Лабораторная работа №5

. Основы работы с Midnight Commander (mc). Структура программы на языке ассемблера NASM. Системные вызовы в ОС GNU Linux

Бунин Арсений Викторович

Содержание

1 Цель работы **5**

2 Задание **6**

3 Теоретическое введение **7**

4 Выполнение лабораторной работы **8**

5 Самостоятельная работа **13**

6 Выводы **17**

Список литературы **18**

Список иллюстраций

4.1 Вызов Midnight Commander 8
4.2 Каталог курса в Midnight Commander 8
4.3 Окно создания папки 9
4.4 Окно создания файла 9
4.5 Файл в редакторе mcedit 9
4.6 Файл в режиме просмотра 10
4.7 Линковка и компоновка 10
4.8 Работа первой программы 10
4.9 Окно замены файла 11
4.10 Код второй программы 11
4.11 Вторая программа 11
4.12 Измененная вторая программа 12

5.1 Третья программа 14
5.2 Третья программа 14
5.3 Третья программа 15
5.4 Четвертая программа 16

Список таблиц

1 Цель работы

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

2 Задание

1. Создать папку и файл на языке Ассемблер в ней с помощью Midnight Commander
2. Создать файл на языке Ассемблер, принимающий на вход строку
3. Загрузить файлы на github

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 (рис. 5.1). В Midnight Commander используются функциональные клавиши F1 — F10 , к которым привязаны часто выполняемые операции

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).

Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике. Инструкция языка ассемблера int предназначена для вызова прерывания с указанным номером.

4 Выполнение лабораторной работы

Открываем терминал и запускаем Midnight Commander(рис. 4.1).

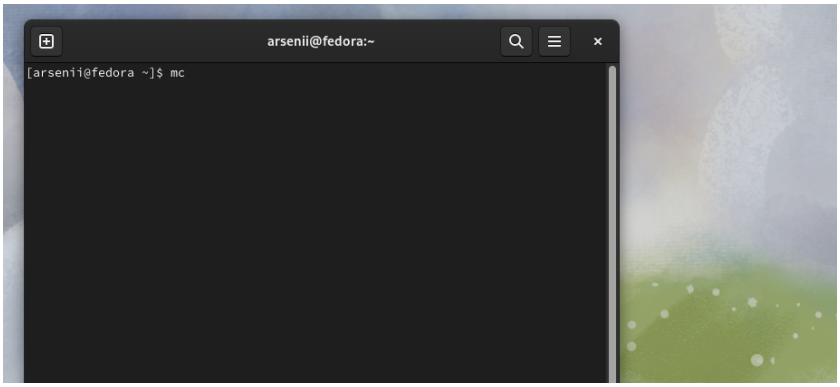

Рис. 4.1: Вызов Midnight Commander

Переходим в каталог курса (рис. 4.2).

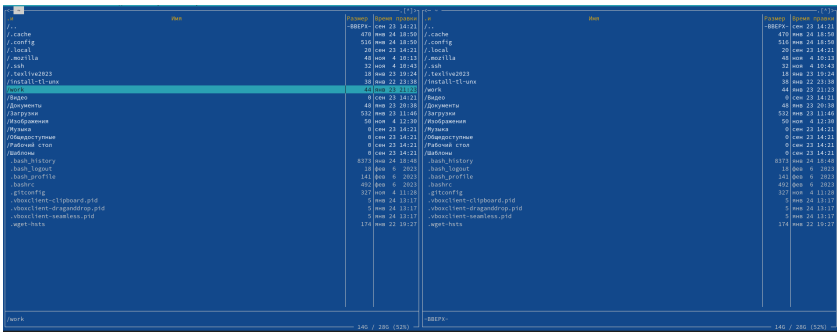

Рис. 4.2: Каталог курса в Midnight Commander

Создаем папку для пятой лабораторной работы (рис. 4.3).

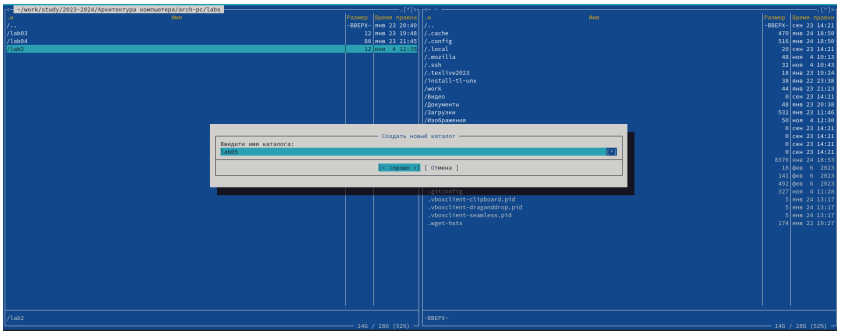

Рис. 4.3: Окно создания папки

Создаем файл lab5-1.asm в новой папке (рис. 4.4).

