**Шаблон отчёта по лабораторной работе**

**Лабораторная работа №5.Основы работы с Midnight Commander**

**(mc).Структура программы на языке ассемблера NASM.Системные вызовы**

**в ОС GNU Linux**

Курбанов Рахман НКАбд-06-24

**Содержание**

**1 Цель работы :**

**2 Описание результатов выполнения лабораторной работы:**

2.1 описание выполняемого задания : . . . . . . . . . . . . . . . . . . . .

2.1.1 1. Открытие Midnight Commander . . . . . . . . . . . . . . . .

2.1.2 2. Переход в каталог ~/work/arch-pc . . . . . . . . . . . . . . .

2.1.3 3. Создание файла lab5-1.asm . . . . . . . . . . . . . . . . . . .

2.1.4 4. Открытие файла lab5-1.asm для редактирования . . . . . .

2.1.5 5. Ввод текста программы . . . . . . . . . . . . . . . . . . . . .

2.1.6 6. Просмотр файла lab5-1.asm . . . . . . . . . . . . . . . . . . .

2.1.7 7. Трансляция программы в объектный файл . . . . . . . . .

2.1.8 8. Ввод ваших ФИО . . . . . . . . . . . . . . . . . . . . . . . . .

2.1.9 9. Скачивание файла in\_out.asm . . . . . . . . . . . . . . . . .

2.1.10 10. Копирование файла in\_out.asm . . . . . . . . . . . . . . . .

2.1.11 11. Создание копии lab5-1.asm . . . . . . . . . . . . . . . . . .

2.1.12 12. Исправление текста программы в lab5-2.asm . . . . . . .

2.1.13 13. Замена подпрограммы sprintLF на sprint . . . . . . . . . .

2.2 выводы по результатам выполнения заданий . . . . . . . . . . . . .

**3 Описание результатов выполнения заданий для самостоятельной работы:**

3.1 описание выполняемого задания: . . . . . . . . . . . . . . . . . . . .

3.1.1 Создание копии файлах lab5-1.asm lab5-2.asm in\_out.asm : .

3.1.2 Получение исполняемого файла . . . . . . . . . . . . . . . . .

3.1.3 4. Проверка работы программы . . . . . . . . . . . . . . . . .

3.2 выводы по результатам выполнения заданий : . . . . . . . . . . . .

**4 Вопросы для самопроверки :**

4.1 1. Каково назначение mc? . . . . . . . . . . . . . . . . . . . . . . . . .

4.2 2. Какие операции с файлами можно выполнить как с помощью команд bash, так и с помощью меню (комбинаций клавиш) mc? Приведите несколько примеров. . . . . . . . . . . . . . . . . . . . . . . .

4.3 3. Какова структура программы на языке ассемблера NASM? . . . .

4.4 4. Для описания каких данных используются секции bss и data в языке ассемблера NASM? . . . . . . . . . . . . . . . . . . . . . . . . .

4.5 5. Для чего используются компоненты db, dw, dd, dq и dt языка ассемблера NASM? . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

4.6 6. Какое произойдет действие при выполнении инструкции mov eax, esi? . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

4.7 7. Для чего используется инструкция int 80h? . . . . . . . . . . . . .

4.8 Выводы,согласованные с целью работы : . . . . . . . . . . . . . . . .

**Список иллюстраций**

2.1 рисунок 01 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.2 рисунок 02 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.3 рисунок 03 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.4 рисунок 04 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.5 рисунок 05 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.6 рисунок 06 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.7 рисунок 07 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.8 рисунок 08 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.9 рисунок 09 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.10 рисунок 10 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.11 рисунок 11 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.12 рисунок 12 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.13 рисунок 13 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.14 рисунок 14 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.15 рисунок 15 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.1 рисунок 16 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.2 рисунок 17 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**1 Цель работы :**

