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: Ибрагимов Гаджимурад Шамильевич

Группа: НКАбд-02-25

МОСКВА

2025 г.

ОГЛАВЛЕНИЕ

Цель работы

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

Теоретическое введение

Midnight Commander (mc)

Midnight Commander (mc) — это файловый менеджер для консоли, который упрощает работу с файлами. Он запускается командой mc и представляет собой две панели, между которыми можно копировать и перемещать файлы.

Основные клавиши, которые я использовал в работе:

F4 (Edit)	Открывает файл в текстовом редакторе.
F5 (Copy)	Копирует выделенный файл в каталог на другой панели.
F6 (Move/Rename)	Перемещает или переименовывает файл.
F7 (Mkdir)	Создает новый каталог.
F10 (Quit)	Выходит из mc.
Tab	Переключение между левой и правой панелями.
Ctrl + O	Скрывает/показывает панели mc, позволяя работать с командной строкой.

Таблица 1. Основные клавиши

Структура программы на NASM

Программа на ассемблере NASM, которую мы писали, состоит из нескольких секций.

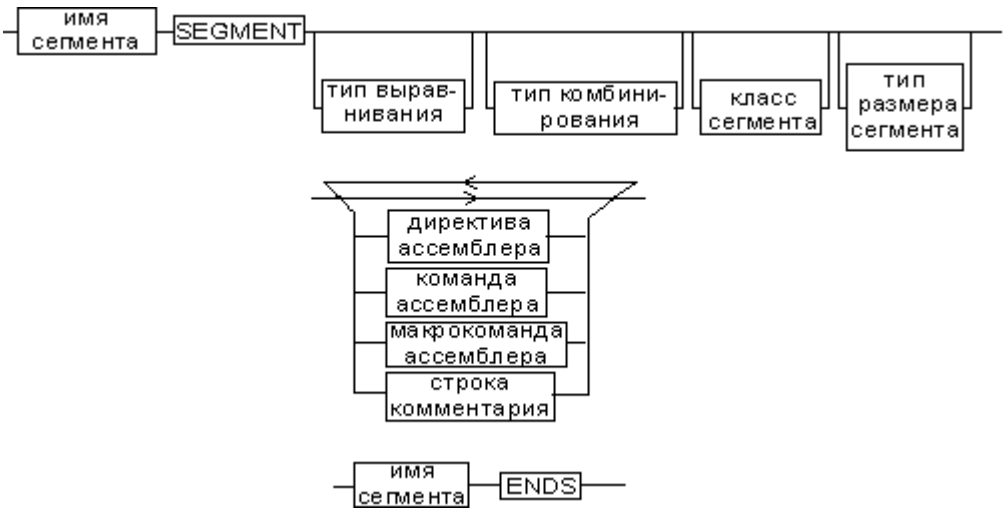


Рисунок 1. Синтаксис описания сегмента

Выполнение программы начинается с метки _start, которую нужно объявить глобальной с помощью GLOBAL _start.

Основные инструкции и системные вызовы

В работе я освоил две главные инструкции:

1. **mov (move):** Инструкция для копирования данных. Она имеет формат mov <приёмник>, <источник>. Например, mov eax, 4 копирует число 4 в регистр eax.
2. **int 80h (interrupt):** Инструкция, которая вызывает ядро Linux для выполнения системной функции (системного вызова).

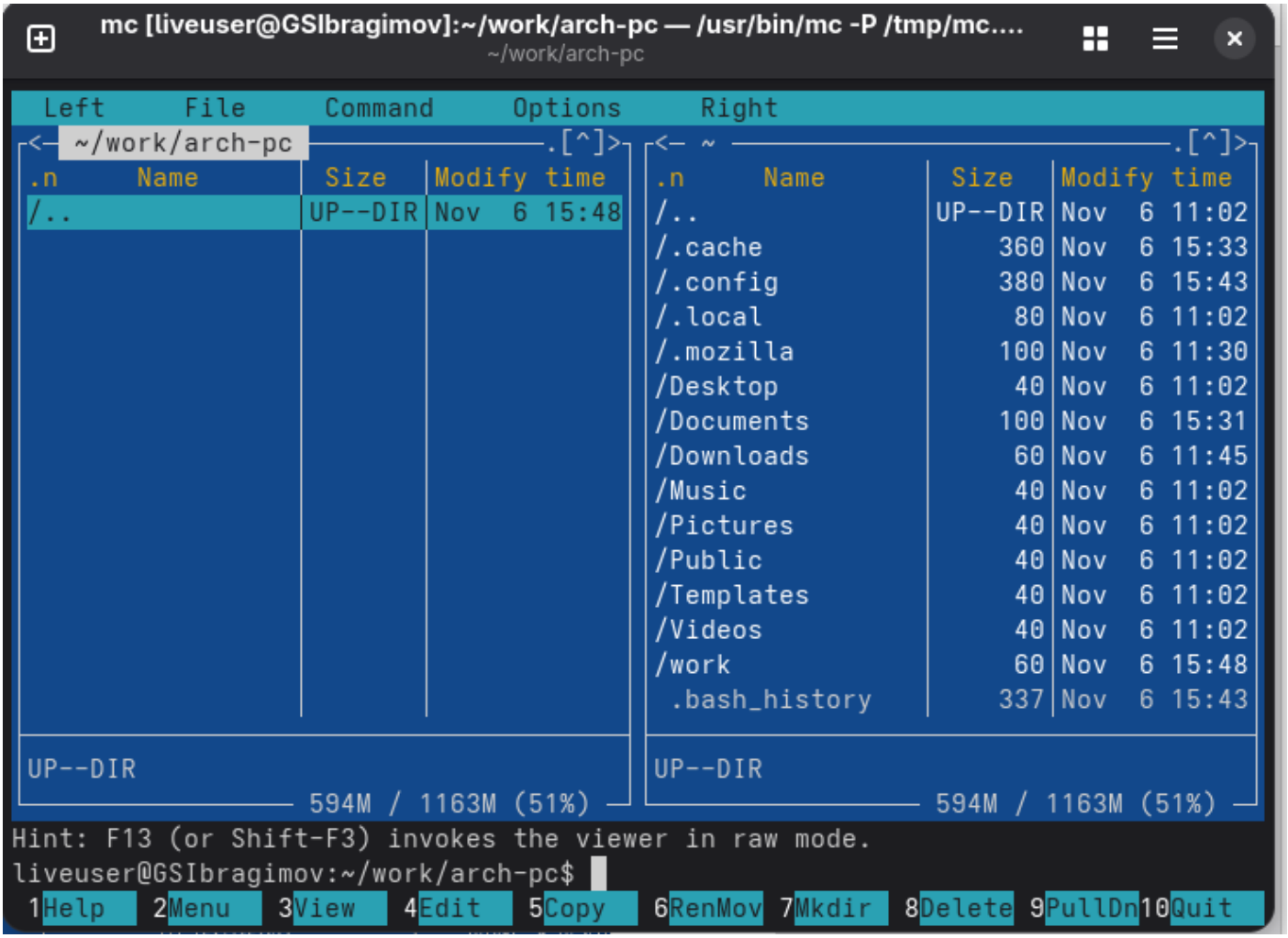
Чтобы ядро поняло, какую функцию мы хотим, мы перед int 80h должны поместить номер функции в регистр eax. Параметры для этой функции кладутся в регистры ebx, ecx, edx.

