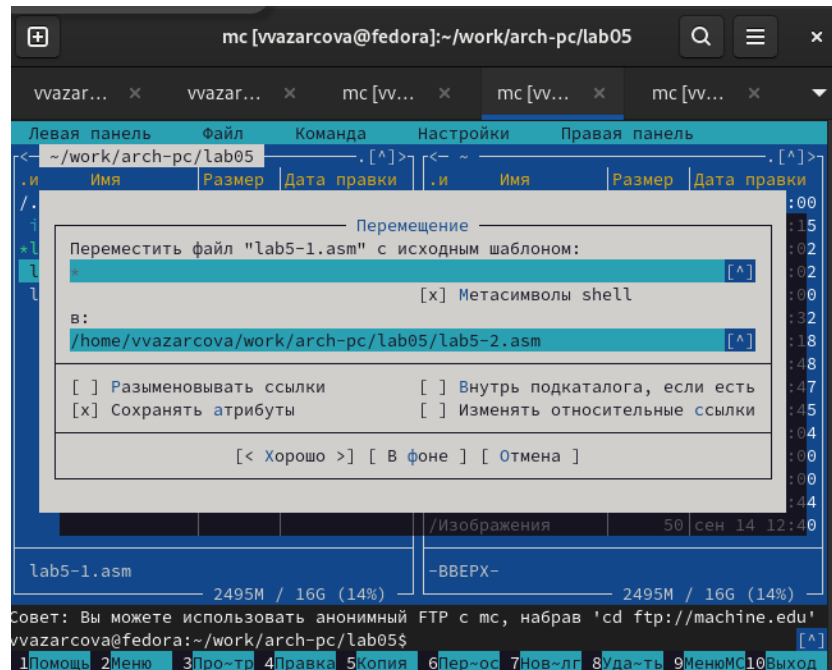


Рис. 4.17: Копирование файла lab5-1.asm

12. Исправляю текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in_out.asm (sprintLF, sread и quit) (рис. 4.18).

```

GNU nano 7.2 /home/vvazarcova/work/arch-pc/lab05/lab5-2.asm
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
mov eax, msg ; запись адреса выводимого сообщения в `EAX`
call sprintf ; вызов подпрограммы печати сообщения
mov ecx, buf1 ; запись адреса переменной в `EAX`
mov edx, 80 ; запись длины вводимого сообщения в `EBX`
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения

```

[Прочитано 17 строк]

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

Рис. 4.18: Исправленный текст программы

Создаю исполняемый файл из lab5-2.asm и проверяю его работу (рис. 4.19).

```

vvazarcova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
vvazarcova@fedora:~/work/arch-pc/lab05$ ls
in_out.asm lab5-1-exe lab5-1.o lab5-2.asm lab5-2.o
vvazarcova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-2.o -o lab5-2-exe
vvazarcova@fedora:~/work/arch-pc/lab05$ ls
in_out.asm lab5-1-exe lab5-1.o lab5-2.asm lab5-2-exe lab5-2.o
vvazarcova@fedora:~/work/arch-pc/lab05$ ./lab5-2-exe
Введите строку:
Азарцова Вероника Валерьевна
vvazarcova@fedora:~/work/arch-pc/lab05$

```

Рис. 4.19: Запуск файла

13. Заменяю подпрограмму sprintf на printf в файле lab5-1.asm (рис. 4.20).

```

mc [wvazarcova@fedora]:~/work/arch-pc/lab05
vvarar... x vvarar... x mc [wv... x mc [wv... x mc [wv... x
GNU nano 7.2 /home/vvazarcova/work/arch-pc/lab05/lab5-2.asm Изменён
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
mov eax, msg ; запись адреса выводимого сообщения в `EAX`
call sprint ; вызов подпрограммы печати сообщения
mov ecx, buf1 ; запись адреса переменной в `EAX`
mov edx, 80 ; запись длины вводимого сообщения в `EBX`
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения

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

