

Отчёт по лабораторной работе №5

Дисциплина: Архитектура компьютера

Абрамова Ульяна Михайловна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Основы работы с mc	9
4.2	Структура программы на языке ассамблера NASM	9
4.3	Подключение внешнего файла	10
4.4	Выполнение заданий для самостоятельной работы	12
5	Выводы	15
6	Список литературы	16

Список иллюстраций

4.1	Создание файла в каталоге	9
4.2	Редактирование файла	10
4.3	Компоновка файла и запуск программы	10
4.4	Скачивание и перемещение файла in_out.asm	11
4.5	Создание нового файла копированием и изменение его содержимого	11
4.6	Компоновка файла и запуск программы	11
4.7	Изменение подпрограммы, компоновка и запуск команды	12
4.8	Изменение программы	12
4.9	Компонирование файла и его запуск	13
4.10	Изменение программы	13
4.11	Компонирование файла и его запуск	14

List of Tables

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 ассамблера 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 (четверённое 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 `int` предназначена 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 mc

Открываю Midnight Commander, введя в терминал `mc`. Перехожу в каталог `~/work/study/2022-2023/Архитектура Компьютера/arch-pc`, используя файловый менеджер `mc`. С помощью функциональной клавиши `F7` создаю каталог `lab05` и перехожу в него, создав файл для дальнейшей работы с помощью команды `touch lab5-1.asm` (рис. 4.1)

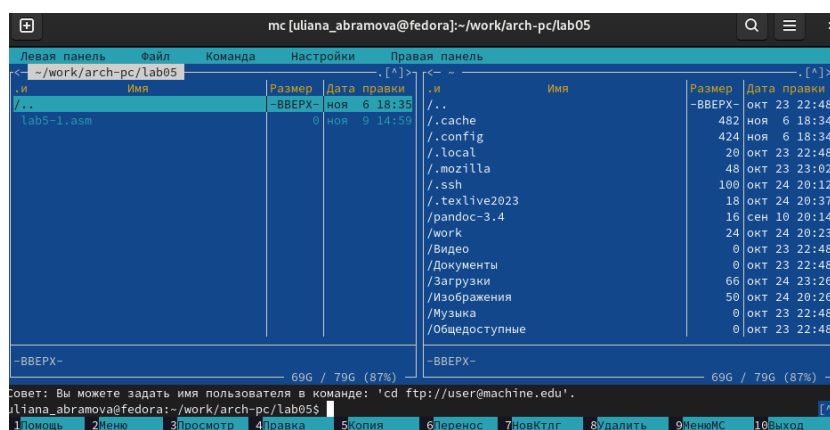
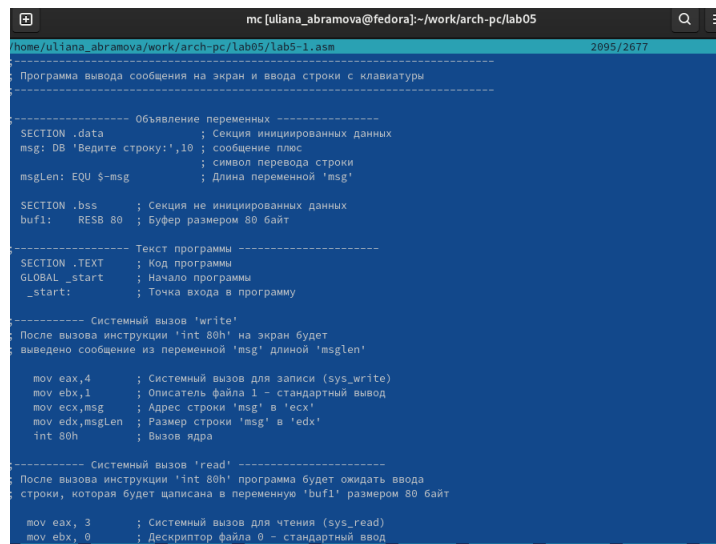