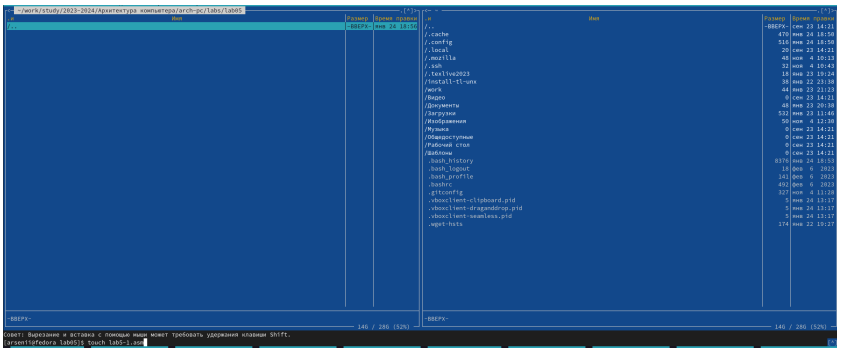

Рис. 4.4: Окно создания файла

Записываем код в файл в редакторе mcedit (рис. 4.5)

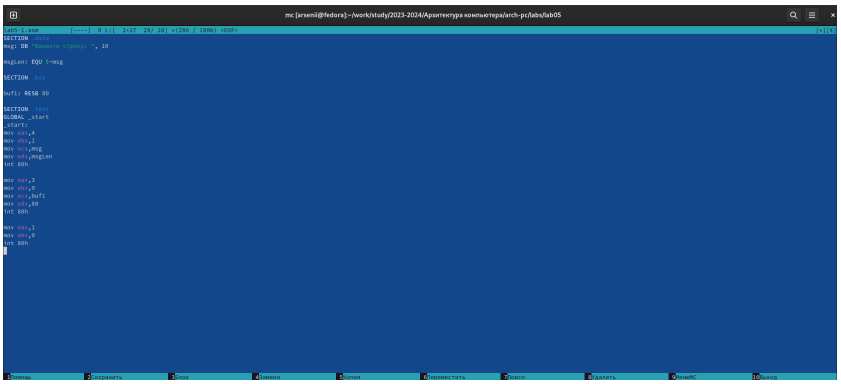

Рис. 4.5: Файл в редакторе mcedit

Проверяем файл в режиме просмотра (рис. 4.6)

Рис. 4.6: Файл в режиме просмотра

Выполняем линковку и компоновку (рис. 4.7)

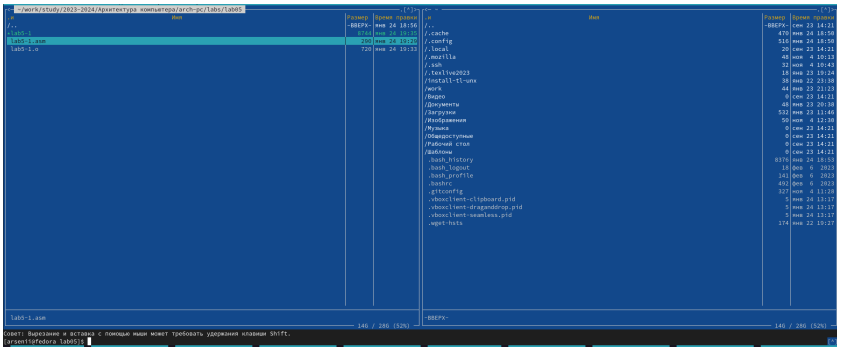

Рис. 4.7: Линковка и компоновка

Результат работы программы(рис. 4.8)

Рис. 4.8: Работа первой программы

Переименовываем файл в lab5-1.asm(рис. 4.9)

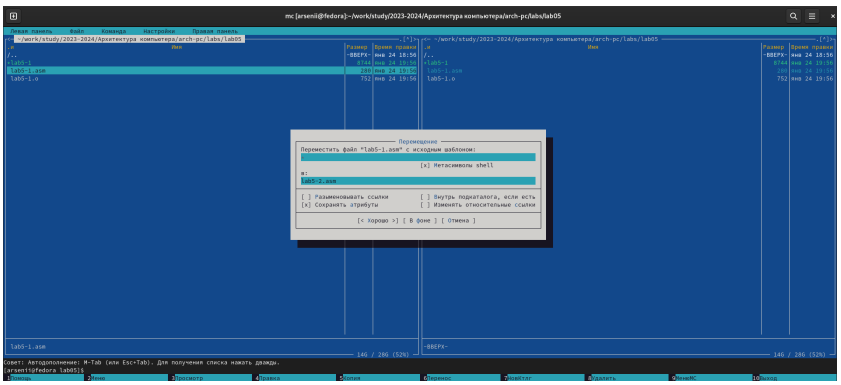

Рис. 4.9: Окно замены файла

Записываем код второй программы в файл(рис. 4.10)