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

**2 Описание результатов выполнения**

**лабораторной работы:**

**2.1 описание выполняемого задания :**

**2.1.1 1. Открытие Midnight Commander**

Сначала мы открываем Midnight Commander, чтобы удобно работать с файлами. Это делается с помощью команды

kurbanov@vbox:~$ mc

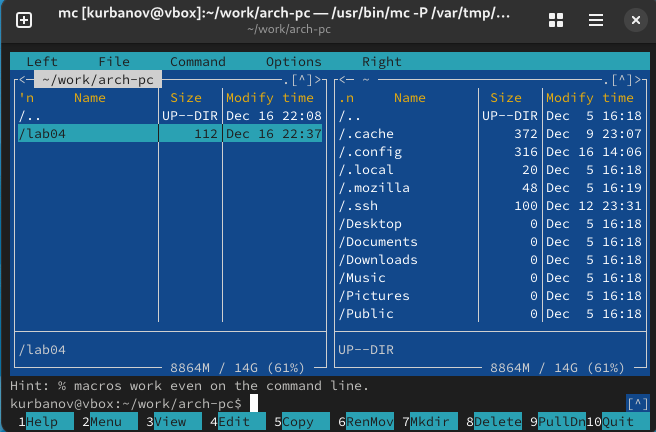
****

Рис. 2.1: рисунок 01

Комментарий: Вот как выглядит Midnight Commander после запуска. Очень удобно, особенно для навигации по файлам и папкам!

**2.1.2 2. Переход в каталог ~/work/arch-pc**

После открытия Midnight Commander, мы переходим в каталог ~/work/arch-pc, который мы создали во время выполнения лабораторной работы №4. Это можно сделать с помощью клавиш на клавиатуре

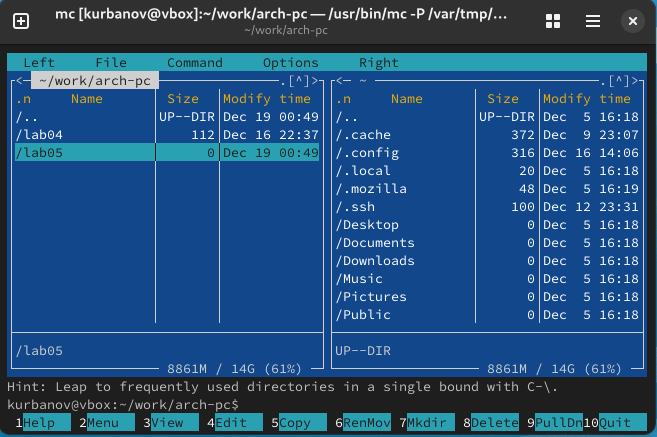


Рис. 2.2: рисунок 02

Комментарий: Мы находимся в каталоге arch-pc. Важно убедиться, что мы в нужном месте перед тем, как продолжать!

**2.1.3 3. Создание файла lab5-1.asm**

Теперь мы создаем новый файл lab5-1.asm. Для этого используем команду touch в строке ввода:

kurbanov@vbox:~$ touch lab5-1.asm

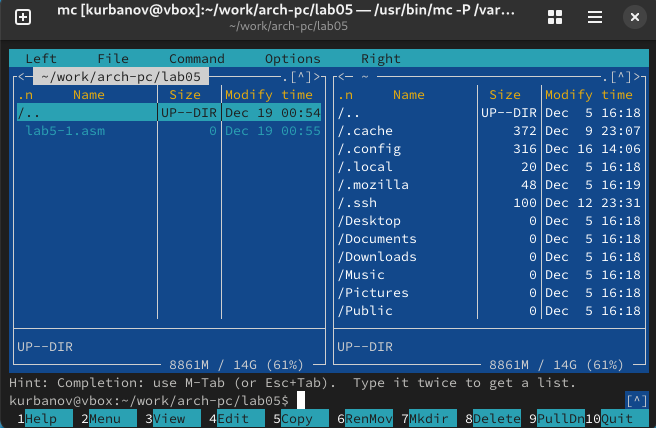
****

Рис. 2.3: рисунок 03

Комментарий: Файл lab5-1.asm успешно создан. Это будет наш первый файл для написания кода на ассемблере.

**2.1.4 4. Открытие файла lab5-1.asm для редактирования**

Далее мы открываем файл lab5-1.asm для редактирования с помощью функциональной клавиши F4. В Midnight Commander встроенный редактор обычно либо nano, либо mcedit

**2.1.5 5. Ввод текста программы**

Теперь вводим текст программы из листинга 5.1 (можно без комментариев). После того как введем текст, не забываем сохранить изменения и закрыть файл.

**2.1.6 6. Просмотр файла lab5-1.asm**

После редактирования открываем файл lab5-1.asm для просмотра с помощью F3. Это позволит нам убедиться, что все записано правильно.