Описание лабораторной работы

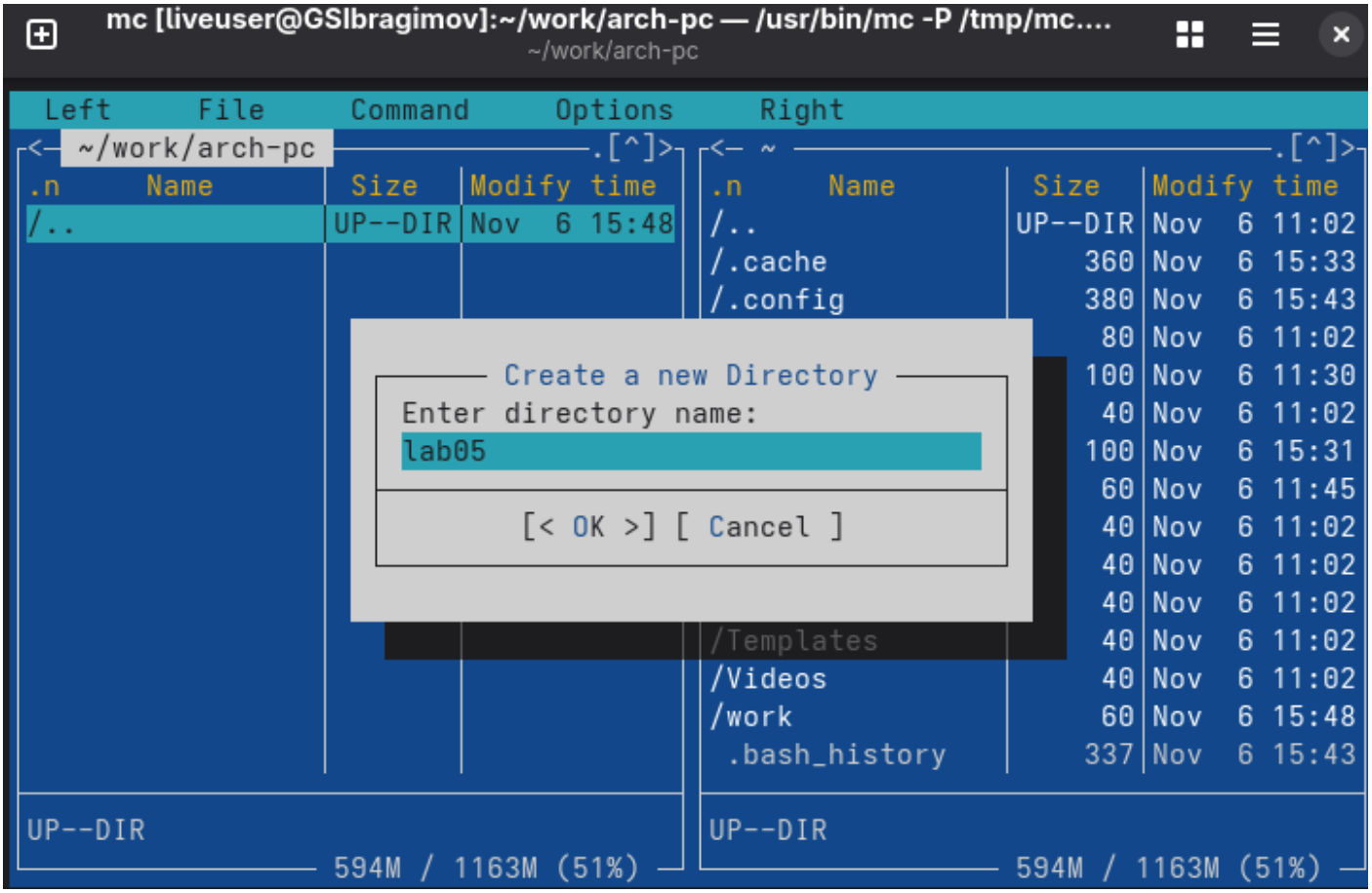
Задание 1. Работа в Midnight Commander и создание первой программы

Необходимо было запустить Midnight Commander, создать в нем рабочий каталог lab05, создать файл lab5-1.asm, отредактировать его, вписав программу из Листинга 5.1, а затем скомпилировать и запустить ее.