Рис. 4.1: Создание файла в каталоге

4.2 Структура программы на языке ассемблера NASM

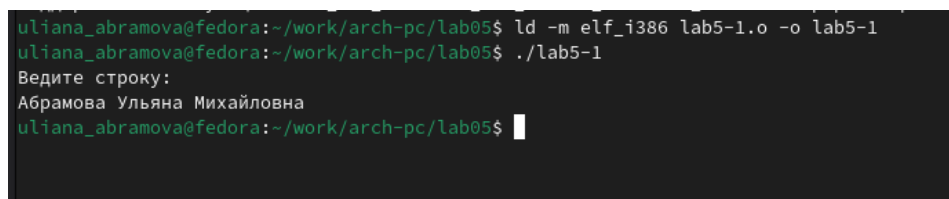
С помощью функциональной клавиши `F4` открываю созданный файл для редактирования и ввожу в него код программы для запроса строки у пользователя. (рис. 4.2)



```
mc [uliana_abramova@fedora:]-/work/arch-pc/lab05
home/uliana_abramova/work/arch-pc/lab05/lab5-1.asm 2095/2677
-----
Программа вывода сообщения на экран и ввода строки с клавиатуры
-----
Объявление переменных
SECTION .data ; Секция инициализированных данных
msg: DB 'Ведите 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
-----
Текст программы
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
```

Рис. 4.2: Редактирование файла

Оттранслирую текст программы lab5-1.asm в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл. (рис. 4.3)



```
uliana_abramova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-1.o -o lab5-1
uliana_abramova@fedora:~/work/arch-pc/lab05$ ./lab5-1
Ведите строку:
Абрамова Ульяна Михайловна
uliana_abramova@fedora:~/work/arch-pc/lab05$
```

Рис. 4.3: Компоновка файла и запуск программы

4.3 Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС и из каталога Загрузки копирую в созданный каталог lab05 (рис. 4.4)

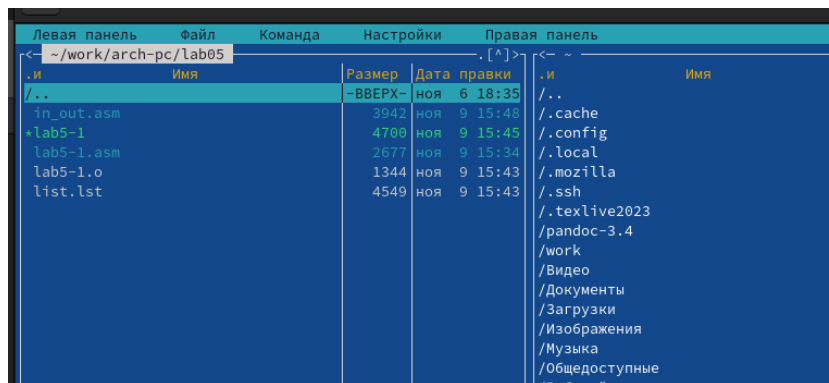


Рис. 4.4: Скачивание и перемещение файла in_out.asm

С помощью функциональной клавиши F5 копирую файл lab5-1.asm в тот же каталог, но с именем lab5-2.asm, и изменяю содержимое, чтобы в программе использовались подпрограммы из внешнего файла in_out.asm. (рис. 4.5)

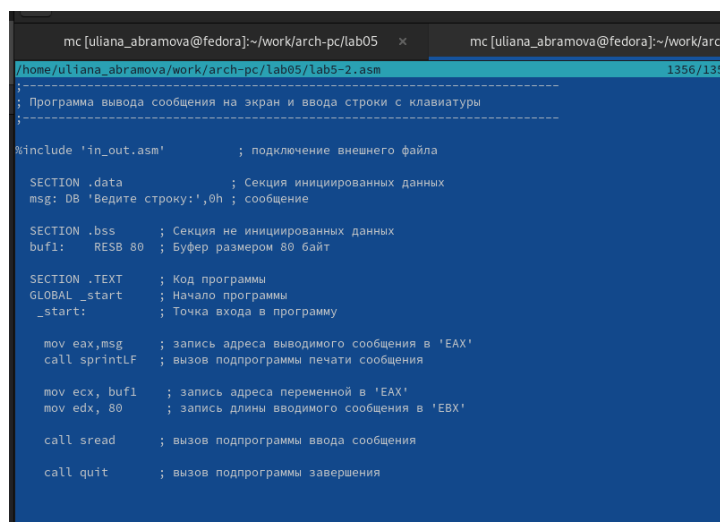


Рис. 4.5: Создание нового файла копированием и изменение его содержимого

Оттранслирую текст программы lab5-2.asm в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл. (рис. 4.6)