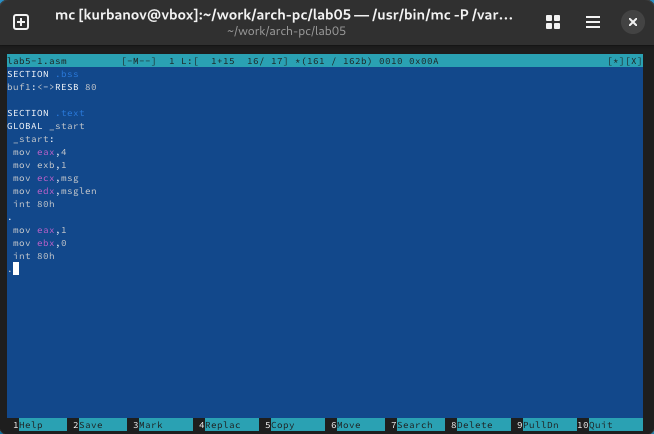


Рис. 2.4: рисунок 04

Комментарий: Проверяем файл. Всё выглядит хорошо, код на месте!

**2.1.7 7. Трансляция программы в объектный файл**

Теперь мы переводим текст программы в объектный файл. Для этого используем NASM и LD:

kurbanov@vbox:~$ nasm -f elf lab5-1.asm

kurbanov@vbox:~$ ld -m elf\_i386 -o lab5-1 lab5-1.o

После этого запускаем исполняемый файл:

kurbanov@vbox :~$ ./lab5-1

Введите строку:

**2.1.8 8. Ввод ваших ФИО**

На запрос вводим свои ФИО.

Имя пользователя

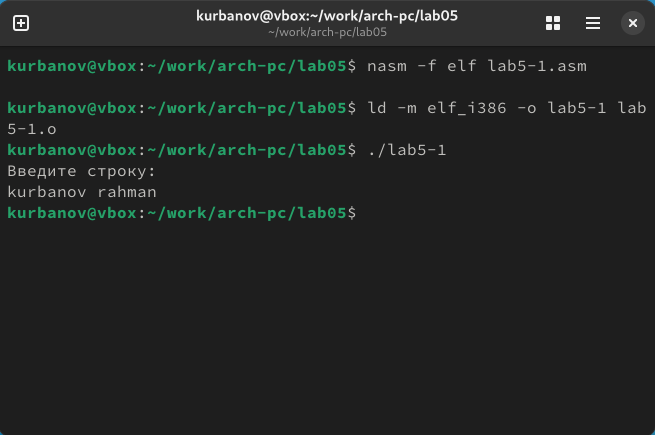


Рис. 2.5: рисунок 05

Комментарий: Вводим свои данные. Это часть, когда мы тестируем, как программа реагирует на ввод!

**2.1.9 9. Скачивание файла in\_out.asm**

Следующим шагом скачиваем файл in\_out.asm со страницы курса в ТУИС. Этот файл будет содержать подпрограммы, которые нам понадобятся.

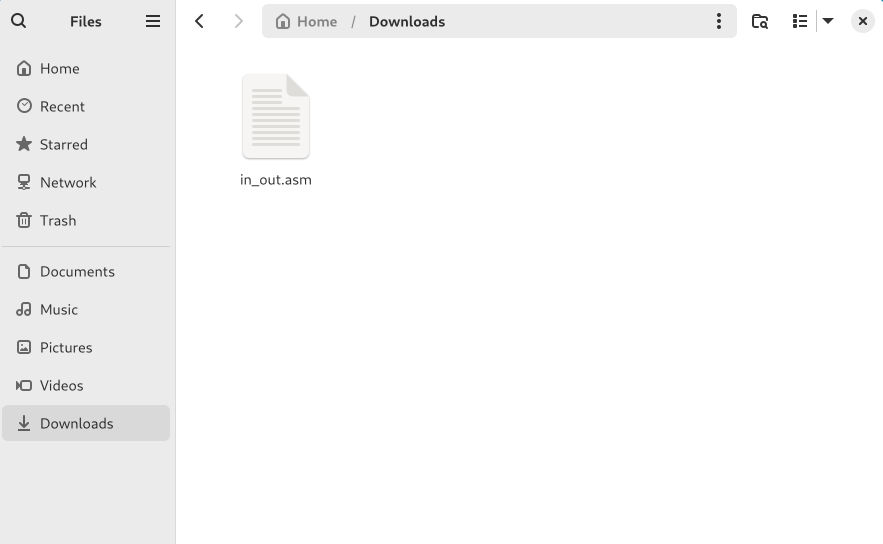
****

Рис. 2.7: рисунок 07

Комментарий: Скачивание файла in\_out.asm. Это важно для следующих шагов в лабораторной работе!

