Отчет по лабораторной работе №6

Архитектура компьютера

Овчинников Антон Григорьевич

Содержание

# 1 Цель работы

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

# 2 Задание

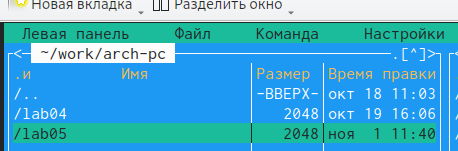
Сделать отчет

# 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 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 (учетверённое слово); • DT (define ten bytes) — определяет переменную размером в 10 байт Директивы используются для объявления простых переменных и для объявления масси- вов. Для определения строк принято использовать директиву DB в связи с особенностями хранения данных в оперативной памяти.

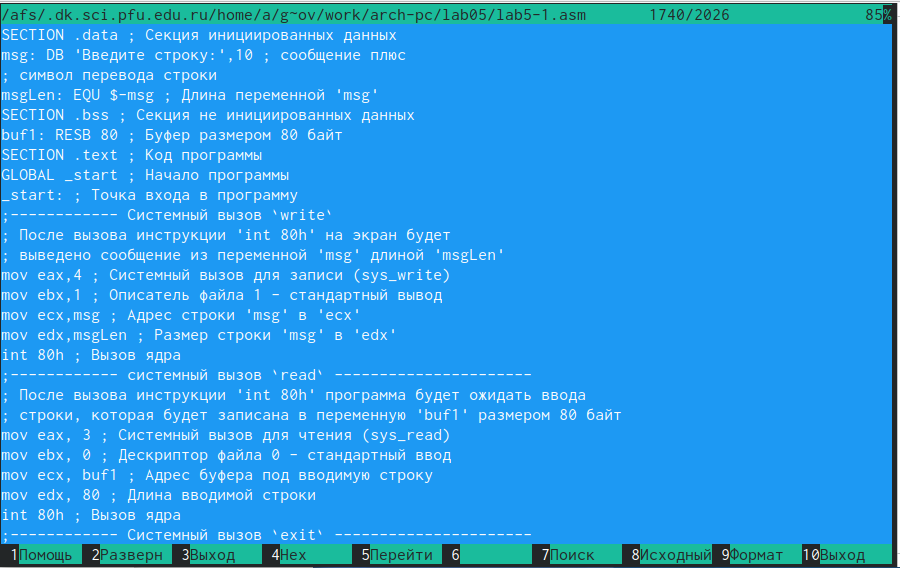
# 4 Выполнение лабораторной работы

Создаю каталог lab05 (рис. ??).



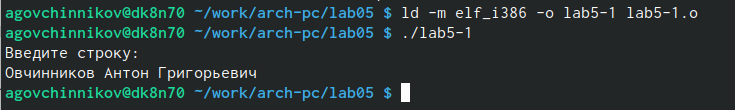
Каталог lab05

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



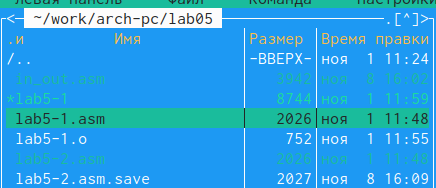
Файл lab5-1

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



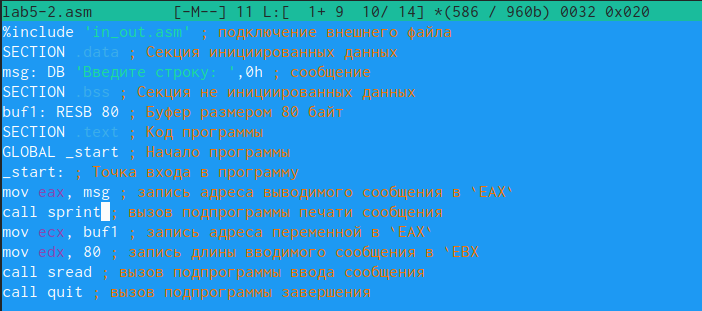
Запуск программы

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



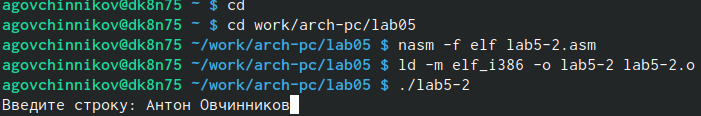
Скопированный файл

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



Измененный текст программы

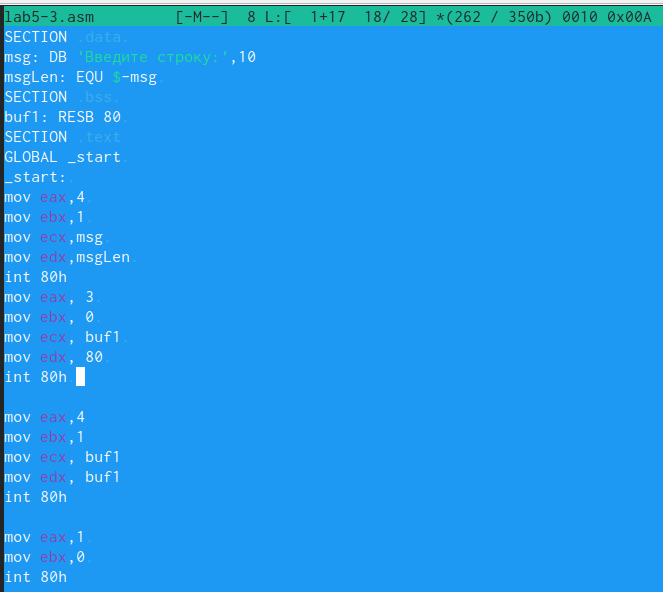
Создаю объектный файл lab5-2.o, выполняю компоновку объектного файла и запускаю исполняемый файл (рис. ??) .Из-за смены подпрограммы sprintlF на sprint ввод производится на той же строке,что и вывод,убран символ перевода строки после вывода.



Запуск программы

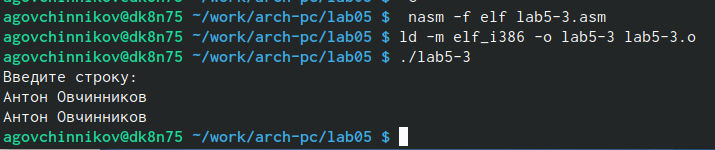
# 5 Выполнение заданий для самостоятельной работы

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



Изменение программы

Создаю объектный файл lab5-3.o, компонипую его в исполняемый файл (рис. ??)



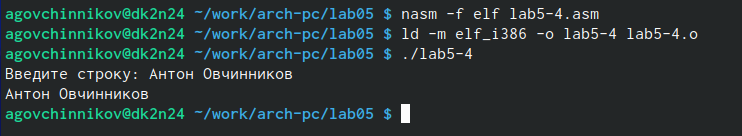
Запуск программы

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



Изменение программы

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



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  
;------------ Cистемный 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