Я установил на локальное устройство данный набор пакетов и тперь мне стал доступен Midnight Commander, через консольную команду mc

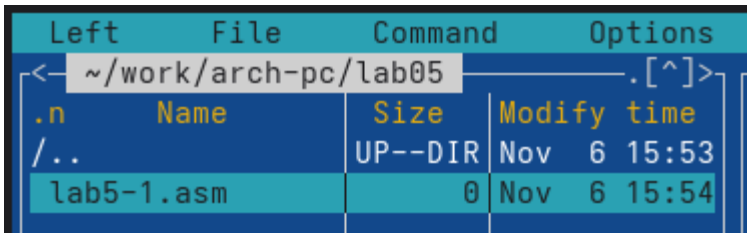


Используя навигацию, я перешел в каталог ~/work/arch-pc, который создал в прошлой лабораторной



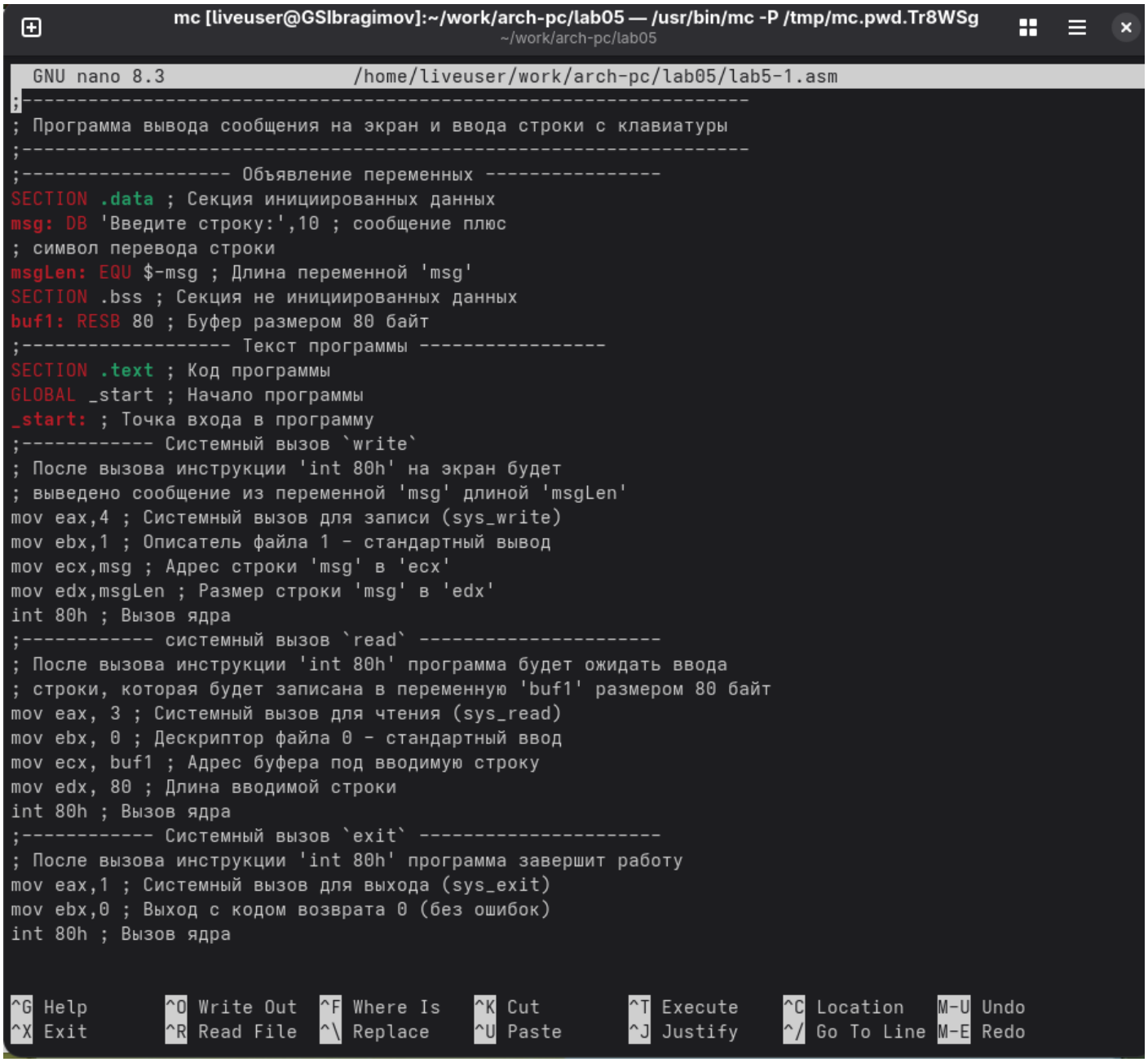
и нажал клавишу F7 для создания нового каталога и ввел имя lab05.

Зашел в созданный каталог lab05 и в командной строке под панелями ввел touch lab5-1.asm и нажал Enter. Файл появился в панели.

