

Лабораторная работа №5

НКАбр-06-23

Улитина Мария Максимовна

Содержание

1 Цель работы.....	1
2 Задание	1
3 Теоретическое введение.....	2
3.1 Структура программы на языке ассемблера NASM.....	2
3.2 Элементы программирования.....	3
3.2.1 Описание инструкции mov.....	3
3.2.2 Описание инструкции int.....	3
4 Выполнение лабораторной работы.....	3
4.1 Задания для самостоятельной работы	8
5 Выводы.....	11
Список литературы	11

1 Цель работы

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

2 Задание

1. Создание и исполнение файла lab5-1.
2. Подключение внешнего файла in_out.asm
3. Создание копии файла lab5-1; внесение изменения в программу для вывода введённой строки на экран.
4. Создание копии файла lab5-2; внесение изменения в программу для вывода введённой строки на экран.

3 Теоретическое введение

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 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.

3.1 Структура программы на языке ассемблера NASM

Программа на языке ассемблера NASM, как правило, состоит из трёх секций: секция кода программы (SECTION .text), секция иницированных (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 иницированных 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.

3.2 Элементы программирования

3.2.1 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).

3.2.2 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).

4 Выполнение лабораторной работы

Откроем Midnight Commander, перейдём в необходимый каталог (рис.[1])

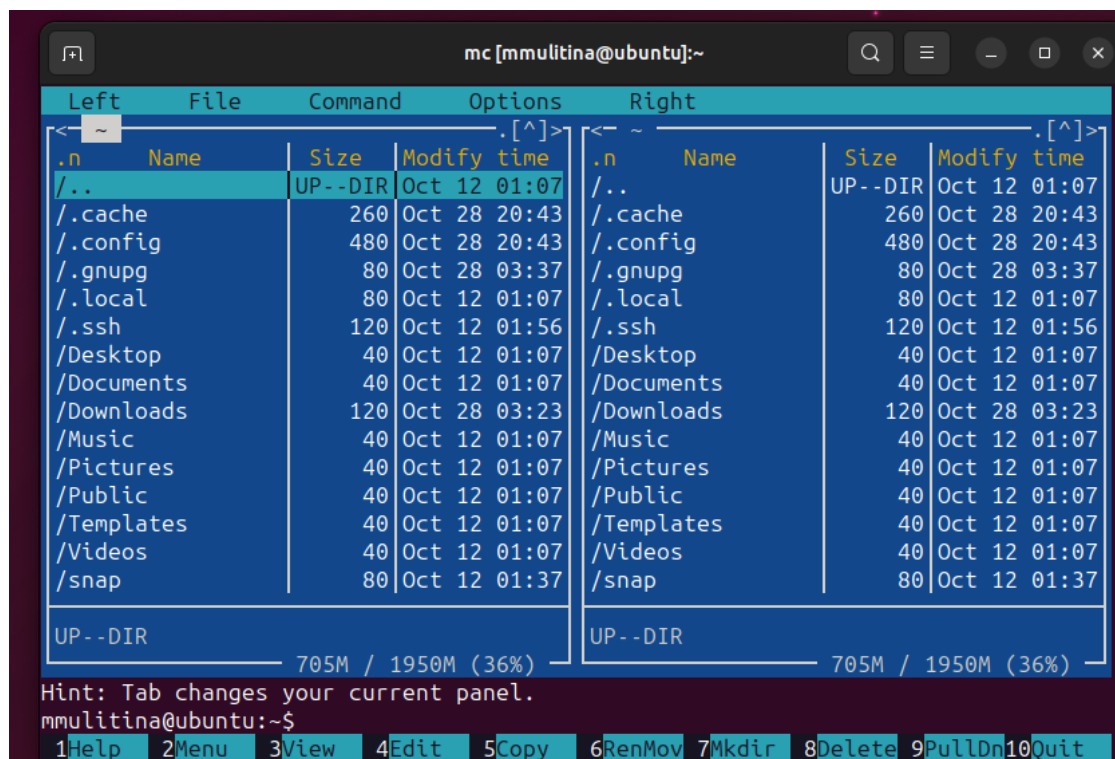


Figure 1: Midnight Commander

С помощью функциональной клавиши F7 создадим папку lab05 и перейдем в созданный каталог (рис.[2])

