

Отчет по лабораторной работе №6

Архитектура компьютера

Овчинников Антон Григорьевич

Содержание

| | | |
|----------|--|-----------|
| 1 | Цель работы | 5 |
| 2 | Задание | 6 |
| 3 | Теоретическое введение | 7 |
| 4 | Выполнение лабораторной работы | 8 |
| 5 | Выполнение заданий для самостоятельной работы | 11 |
| 6 | Листинги | 13 |
| 7 | Выводы | 15 |

Список иллюстраций

| | | |
|-----|--------------------------------------|----|
| 4.1 | Каталог lab05 | 8 |
| 4.2 | Файл lab5-1 | 8 |
| 4.3 | Запуск программы | 9 |
| 4.4 | Скопированный файл | 9 |
| 4.5 | Измененный текст программы | 9 |
| 4.6 | Запуск программы | 10 |
| 5.1 | Изменение программы | 11 |
| 5.2 | Запуск программы | 12 |
| 5.3 | Изменение программы | 12 |
| 5.4 | Запуск программы | 12 |

Список таблиц

1 Цель работы

Целью работы является приобретение практических навыков работы в Midnight Commander. А также освоение инструкций языка ассемблера `mov` и `int`.

2 Задание

Сделать отчет

3 Теоретическое введение

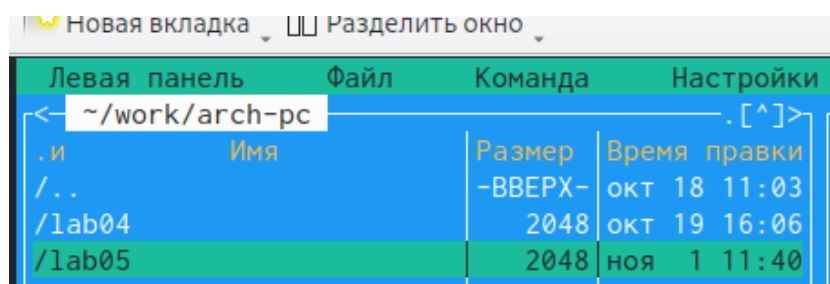
Midnight Commander (или просто mc) — это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системой, т.е. mc является файловым менеджером. Midnight Commander позволяет сделать работу с файлами более удобной и наглядной. Программа на языке 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 (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.

4 Выполнение лабораторной работы

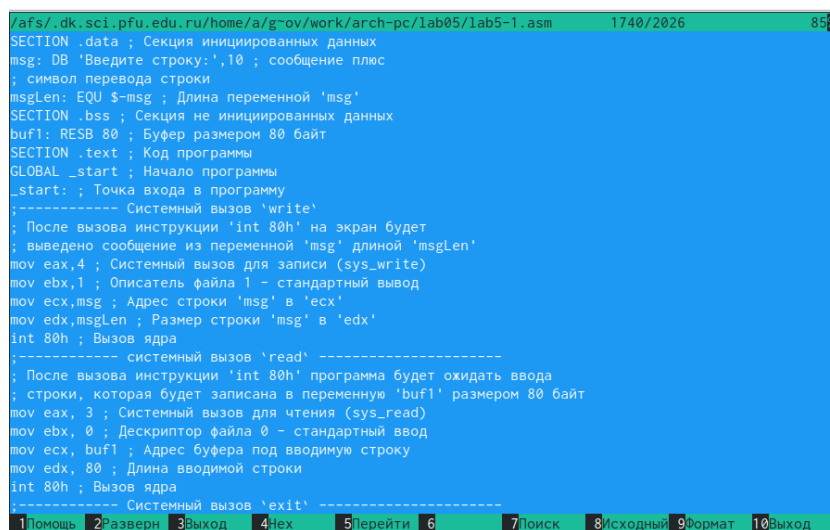
Создаю каталог lab05 (рис. 4.1).



| Левая панель | Файл | Команда | Настройки |
|----------------|--------|---------|--------------|
| ~/work/arch-pc | | | |
| | .и | Имя | Размер |
| | /.. | -ВВЕРХ- | Время правки |
| | /lab04 | 2048 | окт 18 11:03 |
| | /lab05 | 2048 | окт 19 16:06 |
| | | | ноя 1 11:40 |

Рис. 4.1: Каталог lab05

Ввожу текст из листинга 5.1 в файл lab5-1 (рис. 4.2)



```
/afs/.dk.sci.pfu.edu.ru/home/a/g-ov/work/arch-pc/lab05/lab5-1.asm 1740/2026 85%
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных 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 вводимую строку
mov edx,80 ; Длина вводимой строки
int 80h ; Вызов ядра
;----- Системный вызов 'exit' -----
1Помощь 2Разверн 3Выход 4Нех 5Перейти 6 7Поиск 8Исходный 9Формат 10Выход
```

Рис. 4.2: Файл lab5-1

Выполняю компоновку объектного файла и запускаю программу (рис. 4.3)


```

agovchinnikov@dk8n70 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
agovchinnikov@dk8n70 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Овчинников Антон Григорьевич
agovchinnikov@dk8n70 ~/work/arch-pc/lab05 $

```

Рис. 4.3: Запуск программы

Создаю копию файла lab5-1.asm с именем lab5-2.asm (рис. 4.4)

| Имя | Размер | Время правки |
|-----------------|---------|--------------|
| ../ | -ВВЕРХ- | ноя 1 11:24 |
| in_out.asm | 3942 | ноя 8 16:02 |
| *lab5-1 | 8744 | ноя 1 11:59 |
| lab5-1.asm | 2026 | ноя 1 11:48 |
| lab5-1.o | 752 | ноя 1 11:55 |
| lab5-2.asm | 2026 | ноя 1 11:48 |
| lab5-2.asm.save | 2027 | ноя 8 16:09 |