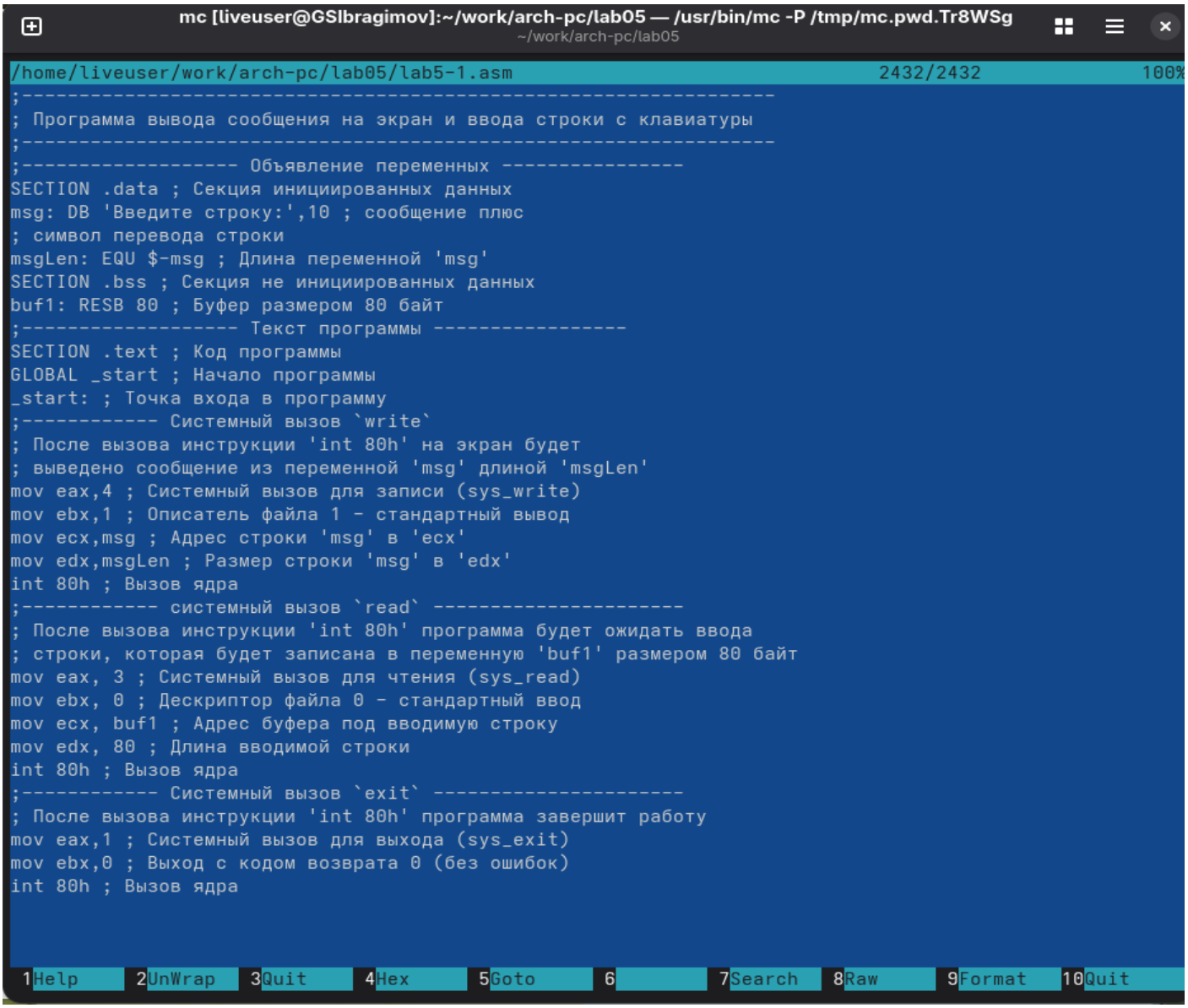


Далее я вбил команды для работы программы следующим образом

я нажал F4, чтобы открыть его во встроенном редакторе



Я сохранил файл (клавиша F2) и вышел из редактора (F10), После сохранения я выполнил трансляцию, компоновку и запуск программы.




```
liveuser@GSIbragimov:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
liveuser@GSIbragimov:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1.o
ld: no input files
liveuser@GSIbragimov:~/work/arch-pc/lab05$ ls
lab5-1.asm  lab5-1.o
liveuser@GSIbragimov:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1.o
ld: no input files
liveuser@GSIbragimov:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1.o
ld: no input files
liveuser@GSIbragimov:~/work/arch-pc/lab05$ ./lab5-1
bash: ./lab5-1: No such file or directory
liveuser@GSIbragimov:~/work/arch-pc/lab05$ ld
ld: no input files
liveuser@GSIbragimov:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
liveuser@GSIbragimov:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Ибрагимов Гаджимурад Шамильевич
liveuser@GSIbragimov:~/work/arch-pc/lab05$
```

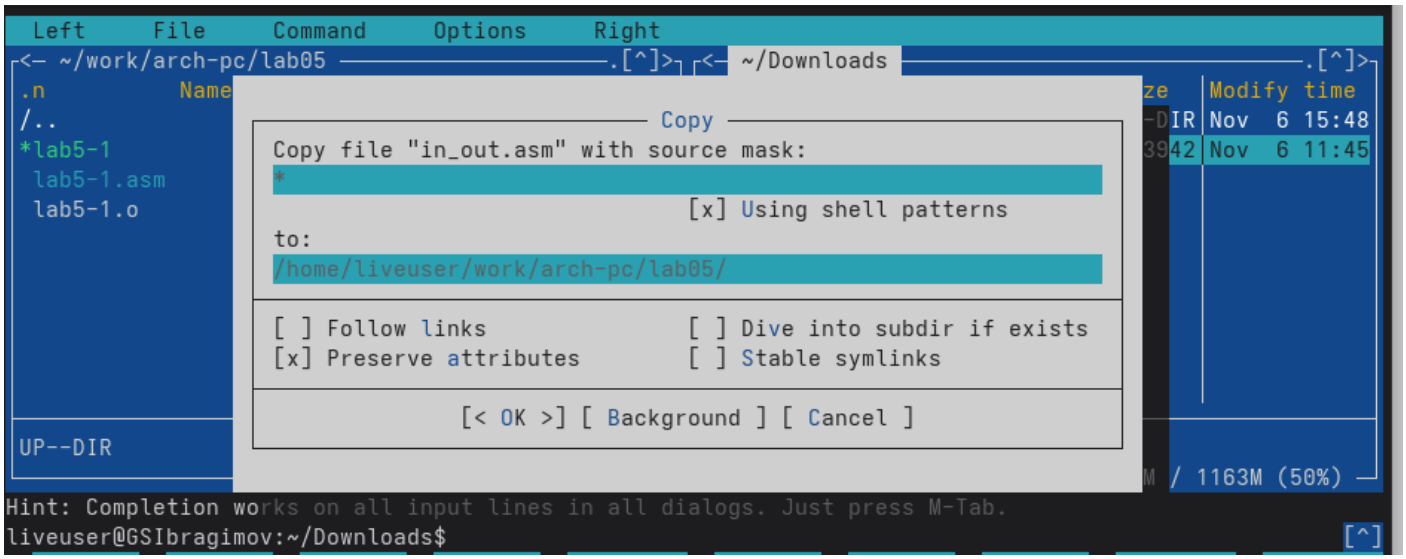
Программа вывела сообщение "Введите строку:". Я ввел свои ФИО и нажал Enter. Программа корректно завершила работу.