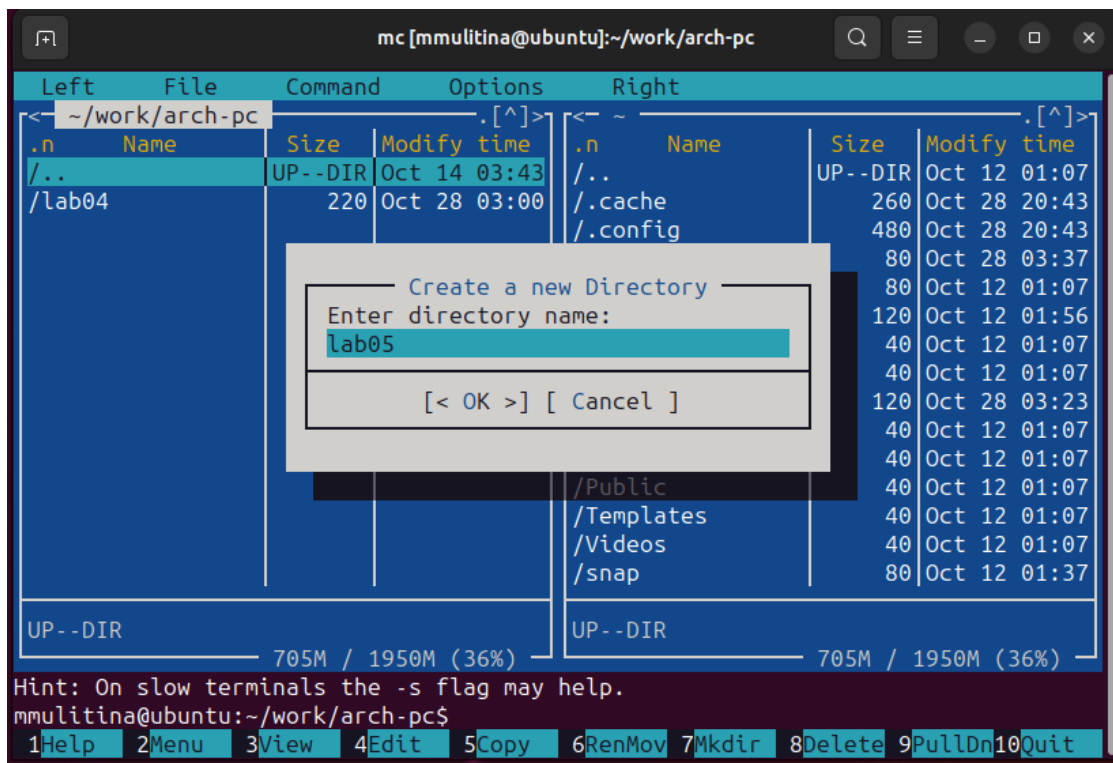


Figure 2: Создание папки

С помощью `touch` создадим файл `lab5-1.asm` (рис.[3])

```
mmulitina@ubuntu:~/work/arch-pc/lab05$ touch lab5-1.asm
```