**2.1.10 10. Копирование файла in\_out.asm**

Убедитесь, что файл in\_out.asm находится в том же каталоге, что и lab5-1.asm. В одной из панелей Midnight Commander открываем каталог с lab5-1.asm, а в другой — каталог со скачанным файлом. Используем функциональную клавишу F5 для копирования.

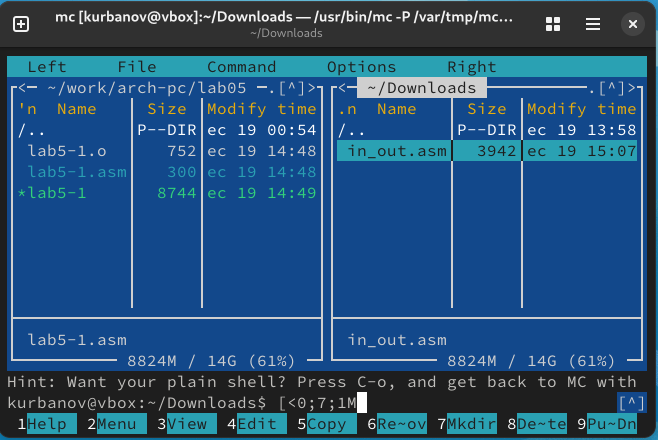


Рис. 2.8: рисунок 08

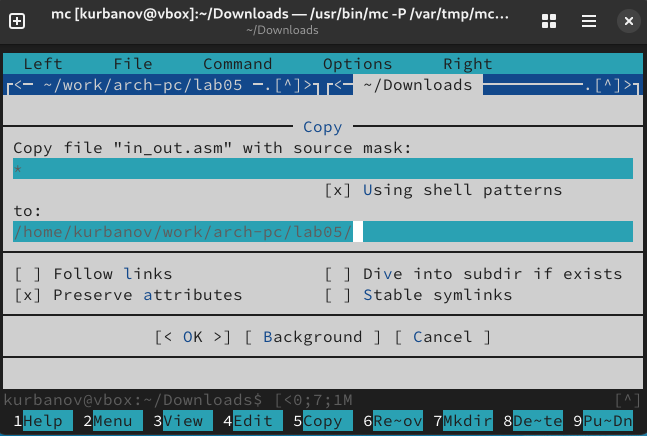


Рис. 2.9: рисунок 09

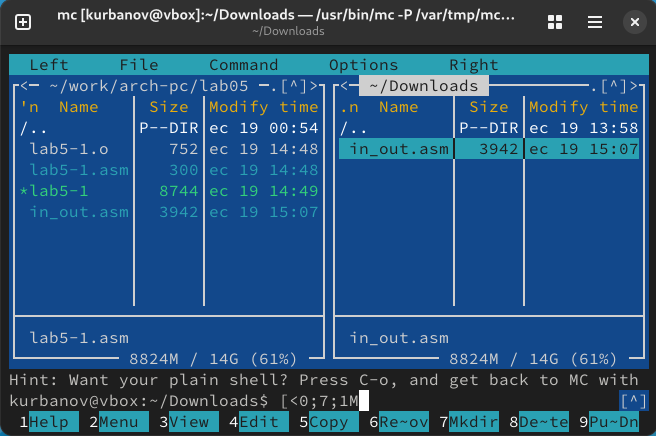


Рис. 2.10: рисунок 10

Комментарий: Копируем in\_out.asm в нужный каталог. Теперь у нас есть все необходимые файлы!

**2.1.11 11. Создание копии lab5-1.asm**

Теперь создаем копию файла lab5-1.asm с именем lab5-2.asm с помощью F5. Это позволяет нам работать с новой версией программы, не теряя оригинал.