```

Рис. 4.20: Исправленный текст программы с sprint

Создаю исполняемый файл и проверяю его работу (рис. 4.21).

```

vvazarcova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm -o lab5-2-sprint.o
vvazarcova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-2-sprint.o -o lab5-2-sprint-exe
vvazarcova@fedora:~/work/arch-pc/lab05$ ls
in_out.asm lab5-1.o lab5-2-exe lab5-2-sprint-exe
lab5-1-exe lab5-2.asm lab5-2.o lab5-2-sprint.o
vvazarcova@fedora:~/work/arch-pc/lab05$ ./lab5-2-sprint-exe
Введите строку: Азарцова Вероника Валерьевна
vvazarcova@fedora:~/work/arch-pc/lab05$

```

Рис. 4.21: Запуск файла с sprint

Разница между программой со `sprintLF` и программой со `sprint` состоит в том, что первая программа, после вывода запроса, запрашивает ввод на новой строке, в то время как вторая запрашивает ввод на той же строке, где выводит запрос.

5 Задания для самостоятельной работы

1. Создаю копию файла lab5-1.asm с помощью mc с названием lab5-3.asm (рис. 5.1).

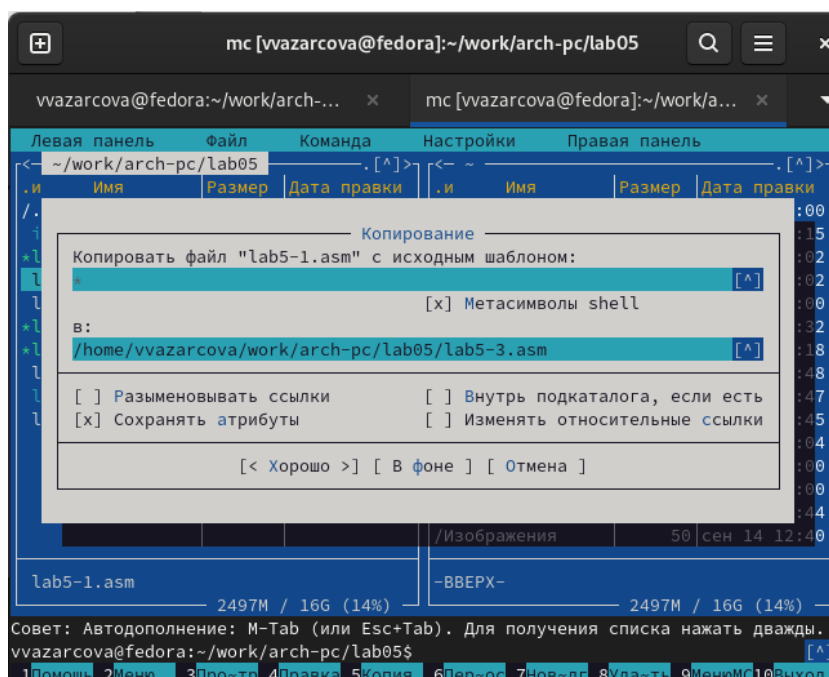


Рис. 5.1: Копирование lab5-1.asm в lab5-3.asm

Требуется изменить программу так, чтобы она работала по следующему алгоритму:

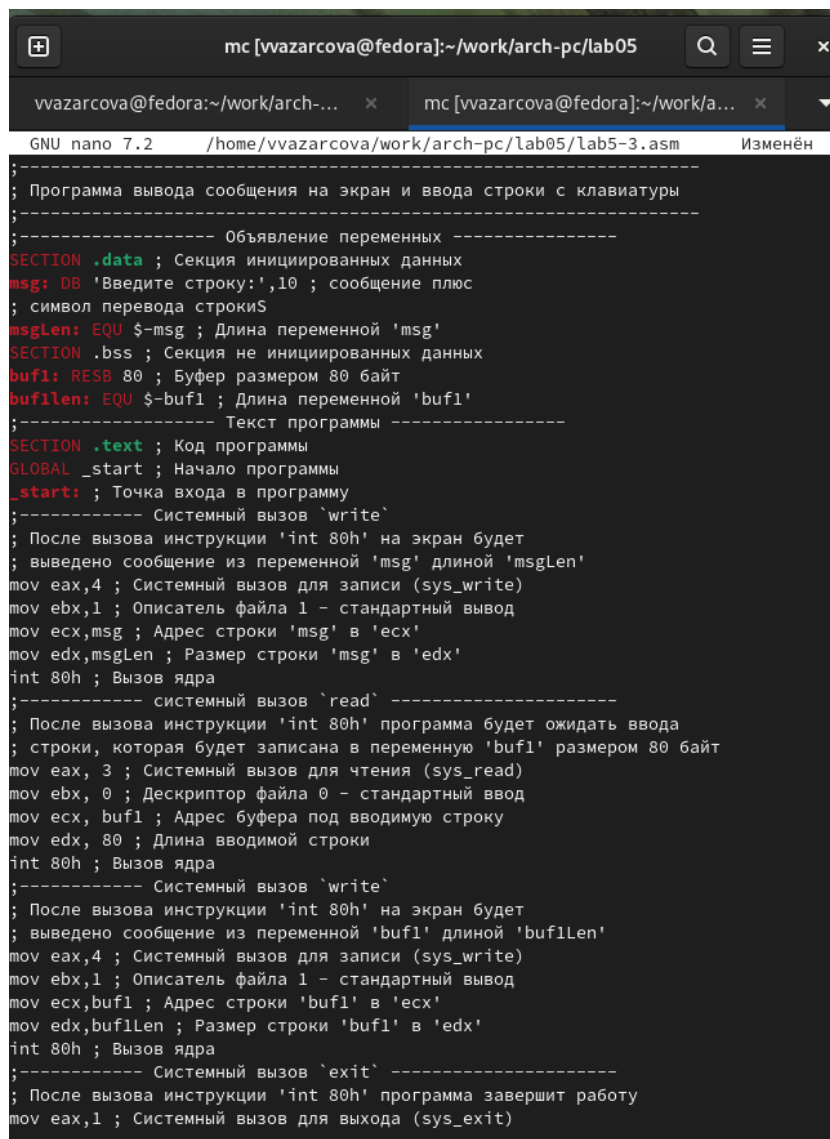
Вывести приглашение типа “Введите строку:”;

Ввести строку с клавиатуры;

Вывести введенную строку на экран.

Для этого добавляю в исходную программу переменную `buf1Len`, в которой будет записана длина строки `buf1`, введенной пользователем.

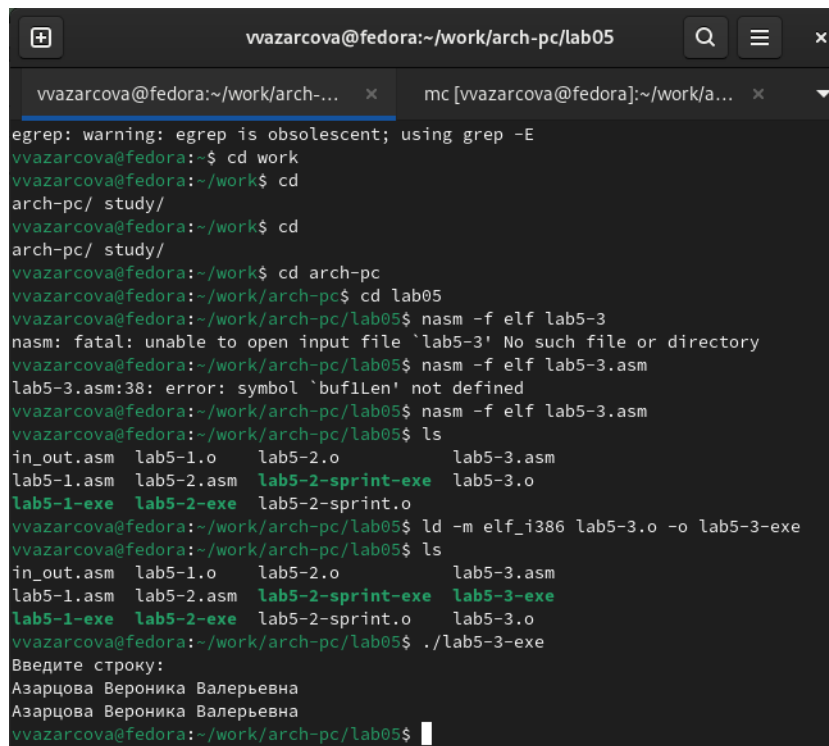
Далее, используя эту переменную, пропишу системный вызов `write`, выводящий на экран содержимое переменной `buf1` (рис. 5.2).



```
GNU nano 7.2 /home/vvazarcova/work/arch-pc/lab05/lab5-3.asm Изменён
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт
buf1len: EQU $-buf1 ; Длина переменной 'buf1'
;----- 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 `write` -----
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'buf1' длиной 'buf1len'
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,buf1 ; Адрес строки 'buf1' в 'ecx'
mov edx,buf1len ; Размер строки 'buf1' в 'edx'
int 80h ; Вызов ядра
;----- Системный вызов `exit` -----
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,1 ; Системный вызов для выхода (sys_exit)
```

