Отчет по лабораторной работе №5

Дисциплина: Архитектура компьютера

Орлов Илья Сергеевич

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Целью данной лабораторной работы является приобретение практических навыков работы в Midnight Commander, освоение инструкций языка ассемблера mov и int.

2 Задание

1. Основы работы с mc
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла
4. Выполнение заданий для самостоятельной работы

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. 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 (учетве- рённое 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. Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике.

```
mov dst,src
```

Здесь операнд dst — приёмник, а src — источник. В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и непосредственные значения (const). Инструкция языка ассемблера intпредназначена для вызова прерывания с указанным номером.

```
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

4.1 Основы работы с Midnight Commander

Введя соответствующ комманду в терминале (рис. -fig. 4.1), я открываю Midnight Commander (рис. -fig. 4.2).

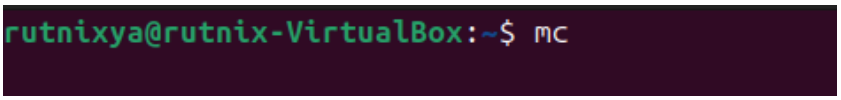

Рис. 4.1: Отркрытие Midnight Commander

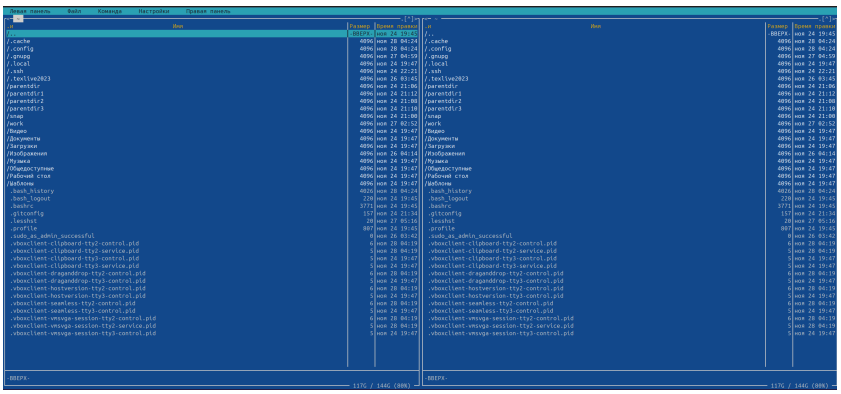

Рис. 4.2: Интерфейс Midnight Commander

Перехожу в созданный каталог в предыдущей лабораторной работе (рис. -fig. 4.3).

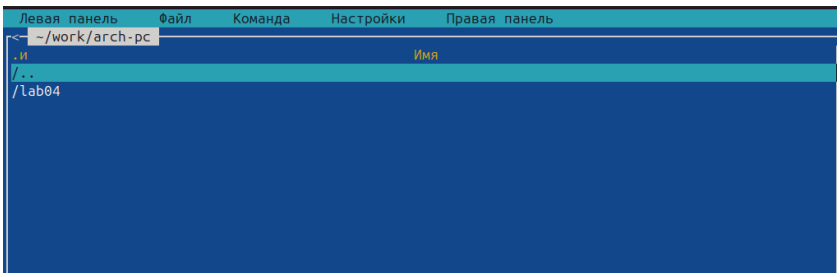

Рис. 4.3: Открытый каталог arch-pc

С помощью функциональной клавиши, я создаю подкаталог lab05, в котором буду работать (рис. -fig. 4.4).

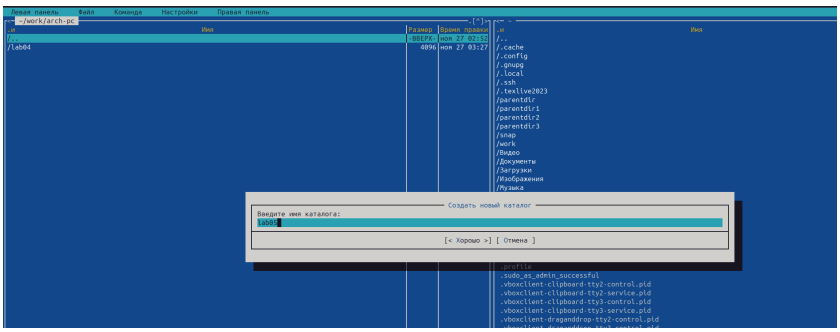

Рис. 4.4: Создание рабочего подкаталога

В строке ввода вводжу команду touch и создаю файл (рис. -fig. 4.5).

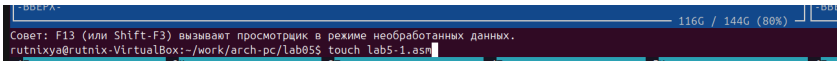

Рис. 4.5: Создание файла в Midnight Commander

4.2 Работа в NASM