Рис. 4.4: Скопированный файл

Изменяю текст программы lab5-2.asm, чтобы в ней использовались функции из подключаемого файла (рис. 4.5)

```

lab5-2.asm  [-M--] 11 L:[ 1+ 9 10/ 14] *(586 / 960b) 0032 0x020
#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 'EBX'
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения

```

Рис. 4.5: Измененный текст программы

Создаю объектный файл lab5-2.o, выполняю компоновку объектного файла и запускаю исполняемый файл (рис. ??). Из-за смены подпрограммы sprintf на

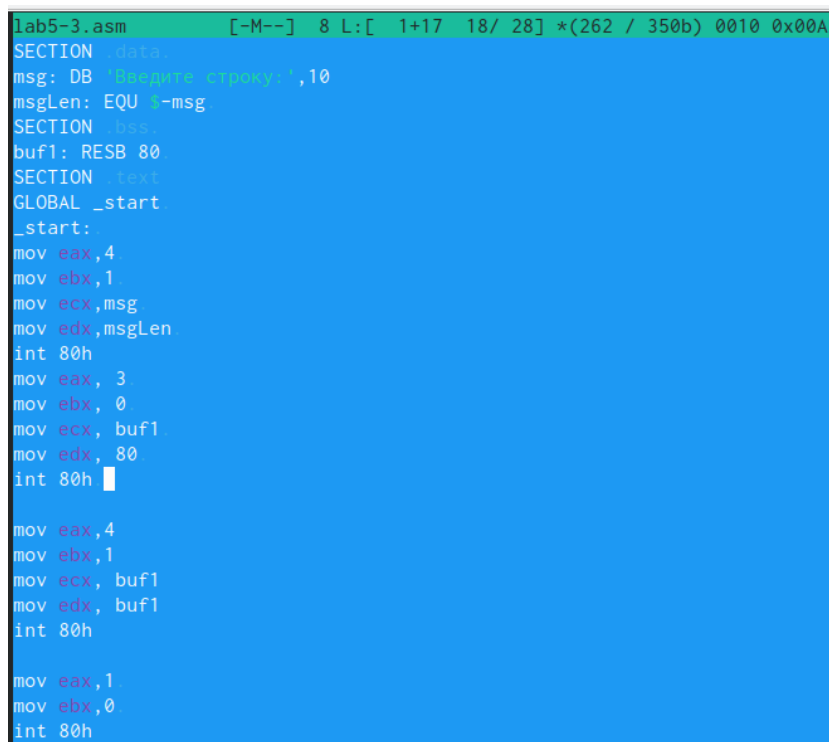
sprint ввод производится на той же строке, что и вывод, убран символ перевода строки после вывода.

```
agovchinnikov@dk8n75 ~ $ cd
agovchinnikov@dk8n75 ~ $ cd work/arch-pc/lab05
agovchinnikov@dk8n75 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
agovchinnikov@dk8n75 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
agovchinnikov@dk8n75 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку: Антон Овчинников
```

Рис. 4.6: Запуск программы

5 Выполнение заданий для самостоятельной работы

Скопировал файл lab5-1.asm с именем lab5-3.asm, а затем изменяю код программы, добавляя вывод введенной строки (рис. 5.1)



```
lab5-3.asm      [-M--]  8 L:[ 1+17 18/ 28] *(262 / 350b) 0010 0x00A
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
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
```

Рис. 5.1: Изменение программы

Создаю объектный файл lab5-3.o, компилирую его в исполняемый файл (рис. 5.2)

```

agovchinnikov@dk8n75 ~/work/arch-pc/lab05 $ nasm -f elf lab5-3.asm
agovchinnikov@dk8n75 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-3 lab5-3.o
agovchinnikov@dk8n75 ~/work/arch-pc/lab05 $ ./lab5-3
Введите строку:
Антон Овчинников
Антон Овчинников
agovchinnikov@dk8n75 ~/work/arch-pc/lab05 $

```

Рис. 5.2: Запуск программы

Скопировал файл lab5-2.asm с именем lab5-4.asm, а затем изменяю код программы, добавляя вывод введенной строки (рис. 5.3)

```

lab5-4.asm [----] 29 L: [ 1+ 2 3/ 24] *(78 / 308b) 0010 0x00A
#include "lab5-2.asm"
SECTION .data
msg: DB "Введите строку: ",0h

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

    call    quit

```

Рис. 5.3: Изменение программы

Создаю объектный файл lab5-4.o, компонирую его в исполняемый файл (рис. ??)

```

agovchinnikov@dk2n24 ~/work/arch-pc/lab05 $ nasm -f elf lab5-4.asm
agovchinnikov@dk2n24 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-4 lab5-4.o
agovchinnikov@dk2n24 ~/work/arch-pc/lab05 $ ./lab5-4
Введите строку: Антон Овчинников
Антон Овчинников
agovchinnikov@dk2n24 ~/work/arch-pc/lab05 $

```

Рис. 5.4: Запуск программы

6 Листинги

```
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
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 `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

%include 'in_out.asm' ; подключение внешнего файла
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 sprintLF ; 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
call quit ; вызов подпрограммы завершения

```

7 Выводы

Я приобрел практические навыки работы в Midnight Commander и освоил инструкции языка ассемблера `mov` и `int`