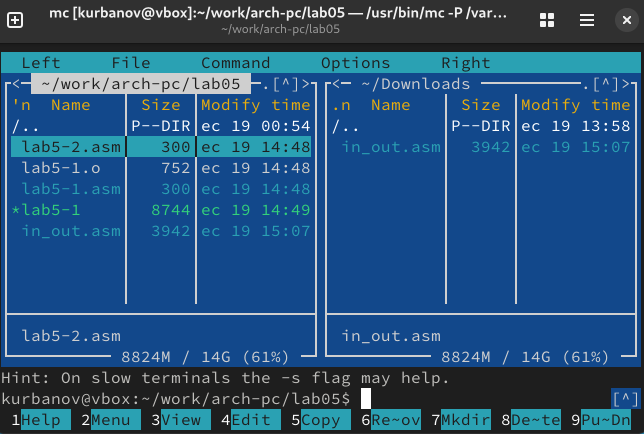


Рис. 2.11: рисунок 11

Комментарий: Создаем копию. Всегда полезно иметь резервную копию оригинального файла!

**2.1.12 12. Исправление текста программы в lab5-2.asm**

Теперь мы изменяем текст программы в lab5-2.asm, используя подпрограммы из in\_out.asm (например, sprintLF, sread и quit). После этого создаем исполняемый файл и проверяем его работу

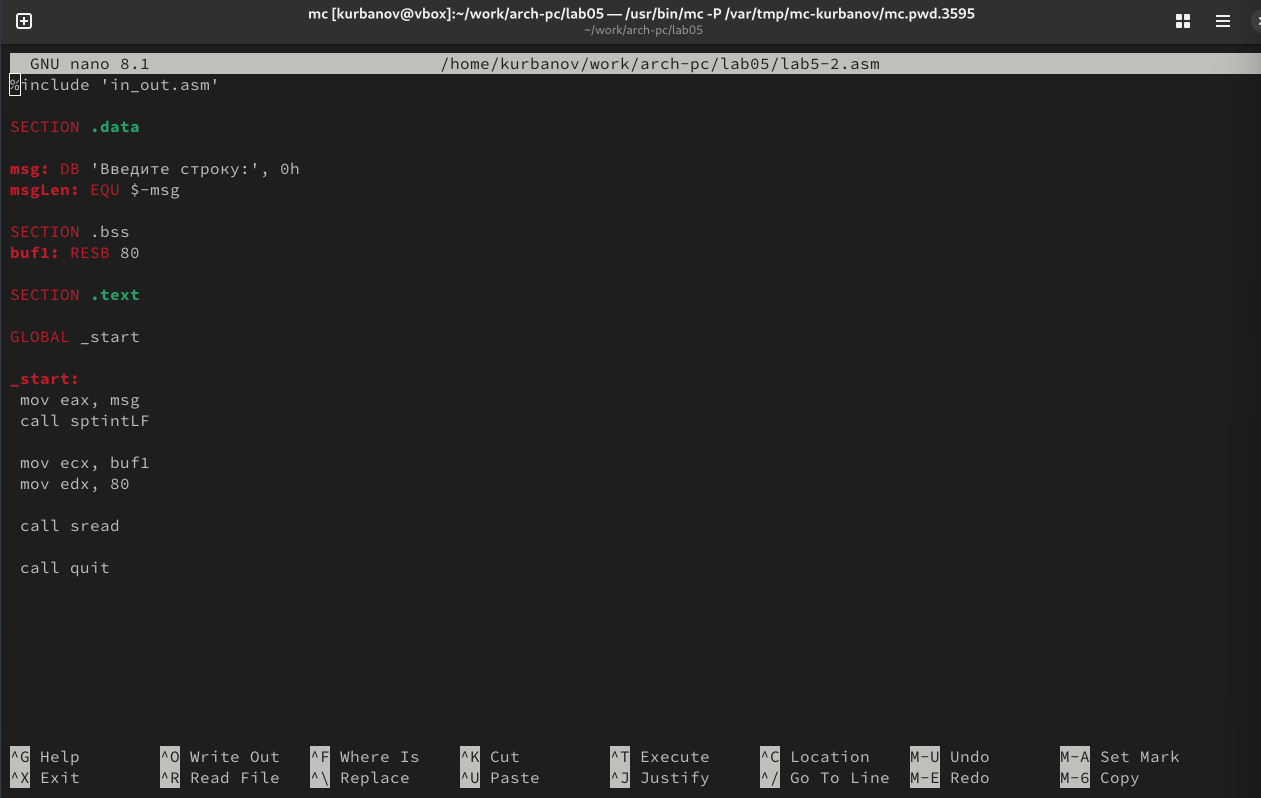


Рис. 2.12: рисунок 12

Комментарий: Внесенные изменения. Теперь программа должна использовать новые подпрограммы!