На этом этапе я научился базовым операциям в mc: навигации, созданию каталогов (F7), созданию файлов (touch), редактированию (F4) и просмотру (F3). Я также собрал и запустил свою первую программу на NASM, используя системные вызовы write (для вывода), read (для ввода) и exit (для завершения).

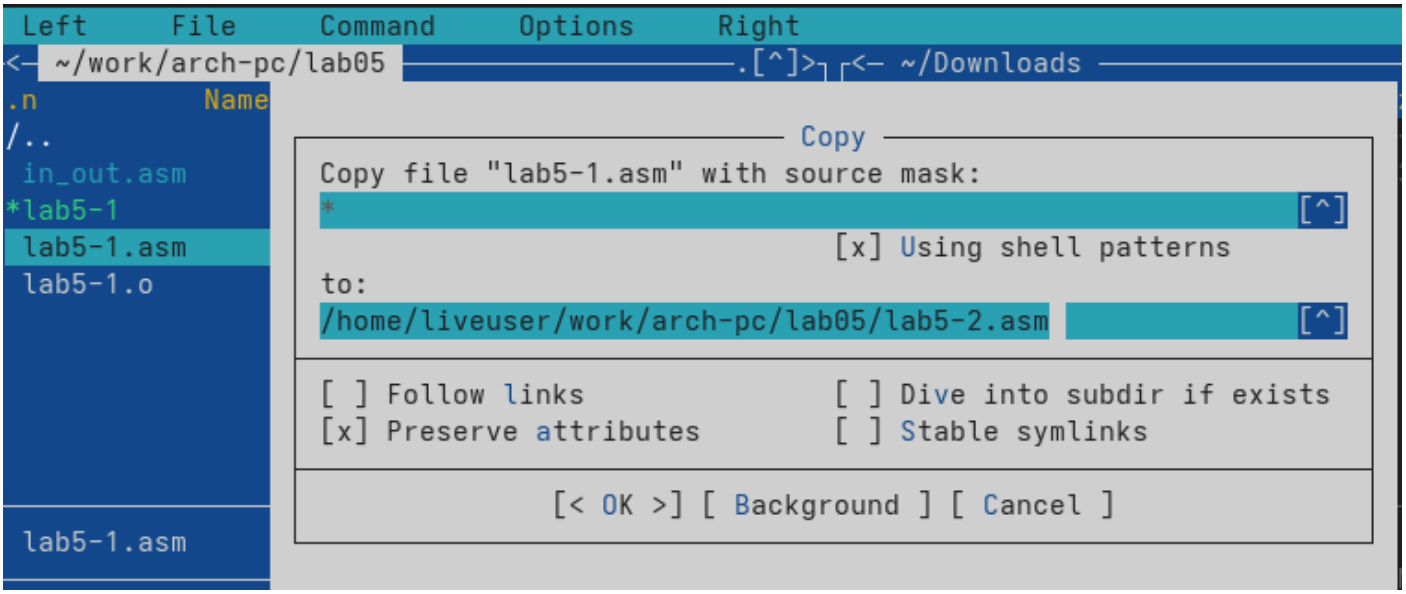
Задание 2. Подключение внешнего файла in_out.asm

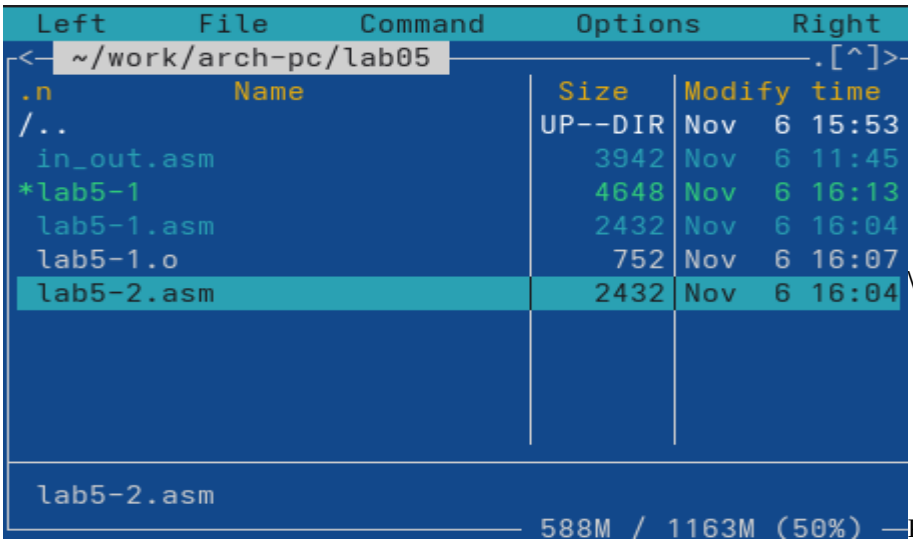
Необходимо было скачать файл in_out.asm, скопировать его в рабочий каталог, создать копию lab5-1.asm с именем lab5-2.asm и модифицировать ее для использования подпрограмм из in_out.asm.