Figure 3: Создание файла

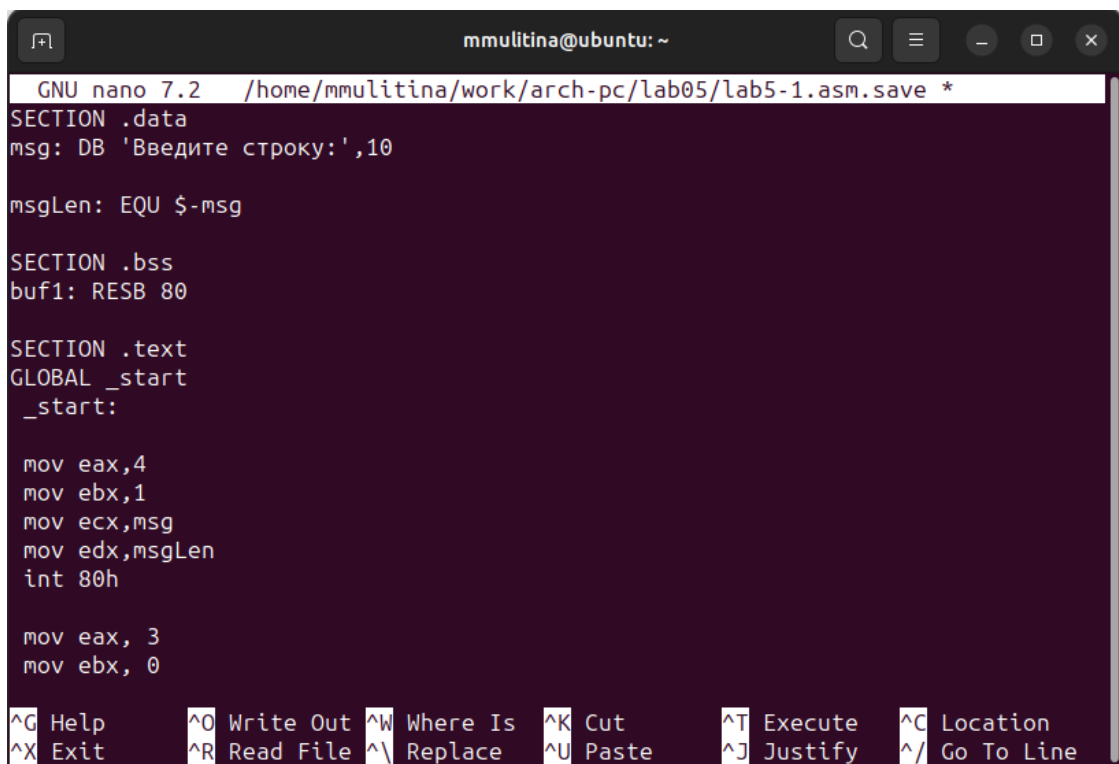
С помощью `F4` откроем файл `lab5-1.asm` для редактирования в редакторе `nano` (рис.[4])



The screenshot shows the GNU nano 7.2 editor interface. The title bar at the top indicates the file path: /home/mmulitina/work/arch-pc/lab05/lab5-1.asm. The main editing area is empty. At the bottom, a status bar displays various keyboard shortcuts: ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^C Location, ^X Exit, ^R Read File, ^\ Replace, ^U Paste, ^J Justify, and ^_ Go To Line. A small prompt [Read 0 lines] is visible above the shortcuts.

Figure 4: Редактирование

Введем текст программы (рис.[5])



The screenshot shows the GNU nano 7.2 editor with the file name lab5-1.asm.save *. The editor contains the following assembly code:

```
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
```

The bottom status bar shows the same keyboard shortcuts as in Figure 4.

Figure 5: Текст программы

С помощью F3 откроем файл lab5-1.asm для просмотра и убедимся, что она содержит текст программы.

Оттранслируем программу в объектный файл. Выполним компоновку объектного файла (рис.[6])

```
mmulitina@ubuntu:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
mmulitina@ubuntu:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
mmulitina@ubuntu:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
mmulitina
mmulitina@ubuntu:~/work/arch-pc/lab05$
```

Figure 6: Трансляция и компоновка

Скачаем файл in_out.asm и поместим его в соответствующий каталог. Создадим копию lab5-1.asm с помощью F6 (рис.[7])