**2.1.13 13. Замена подпрограммы sprintLF на sprint**

Наконец, мы заменяем подпрограмму sprintLF на sprint в lab5-2.asm. После этого создаем исполняемый файл и проверяем его работу.

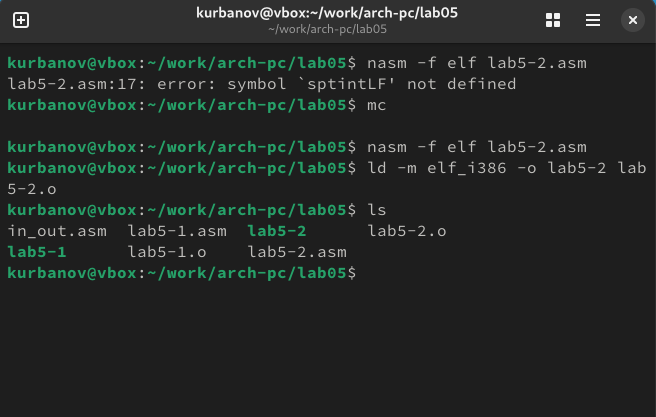
****

Рис. 2.13: рисунок 13

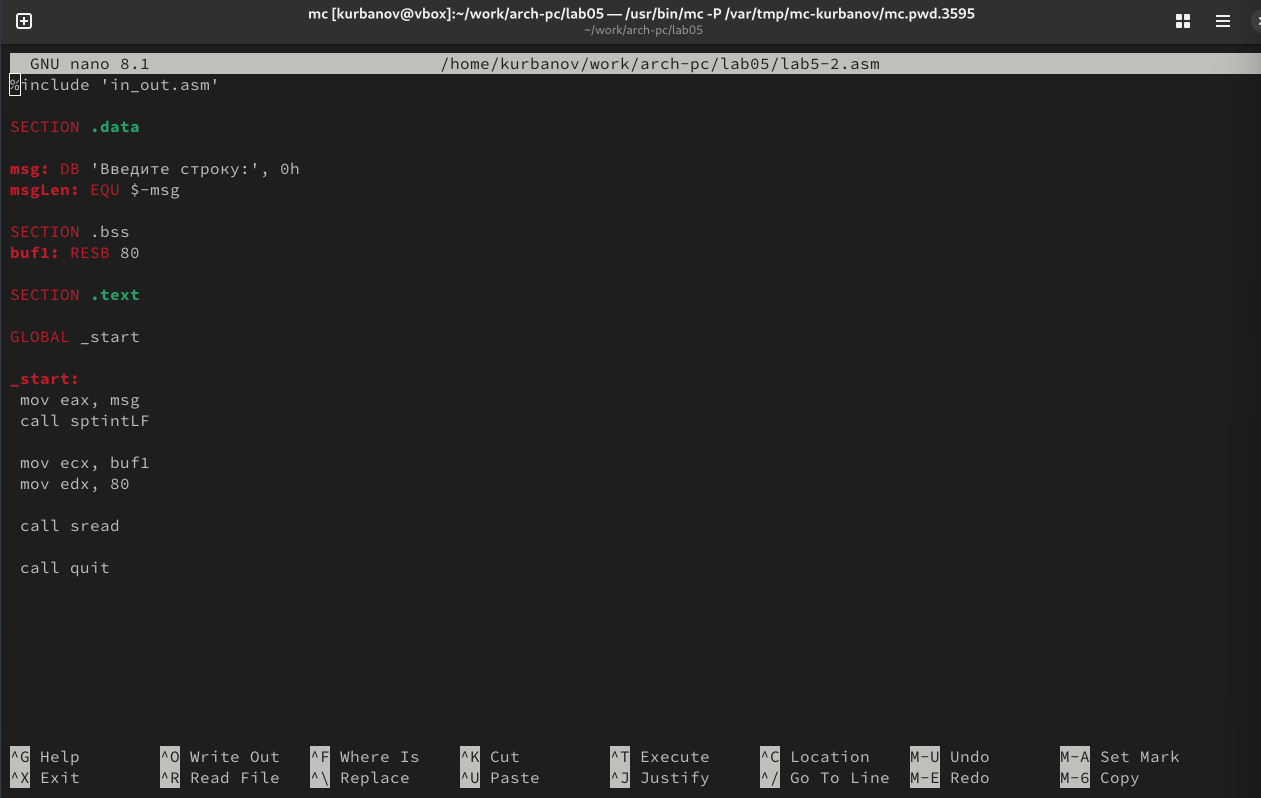


Рис. 2.14: рисунок 14

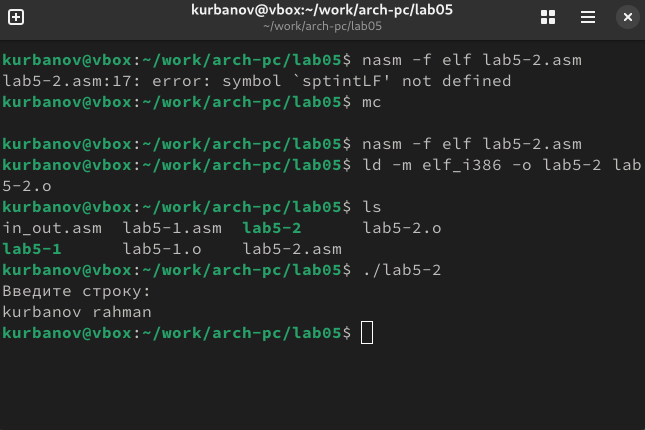


Рис. 2.15: рисунок 15

Комментарий: Подпрограмма заменена, всё готово к тестированию. Важно понимать разницу между sprint и sprintLF: первая просто выводит строку, а вторая — с 16 добавлением новой строки!

