

Отчёт по лабораторной работе

Дисциплина: архитектура компьютера

Шакиров Индус Равилевич

Содержание

1 Цель работы

Целью данной лабораторной работы является приобретение практических навыков работы в Midnight Commander, освоение инструкций языка ассемблера `mov n 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 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

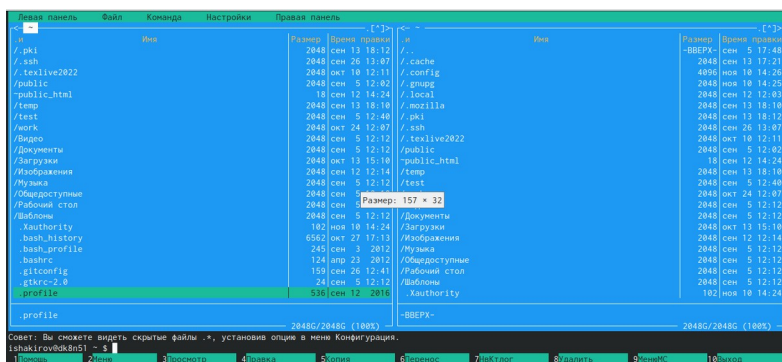
байт (учетверённое слово); • DT (define ten bytes) — определяет переменную размером в 10 байт.

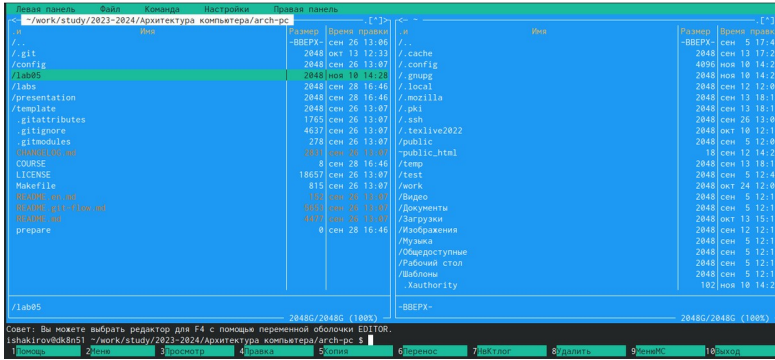
Директивы используются для объявления простых переменных и для объявления массивов. Для определения строк принято использовать директиву DB в связи с особенностями хранения данных в оперативной памяти.

Для объявления инициализированных данных в секции .bss используются директивы resb, resw, resd и другие, которые сообщают ассемблеру, что необходимо зарезервировать за данное количество ячеек памяти.

4 Выполнение лабораторной работы

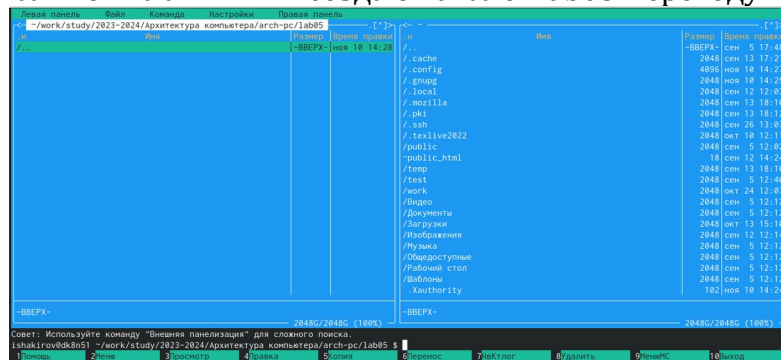
Открываю Midnight Commander, введя в терминал mc





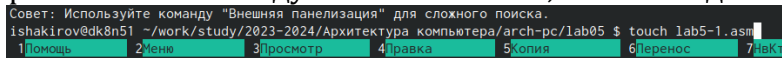
Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab05 Переходу в