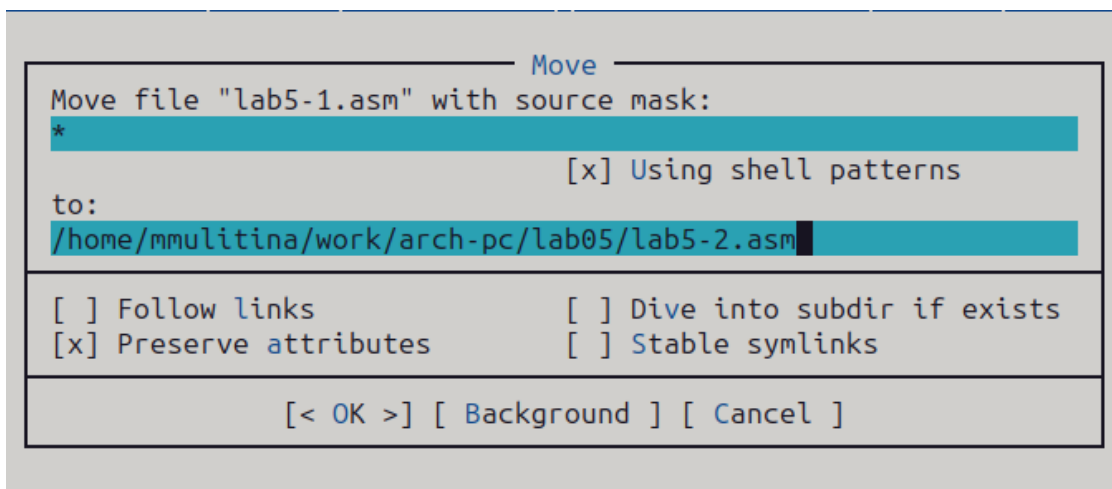


Figure 7: Копия

Исправим текст программы с использованием подпрограмм из внешнего файла (рис.[8])

```
SECTION .text
GLOBAL _start
_start:

mov eax,msg
call sprintf

mov ecx, buf1
mov edx, 80

call sread

call quit|
```

Figure 8: Внешний файл

Создадим исполняемый файл и проверим его работу (рис.[9])

```
mmulitina@ubuntu:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
mmulitina@ubuntu:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
mmulitina@ubuntu:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
mmulitina
```

Figure 9: Проверка

Заменяем sprintf на printf - теперь вывод и ввод на одной строке (рис.[10])

```
mmulitina@ubuntu:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
mmulitina@ubuntu:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
mmulitina@ubuntu:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:mmulitina
```

Figure 10: Замена

4.1 Задания для самостоятельной работы

Создадим копию файла lab5-1.asm (рис.[11])