**2.2 выводы по результатам выполнения заданий**

В ходе лабораторной работы мы создали и изменили ассемблерные файлы, научились работать с Midnight Commander, а также использовали подпрограммы из внешнего файла. Это был полезный опыт в работе с ассемблером и системным программированием!

3 Описание результатов выполнения заданий для самостоятельной работы:

**3.1 описание выполняемого задания:**

**3.1.1 Создание копии файлах lab5-1.asm lab5-2.asm in\_out.asm :**

Сначала мы создаем копию файла lab5-1.asm. Это нужно, чтобы внести изменения, не затрагивая оригинал

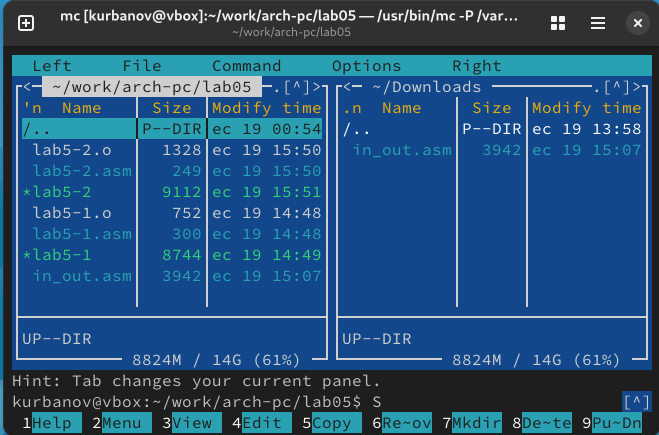


Рис. 3.1: рисунок 16

Комментарий: Копия файла создана. Теперь у нас есть оригинал и копия для работы!