Рис. 4.10: Код второй программы

Результат работы второй программы(рис. 4.11)

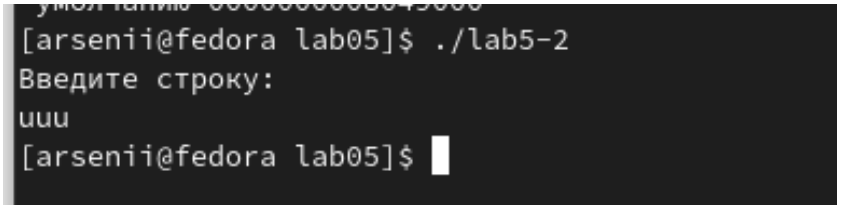

Рис. 4.11: Вторая программа

Результат работы измененной второй программы. Теперь ввод текста происходит на строке с текстовым сообщением, а не на новой строке(рис. 4.12)

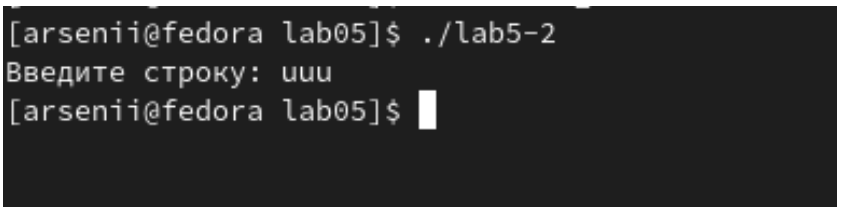

Рис. 4.12: Измененная вторая программа

5 Самостоятельная работа

1. Создайте копию файла lab5-1.asm. Внесите изменения в программу (без использования внешнего файла in_out.asm), так чтобы она работала по следующему алгоритму: • вывести приглашение типа “Введите строку:”; • ввести строку с клавиатуры; • вывести введённую строку на экран.
2. Получите исполняемый файл и проверьте его работу. На приглашение ввести строку введите свою фамилию.

Текст третьей программы (рис. 5.1)

```
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
;------------ вывод строки
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
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
```

Рис. 5.1: Третья программа

Результат работы третьей программы (рис. 5.2)

```
[arsenii@fedora lab05]$ nasm -f elf lab5-3.asm
[arsenii@fedora lab05]$ ld -m elf_i386 lab5-3.o -o lab5-3
[arsenii@fedora lab05]$ ./lab5-3
Введите строку:
Bunin
Bunin
[arsenii@fedora lab05]$
```

Рис. 5.2: Третья программа

3. Создайте копию файла lab5-2.asm. Исправьте текст программы с использо-

вание подпрограмм из внешнего файла in_out.asm, так чтобы она работала по следующему алгоритму: • вывести приглашение типа “Введите строку:”; • ввести строку с клавиатуры; • вывести введённую строку на экран. Не забудьте, подключаемый файл in_out.asm должен лежать в том же каталоге, что и файл с программой, в которой он используется.

4. Создайте исполняемый файл и проверьте его работу.

Текст четвертой программы (рис. 5.3)

```
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, buf1
mov edx, 80
call sread
mov eax, buf1
call sprint
call quit
```

Рис. 5.3: Третья программа

Результат работы четвертой программы (рис. 5.4)

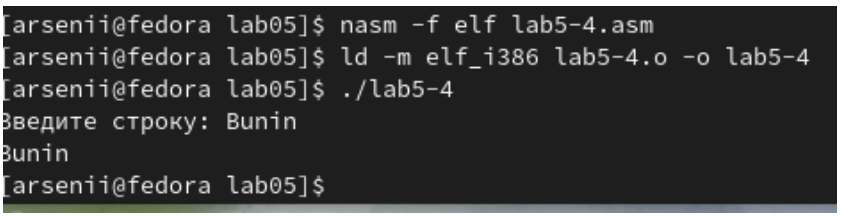

Рис. 5.4: Четвертая программа

6 Выводы

Освоили работу с Midnight Commander и научились использовать команды int и mov языка Assembler

Список литературы