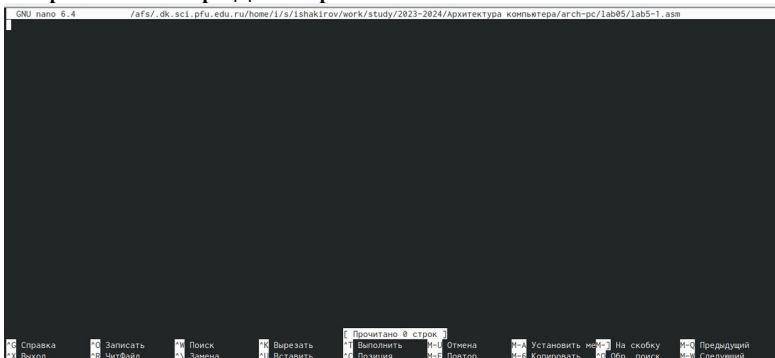


созданный каталог

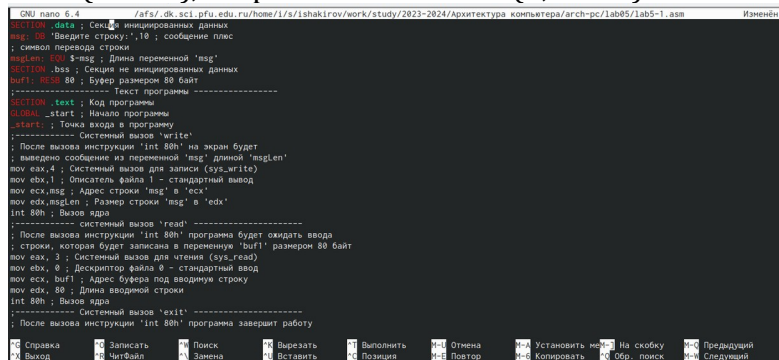
В строке ввода прописываю команду touch lab5-1.asm, чтобы создать файл, в котором буду работать



С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе nano

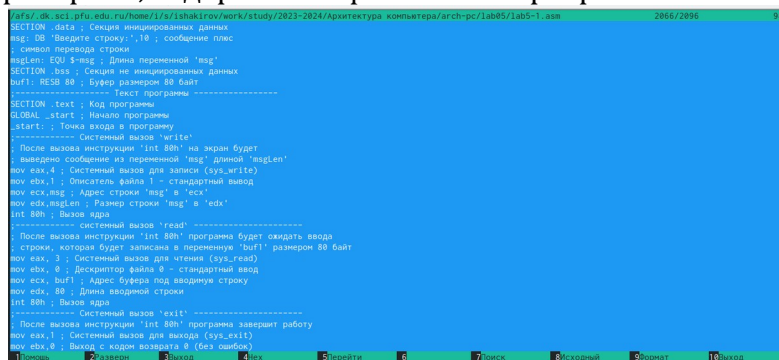


Ввожу в файл код программы для запроса строки у пользователя.Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter).

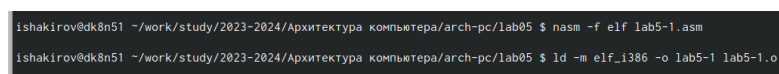


```
GNU nano 6.4 /afs/dk.scf.ru.edu.ru/home/L/s/ishakirov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05/lab5-1.asm
;SECTION .data; Секция инициализированных данных
;msg: DB "Введите строку:",10 ; сообщение плюс
; символ перевода строки
;msglen: EQU $-msg ; длина переменной 'msg'
SECTION .bss ; Секция не инициализированных данных
;buf1: RESB 80 ; Буфер размером 80 байт
;-----
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 входную строку
mov edx,80 ; Длина входной строки
int 80h ; Вызов ядра
;----- Системный вызов 'exit'
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,0 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
```

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы



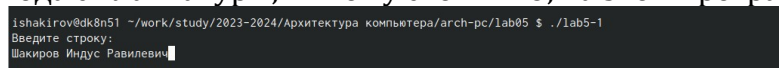
Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm`. Создался объектный файл `lab5-1.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-1 lab5-1.o`. Создался исполняемый файл `lab5-1`.



```
ishakirov@dk8n51 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ nasm -f elf lab5-1.asm
ishakirov@dk8n51 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
```

Компиляция файла

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу



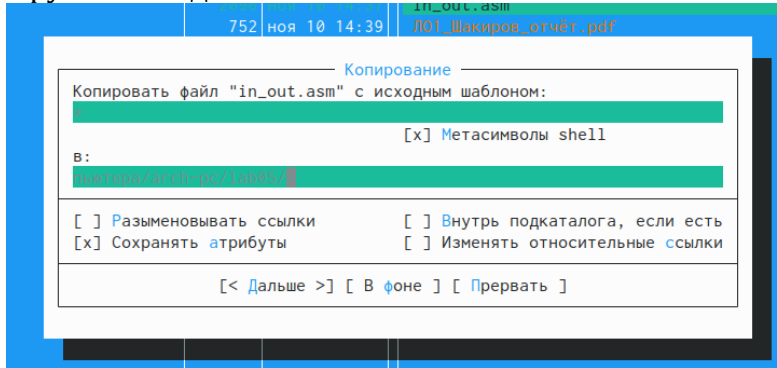
```
ishakirov@dk8n51 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ./lab5-1
Введите строку:
Шакиров Индус Равилевич
```