**3.1.2 Получение исполняемого файла**

После внесения изменений, мы трансформируем наш ассемблерный файл в объектный файл и компилируем его в исполняемый файл. Вот команды, которые мы используем:

kurbanov@vbox:~$ nasm -f elf lab5-1.asm

kurbanov@vbox: ~$ nasm -f elf lab5-2.asm

kurbanov@vbox: ~$ ld -m elf\_i386 -o lab5-1 lab5-1.o

kurbanov@vbox: ~$ ld -m elf\_i386 -o lab5-2 lab5-2.o

**3.1.3 4. Проверка работы программы**

Теперь запускаем исполняемый файл и вводим свои ФИО, когда программа запрашивает строку.

kurbanov@vbox: ~$ ./lab5-1

kurbanov@vbox: ~$ ./lab5-2

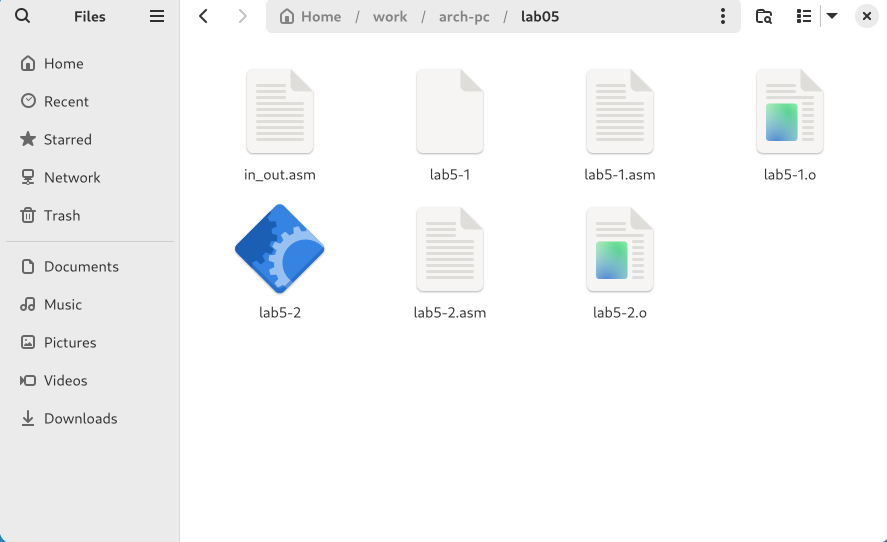


Рис. 3.2: рисунок 17

**3.2 выводы по результатам выполнения заданий :**

В ходе самостоятельной работы мы изменили несколько ассемблерных файлов, проверили их работу и узнали, как использовать подпрограммы из внешнего файла. Это дало нам полезный опыт в написании и компиляции кода на ассемблере!

**4 Вопросы для самопроверки :**

4.1 1. Каково назначение mc?

Ответ: mc (Midnight Commander) — это текстовый файловый менеджер для UNIXподобных систем, который позволяет пользователям удобно управлять файлами и каталогами через графический интерфейс.

4.2 2. Какие операции с файлами можно выполнить как с помощью команд bash, так и с помощью меню (комбинаций клавиш) mc? Приведите несколько примеров.

Ответ: - Копирование файлов: можно использовать команду cp в bash или сочетание клавиш F5 в mc. - Перемещение файлов: можно использовать команду mv в bash или F6 в mc. - Удаление файлов: команда rm в bash и F8 в mc.

4.3 3. Какова структура программы на языке ассемблера NASM?

Ответ: Программа на языке ассемблера NASM состоит из следующих секций: .data (для инициализированных данных), .bss (для неинициализированных 21 данных) и .text (для кода программы).

4.4 4. Для описания каких данных используются секции bss и data в языке ассемблера NASM?

Ответ: - Секция .data: используется для хранения инициализированных данных (например, строки, массивы). - Секция .bss: используется для хранения неинициализированных данных (например, переменные, которые не имеют начального значения).

4.5 5. Для чего используются компоненты db, dw, dd, dq и dt языка ассемблера NASM?

Ответ: - db: для объявления байтов (1 байт). - dw: для объявления слов (2 байта). - dd: для объявления двойных слов (4 байта). - dq: для объявления Quad слов (8 байт). - dt: для объявления десятичных значений (различные размеры).

4.6 6. Какое произойдет действие при выполнении инструкции mov eax, esi?

Ответ: Инструкция mov eax, esi копирует значение из регистра esi в регистр eax.

4.7 7. Для чего используется инструкция int 80h?

Ответ: Инструкция int 80h используется для вызова системных вызовов в Linux, позволяя программе взаимодействовать с операционной системой.

4.8 Выводы,согласованные с целью работы :

Мы освоили язык ассемблера NASM, включая работу с файлами, структуру программ и взаимодействие с операционной системой. Эти навыки углубили наше понимание программирования на низком уровне.