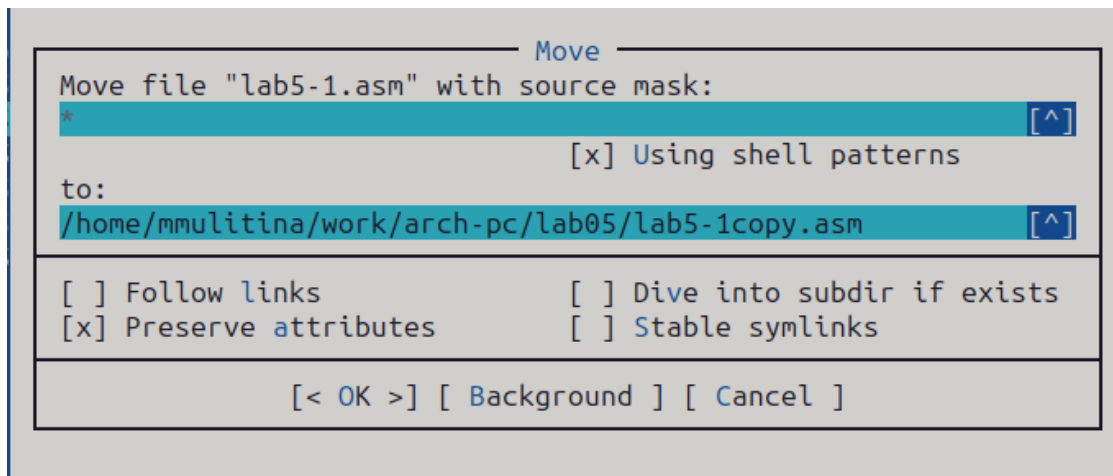


Figure 11: Копия

Внесём изменения в программу, чтобы она выводила введенную пользователем строку на экран (рис.[12])

```
mc [mmilitina@ubuntu]:~/work/arch-pc/lab05
GNU nano 7.2 /home/mmilitina/work/arch-pc/lab05/lab5-1copy.asm

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, 4
mov ebx, 1
mov ecx, buf1
mov edx, 80
int 80h

mov eax,1
```

Figure 12: Копия

Запустим программу (рис.[13])

```
mmilitina@ubuntu:~/work/arch-pc/lab05$ nasm -f elf lab5-1copy.asm
mmilitina@ubuntu:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1copy lab5-1copy.o
mmilitina@ubuntu:~/work/arch-pc/lab05$ ./lab5-1copy
Введите строку:
mmilitina
mmilitina
```

Создадим копию файла lab5-2.asm (рис.[14])

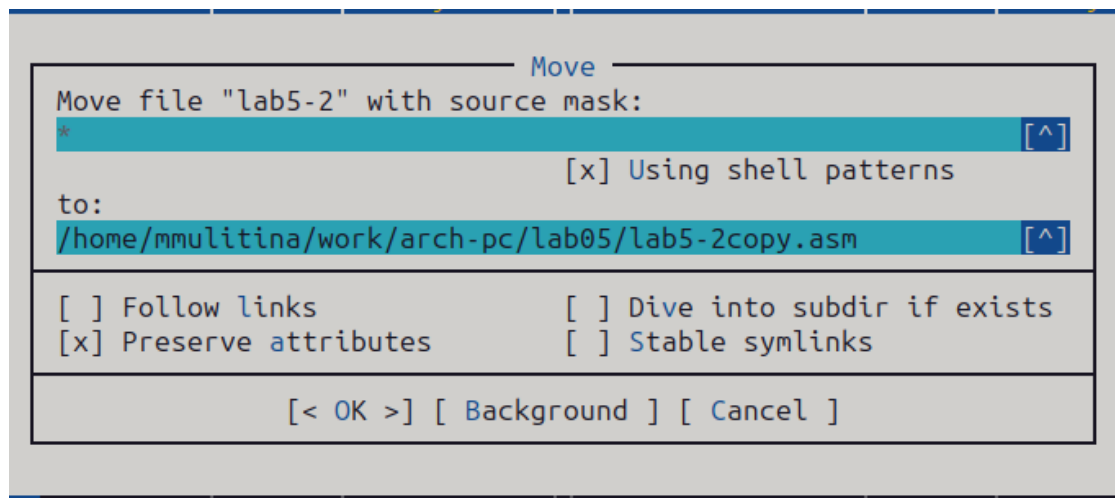


Figure 14: Копия

Внесём изменения в программу, чтобы она выводила введенную пользователем строку на экран (рис.[15])

```
SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

    mov eax,msg
    call sprint

    mov ecx, buf1
    mov edx, 80
    call sread

    mov eax, buf1
    call sprint
```

Figure 15: Копия

Запустим программу (рис.[16])

```
mmilitina@ubuntu:~/work/arch-pc/lab05$ nasm -f elf lab5-2copy.asm
mmilitina@ubuntu:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2copy lab5-2copy.o
mmilitina@ubuntu:~/work/arch-pc/lab05$ ./lab5-2copy
Введите строку:mmilitina
mmilitina
```

5 Выводы

В процессе выполнения работы я приобрела практические навыки работы в Midnight Commander и освоила инструкции на языке ассемблера mov и int.

Список литературы

Архитектура ЭВМ. Лабораторная работа №5.