Скачиваю файл `in out.asm` со страницы курса в ТУИС. Он сохранился в каталог

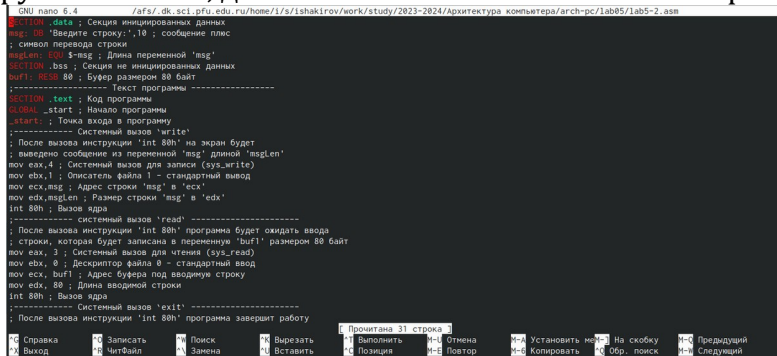
“Загрузки”

Имя	Размер	Время правки
in out.asm	47136	ноя 10 14:24
001_Шакиров_отчёт.pdf	1585936	сен 29 15:46

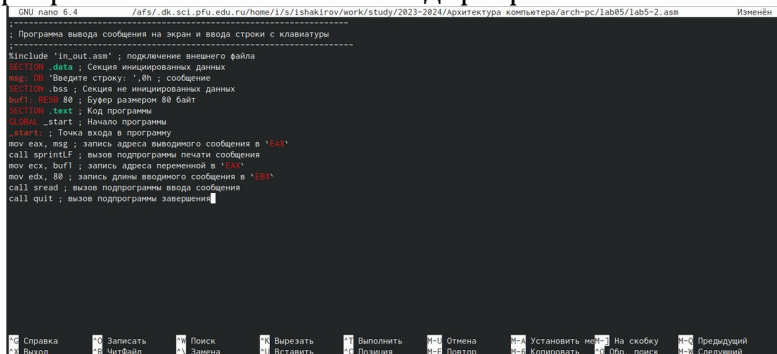
С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога Загрузки в созданный каталог lab05



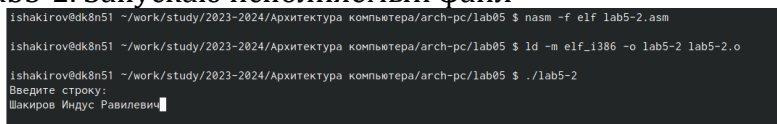
С помощью функциональной клавиши F5 копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне тс прописываю имя для копии файла



Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.



Транслирую текст программы файла в объектный файл командой nasm -f elf lab5-2.asm. Создался объектный файл lab5-2.o. Выполняю компоновку объектного файла с помощью команды ld -m elf_i386 -o lab5-2 lab5-2.o Создался исполняемый файл lab5-2. Запускаю исполняемый файл



Открываю файл lab5-2.asm для редактирования в папо функциональной клавишей F4. Изменяю в нем подпрограмму sprintf.F на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий

```
lab5-2.asm      [-M--] 11 L:[ 1+12 13/ 18] *(847 /1223b) 0032 0x020
; 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 'EDX'
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
```

Снова транслирую файл, выполняю компоновку созданного объектного файла, запуская новый исполняемый файл

```
ishakirov@dk8n51 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ nasm -f elf lab5-2.asm
ishakirov@dk8n51 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
ishakirov@dk8n51 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ./lab5-2
Введите строку: Шакиров Индус Равилович
```

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая выполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами sprintfLF и sprint.

5 Выводы

При выполнении данной лабораторной работы я приобрел практические навыки работы в midnight commander, а также освоил инструкции языка ассемблера mov и int.

Список литературы

1. Лабораторная работа №5