С помощью F4 открываю только что созданный файл и вношу код с листинга (рис. -fig. 4.6).

Рис. 4.6: Редактирование файла в Midnight Commander

Проверяю сохраненные изменения с помощью клавиши F3 (рис. -fig. 4.7).

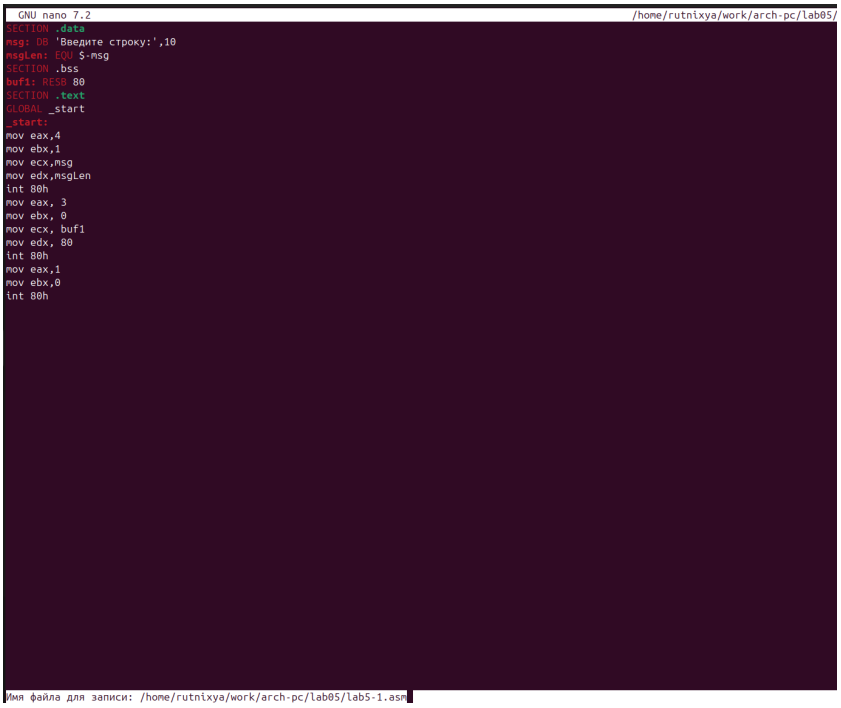

Рис. 4.7: Проверка сохранения сделанных изменений

Транслирую и компоную измененный файл, запускаю (рис. -fig. 4.8).

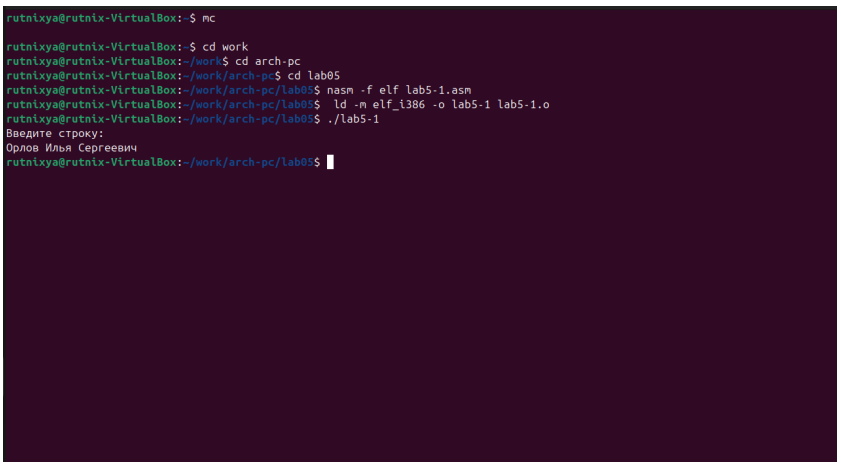

Рис. 4.8: Трансляция, компоновка и последующий запуск программы

4.3 Подключение внешнего файла

Скачанный с ТУИС файл сохраняю в общую папку на своем компьютере, на виртуальной машине в интерфейсе Midnight Commander перехожу в директорию общей папки, копирую файл в рабочий подкаталог. (рис. -fig. 4.9).

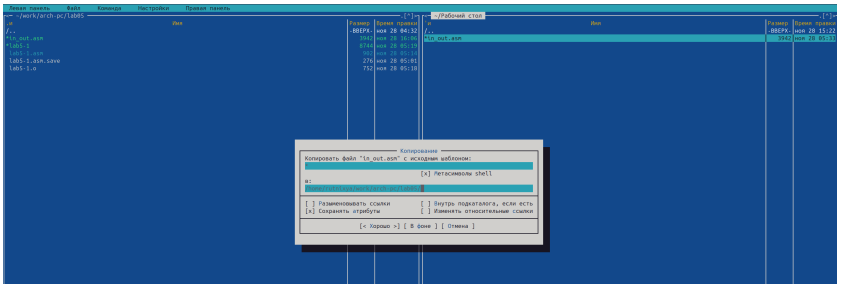

Рис. 4.9: Копирование файла в рабочий каталог

Создаю копию файла для последующей работы с ним (рис. -fig. 4.10).

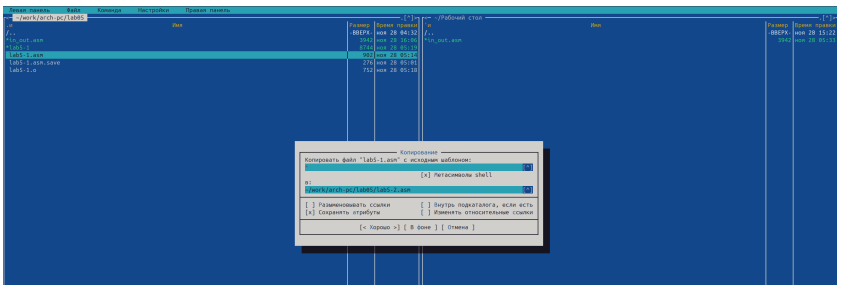

Рис. 4.10: Создание копии файла в Midnight Commander

В копии файла подключаю подпрограмм из подключенного файла (рис. -fig. 4.11).

Рис. 4.11: Изменение программы

Транслирую, компоную и запускаю программу с подключенным файлом (рис. -fig. 4.12).

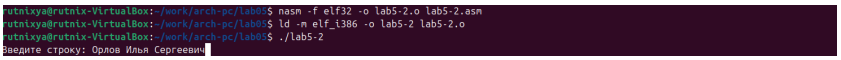

Рис. 4.12: Запуск измененной программы

Редактирую файл и заменяю в нем подпрограмму sprintLF на sprint. Разница подпрограмм в том, что вторая вызывает ввод на той же строке (рис. -fig. 4.13).

4.4 Задание для самостоятельной работы

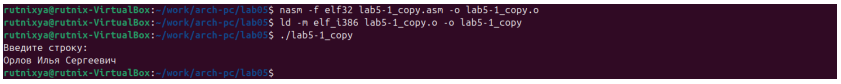

Рис. 4.13: Запуск изменной программы с другой подпрограммой

Создаю копию lab5-1.asm, редактирую так, чтобы в конце выводилась введеная мною строка с клавиатуры (рис. -fig. 4.14).

Рис. 4.14: Редактирование копии

Транслирую, компоную и запускаю свою программу (рис. -fig. 4.15).

Рис. 4.15: Запуск своей программы

Код прикладываю