В одной панели mc я открыл свой рабочий каталог ~/work/arch-pc/lab05, а в другой панели mc (переключившись по Tab) я открыл каталог ~/Загрузки. Я выделил файл in_out.asm и нажал F5 (Копирование), подтвердив копирование в lab05.



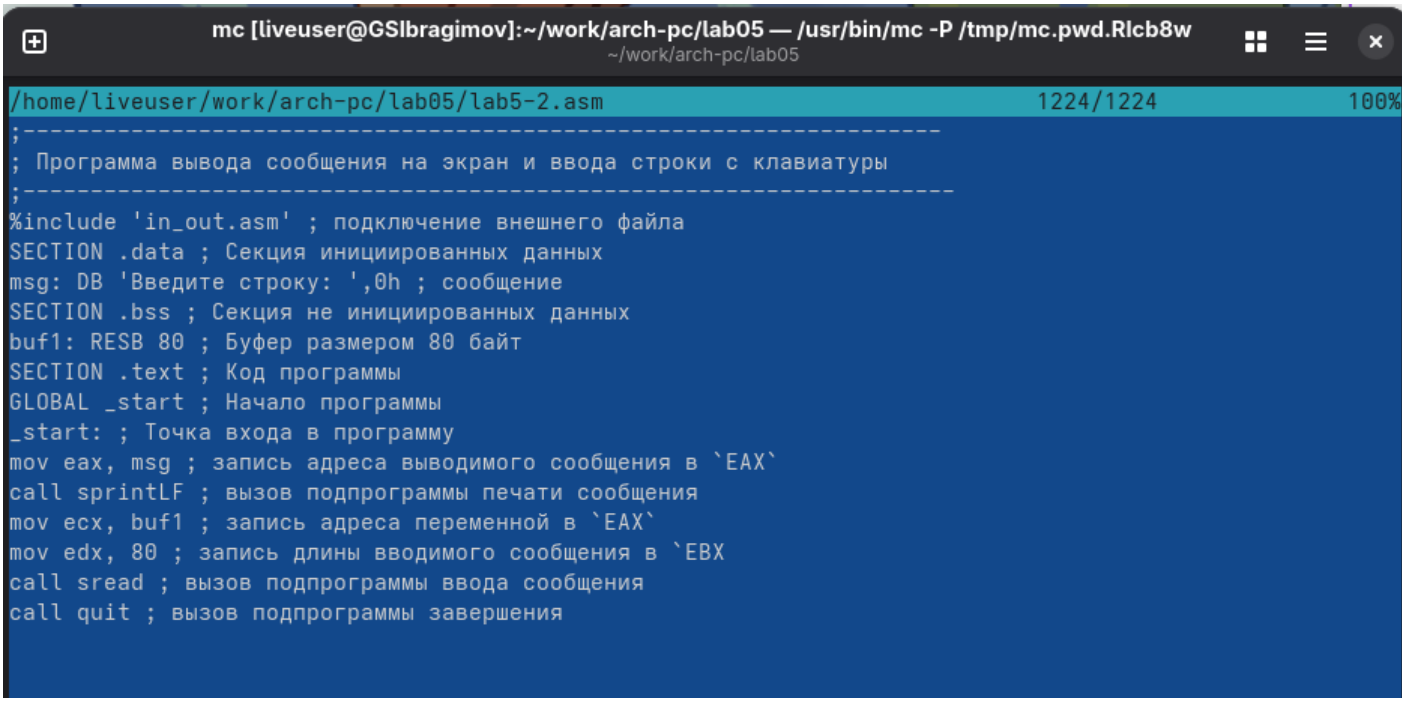
Далее я выделил файл lab5-1.asm в папке lab05, нажал F6 (Переименовать/Переместить) и ввел новое имя lab5-2.asm, чтобы создать копию.