Рис. 5.2: Измененный текст программы в `lab5-3.asm`

2. Создаю исполняемый файл `lab5-4-exe` из `lab5-3.asm` и проверяю его работу (рис. 5.3).



```
egrep: warning: egrep is obsolescent; using grep -E
vvazarcova@fedora:~$ cd work
vvazarcova@fedora:~/work$ cd
arch-pc/ study/
vvazarcova@fedora:~/work$ cd
arch-pc/ study/
vvazarcova@fedora:~/work$ cd arch-pc
vvazarcova@fedora:~/work/arch-pc$ cd lab05
vvazarcova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-3
nasm: fatal: unable to open input file `lab5-3' No such file or directory
vvazarcova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-3.asm
lab5-3.asm:38: error: symbol `buf1Len' not defined
vvazarcova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-3.asm
vvazarcova@fedora:~/work/arch-pc/lab05$ ls
in_out.asm  lab5-1.o    lab5-2.o    lab5-3.asm
lab5-1.asm  lab5-2.asm  lab5-2-sprint-exe  lab5-3.o
lab5-1-exe  lab5-2-exe  lab5-2-sprint.o
vvazarcova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-3.o -o lab5-3-exe
vvazarcova@fedora:~/work/arch-pc/lab05$ ls
in_out.asm  lab5-1.o    lab5-2.o    lab5-3.asm
lab5-1.asm  lab5-2.asm  lab5-2-sprint-exe  lab5-3-exe
lab5-1-exe  lab5-2-exe  lab5-2-sprint.o    lab5-3.o
vvazarcova@fedora:~/work/arch-pc/lab05$ ./lab5-3-exe
Введите строку:
Азарцова Вероника Валерьевна
Азарцова Вероника Валерьевна
vvazarcova@fedora:~/work/arch-pc/lab05$
```

Рис. 5.3: Создание и запуск исполняемого файла lab5-3-exe

Программа выводит строку “Введите строку:”, запрашивает ввод с клавиатуры, и затем выводит введенную пользователем строку, т.е. ФИО. Значит, программа работает корректно и согласно алгоритму.

3. Создаю копию файла lab5-2.asm с помощью mc с названием lab5-4.asm (рис. 5.4).