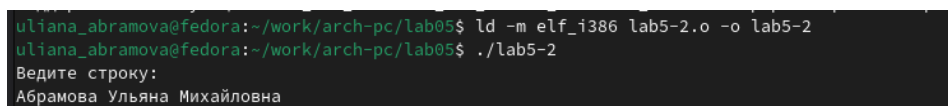


Рис. 4.6: Компоновка файла и запуск программы

Открываю файл lab6-2.asm для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и снова проделываю компоновку файла. (рис 4.7)

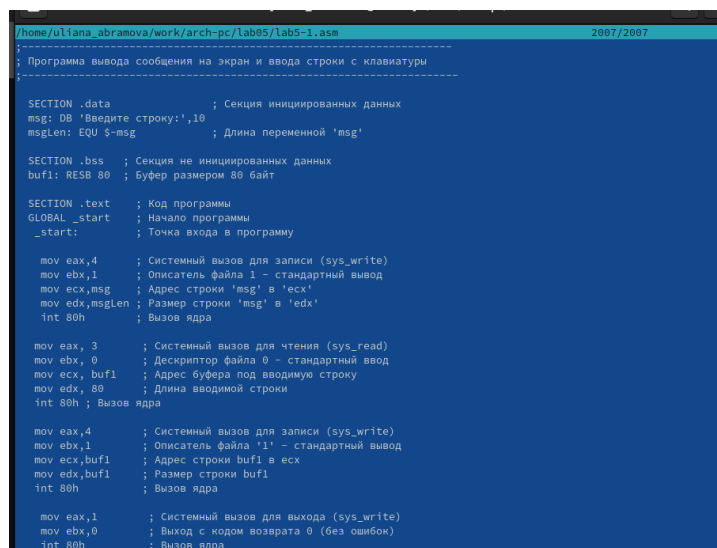
```
uliana_abramova@fedora:~/work/arch-pc/lab05$ ./lab5-2
Ведите строку: Абрамова Ульяна Михайловна
```

Рис. 4.7: Изменение подпрограммы, компоновка и запуск команды

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

4.4 Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5. С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку. (рис. 4.8)



```
/home/uliana_abramova/work/arch-pc/lab05/lab5-1-1.asm 2007/2007
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10
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

mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описание файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки 'msg' в 'ecx'
mov edx,msgLen ; Размер строки 'msg' в 'edx'
int 80h ; Вызов ядра

mov eax,3 ; Системный вызов для чтения (sys_read)
mov ebx,0 ; Описание файла 0 - стандартный ввод
mov ecx,buf1 ; Адрес буфера под вводимую строку
mov edx,80 ; Длина вводимой строки
int 80h ; Вызов ядра

mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описание файла 1 - стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в ecx
mov edx,buf1 ; Размер строки buf1
int 80h ; Вызов ядра

mov eax,1 ; Системный вызов для выхода (sys_write)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
```

Рис. 4.8: Изменение программы

Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные. (рис. 4.9)

```
uliana_abramova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-1-1.o -o lab5-1-1
uliana_abramova@fedora:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Абрамова Ульяна Михайловна
Абрамова Ульяна Михайловна
```

Рис. 4.9: Компонование файла и его запуск

2. Создаю копию файла lab5-2.asm с именем lab5-2-2.asm с помощью функциональной клавиши F5. С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку. (рис. 4.10)

```
/home/uliana_abramova/work/arch-pc/lab05/lab5-2-2.asm 1631/1631
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----

%include 'in_out.asm'      ; подключение внешнего файла

SECTION .data              ; Секция иницированных данных
msg: DB 'Введите строку:',0h ; сообщение

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

    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
```

Рис. 4.10: Изменение программы

Создаю объектный файл lab5-2-2.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-2, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные. (рис. 4.11)

```
uliana_abramova@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-2-2.o -o lab5-2-2
uliana_abramova@fedora:~/work/arch-pc/lab05$ ./lab5-2-2
Ведите строку: Абрамова
Абрамова
```

Рис. 4.11: Компонирование файла и его запуск

5 Выводы

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

6 Список литературы

1. Архитектура ЭВМ