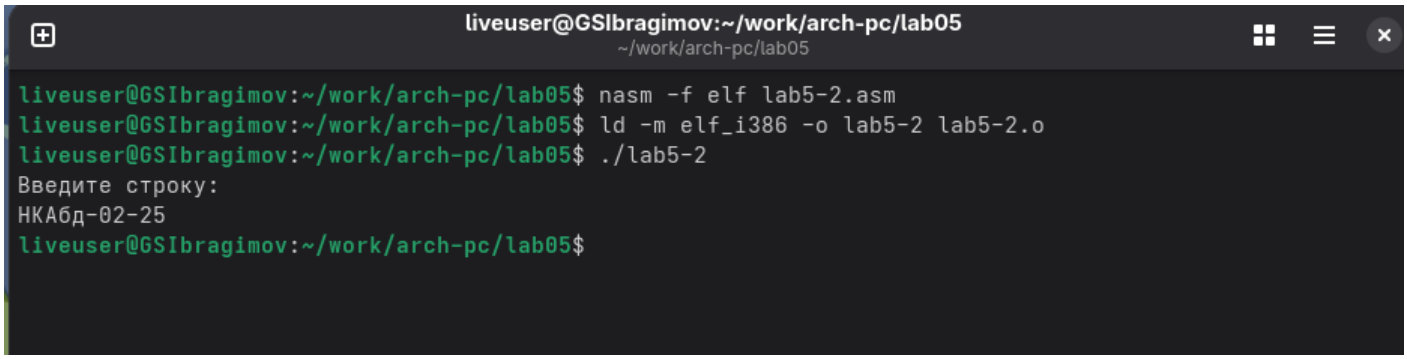


Копия создана

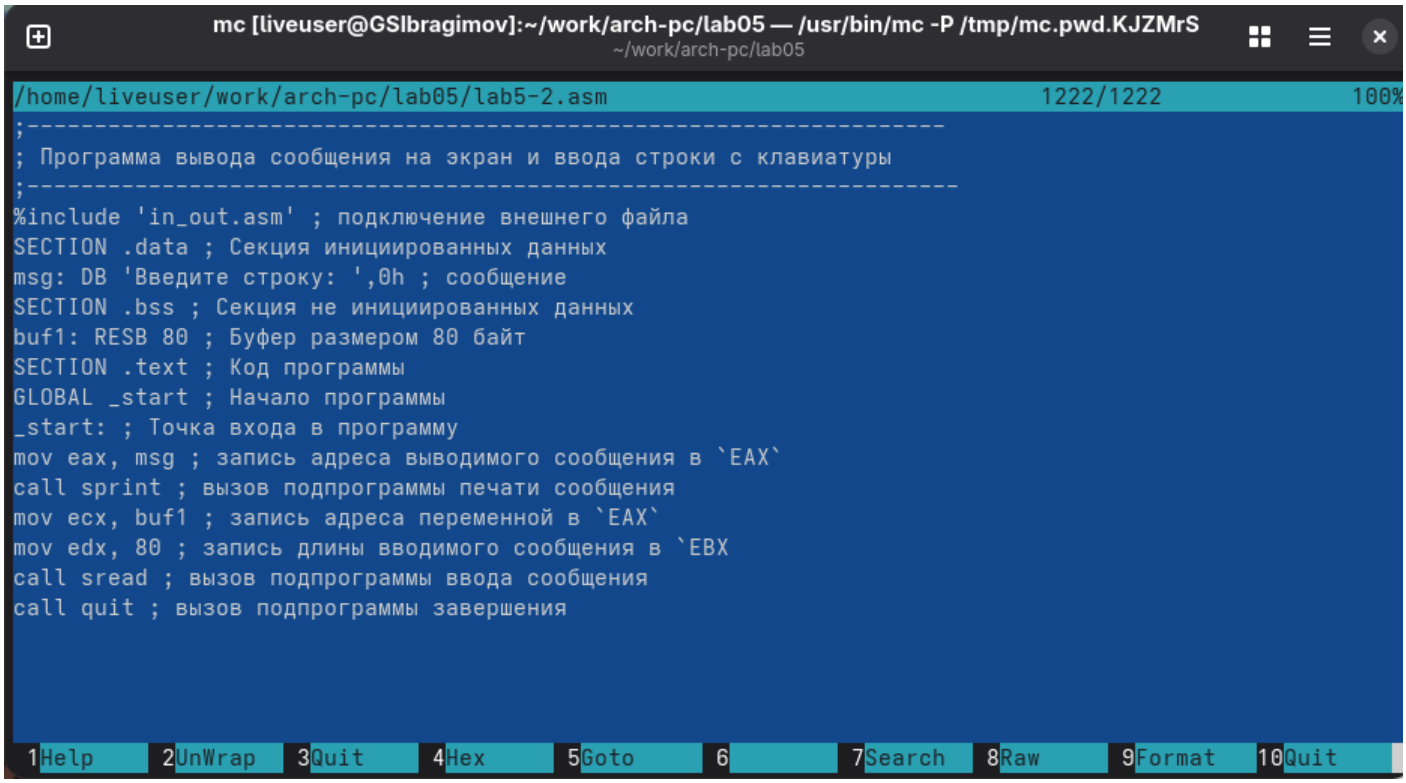
Я открыл lab5-2.asm на редактирование (F4) и изменил код в соответствии с Листингом 5.2, используя директиву %include и вызовы call. Далее просмотрел полученный результат.



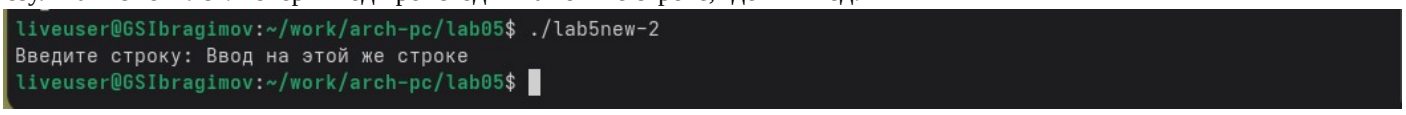
Теперь выполним трансляцию, компановку и запуск программы. Я ввел номер группы на следующей строке. Программа отработала идентично предыдущей.



Я снова открыл lab5-2.asm и заменил call sprintLF на call sprint. Сохранил под новым названием, скомпилировал и запустил.



Результат изменился. Теперь ввод происходит на той же строке, где и вывод.



Я научился использовать mc для операций с файлами между панелями (F5, F6). Использование in_out.asm сильно упрощает код: вместо кучи строк для каждого системного вызова теперь достаточно одной строки call.

Описание самостоятельной работы

Задание 1. Вывод введенной строки (без in_out.asm)

Мне нужно создать копию lab5-1.asm. Модифицировать программу так, чтобы она сначала выводила приглашение, затем считывала строку с клавиатуры, а после этого выводила эту же введенную строку обратно на экран.