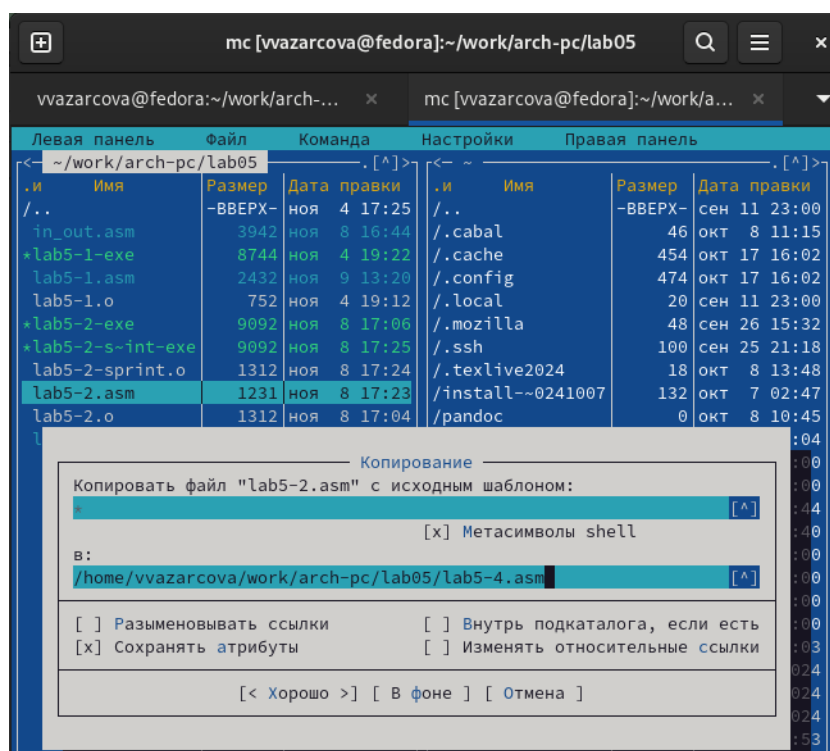


Рис. 5.4: Копирование lab5-2.asm в lab5-4.asm

Требуется изменить программу аналогично пункту номер 1, то есть:

- Вывести приглашение типа “Введите строку:”;
- Ввести строку с клавиатуры;
- Вывести введенную строку на экран.

Для этого, во-первых, меняю в исходной программе `sprint` обратно на `sprintLF` для более понятного ввода и вывода, т.к. поменяла это во время выполнения пункта номер 13 лабораторной работы.

Далее, используя подпрограммы из внешнего файла `in_out.asm`, прописываю запись адреса введенного пользователем сообщения `buf1` в `EAX`, и вызов подпрограммы печати сообщения `sprintLF` (рис. 5.5).

```

GNU nano 7.2 /home/vvazarcova/work/arch-pc/lab05/lab5-4.asm
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
mov eax, msg ; запись адреса выводимого сообщения в `EAX`
call sprintf ; вызов подпрограммы печати сообщения
mov ecx, buf1 ; запись адреса переменной в `EAX`
mov edx, 80 ; запись длины вводимого сообщения в `EBX`
call read ; вызов подпрограммы ввода сообщения
mov eax, buf1 ; запись адреса выводимого сообщения в `EAX`
call sprintf ; вызов подпрограммы печати сообщения
call quit ; вызов подпрограммы завершения

```

Рис. 5.5: Измененный текст программы в lab5-4-ехе

2. Создаю исполняемый lab5-4-ехе файл из lab5-4.asm и проверяю его работу (рис. 5.6).

```

vvazarcova@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-4.asm
vvazarcova@fedora:~/work/arch-pc/lab05$ ls
in_out.asm lab5-1.o lab5-2.o lab5-3.asm lab5-4.asm
lab5-1.asm lab5-2.asm lab5-2-sprint.exe lab5-3.exe lab5-4.o
lab5-1-exe lab5-2-exe lab5-2-sprint.o lab5-3.o
vvazarcova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-4.o -o lab5-4-exe
vvazarcova@fedora:~/work/arch-pc/lab05$ ls
in_out.asm lab5-1.o lab5-2.o lab5-3.asm lab5-4.asm
lab5-1.asm lab5-2.asm lab5-2-sprint.exe lab5-3.exe lab5-4-exe
lab5-1-exe lab5-2-exe lab5-2-sprint.o lab5-3.o lab5-4.o
vvazarcova@fedora:~/work/arch-pc/lab05$ ./lab5-4-exe
Введите строку:
Азарцова Вероника Валерьевна
Азарцова Вероника Валерьевна
vvazarcova@fedora:~/work/arch-pc/lab05$

```

Рис. 5.6: Создание и запуск исполняемого файла lab5-4-ехе

Программа выводит строку “Введите строку:”, запрашивает ввод с клавиатуры, и затем выводит введенную пользователем строку, т.е. ФИО, и при этом использует

внешний файл in_out.asm. Значит, программа работает корректно и согласно алгоритму.

6 Выводы

Подводя итоги данной лабораторной работы, я научилась пользоваться Midnight Commander и успешно написала и запустила несколько программ с системными вызовами для обеспечения диалога с пользователем, используя в нескольких из них внешний файл.

Список литературы