```
ECTION .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
    mov     ecx, buf1
    mov     edx, 80
    int     80h
    mov     eax, 4
    mov     ebx, 1
    mov     ecx, buf1
    mov     edx, buf1
    int     80h
    mov     eax, 1
    mov     ebx, 0
    int     80h
```

Создаю копию lab5-2.asm, редактирую так, чтобы в конце выводилась введеная мною строка с клавиатуры (рис. -fig. 4.16).

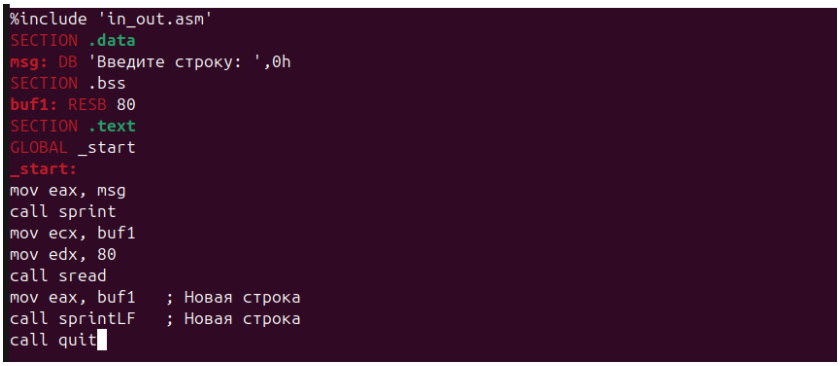

Рис. 4.16: Редактирование копии

Транслирую, компоную и запускаю свою программу (рис. -fig. 4.17).

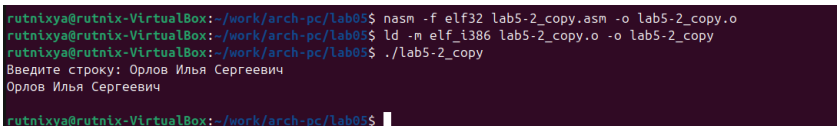

Рис. 4.17: Запуск своей программы

Код прикладываю:

```
%include 'in_out.asm'
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициированных 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
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
int 80h ; Вызов ядра
call quit ; вызов подпрограммы завершения
```

5 Выводы

При выполнении данной лабораторной работы я приобрёл практические навыки работы в Midnight Commander, а также освоил инструкции языка ассемблера mov и int.

6 Выводы

При выполнении данной лабораторной работы я приобрёл практические навыки работы в Midnight Commander, а также освоил инструкции языка ассемблера mov и int.

Список литературы

1. Пример выполнения лабораторной работы
2. Курс на ТУИС
3. Лабораторная работа №5
4. Программирование на языке ассемблера NASM Столяров А. В.