Я создал копию lab5-1.asm и назвал ее lab5-1_.asm и открыл ее в редакторе (F4). После блока "системный вызов read" я добавил еще один "системный вызов write". Этот новый вызов использует в качестве адреса (ecx) буфер buf1, куда read сохранил введенную строку, и ту же длину 80 в edx.

mc [liveuser@GSIbragimov]:~/work/arch-pc/lab05 — /usr/bin/mc -P /tmp/mc.pwd.lcSI5v
~/work/arch-pc/lab05

/home/liveuser/work/arch-pc/lab05/lab5-1_.asm 2882/2882 100%

```
;------------------- 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 `write`
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из buf1
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx, buf1 ; Адрес строки '
mov edx, 80 ; Размер 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
```

Результат работы программы. Ввод и вывод на следующей строке.

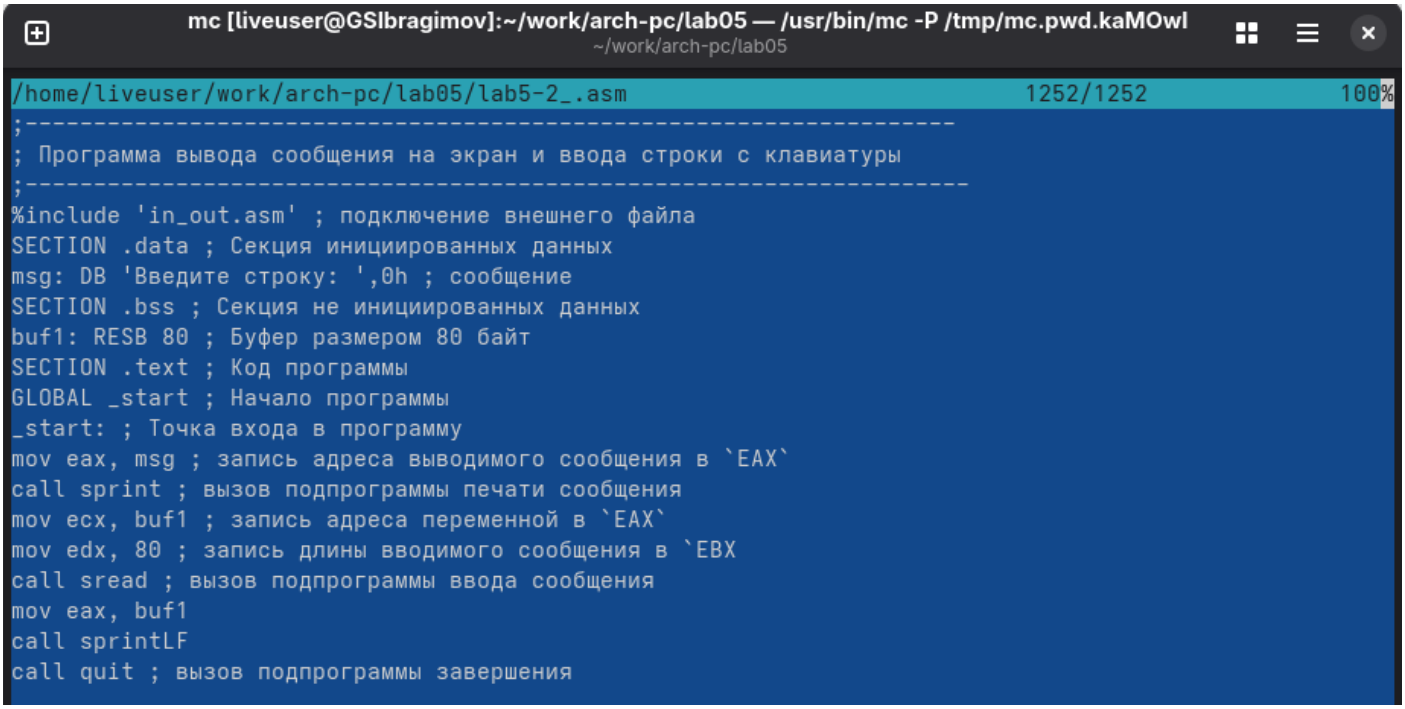
liveuser@GSIbragimov:~/work/arch-pc/lab05
~/work/arch-pc/lab05

```
liveuser@GSIbragimov:~/work/arch-pc/lab05$ ./lab5-1_
Введите строку:
Ибрагимов Гаджимурад
Ибрагимов Гаджимурад
liveuser@GSIbragimov:~/work/arch-pc/lab05$
```

Задание 2. Вывод введенной строки (с in_out.asm)

Теперь реализуем тот же алгоритм (приглашение -> ввод -> вывод введенной строки), но с использованием подпрограмм из in_out.asm.

Я создал копию lab5-2.asm и назвал ее lab5-2_.asm и открыл ее в редакторе (F4). Модифировал и сохранил.



Эта задача решается еще проще с in_out.asm. Я просто добавил mov eax, buf1 и call sprintLF после call sread.

Результат работы программы…

```
liveuser@GSIbragimov:~/work/arch-pc/lab05$ ./lab5-2_
Введите строку: Добрый вечер
Добрый вечер

liveuser@GSIbragimov:~/work/arch-pc/lab05$
```

Вывод

Я получил практические навыки работы с файловым менеджером Midnight Commander. Также я освоил базовую структуру программы на ассемблере NASM (секции .data, .bss, .text) и научился использовать ключевые инструкции:

- **mov**: для перемещения данных (адресов, констант) в регистры (eax, ebx, ecx, edx) для подготовки системных вызовов.
- **int 80h**: для вызова ядра Linux и выполнения системных функций, таких как sys_write, sys_read и sys_exit

Я также научился подключать внешние файлы (%include) и использовать подпрограммы (call), что значительно упрощает написание кода.

Источники

1. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander.org/.
2. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
3. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
4. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).