

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: *Архитектура компьютера*

Студент: Магомед Ужахов.

Группа: НПИбд-02-22

МОСКВА

2022 г.

Цель работы:

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

Порядок выполнения лабораторной работы:

Откроем Midnight Commander и перейдем в каталог ~/work/arch-pc созданный при выполнении лабораторной работы №5

Затем создадим папку lab06 и создадим в ней файл lab6-1.asm

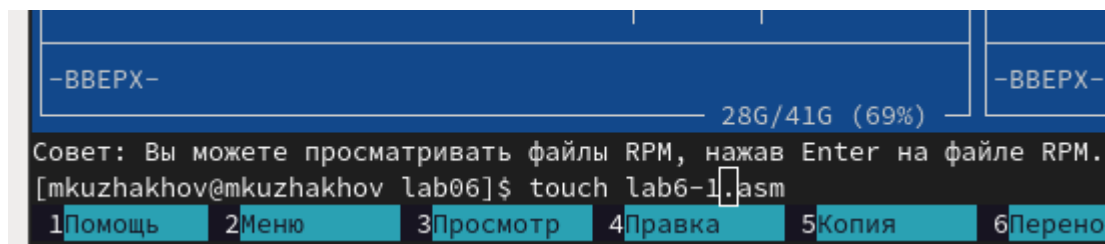
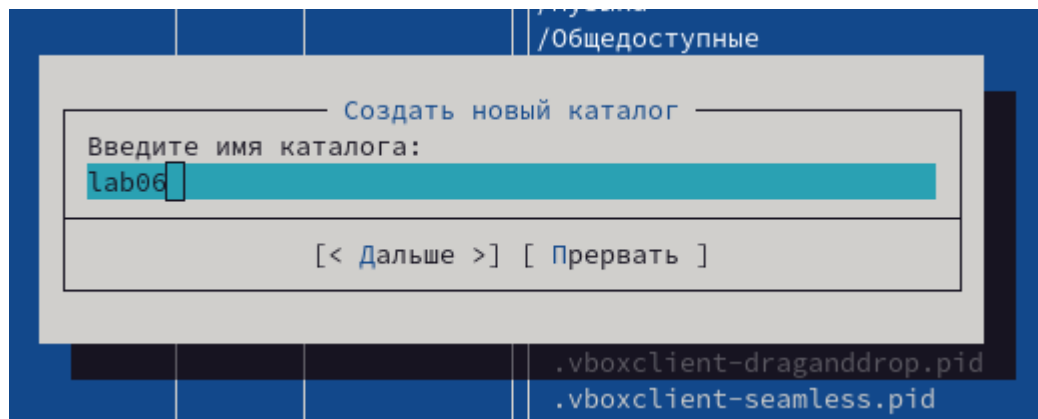


Рис. 1. Создание папки lab06 и файла lab6-1.asm

Потом откроем файл lab6-1.asm и введем следующий текст программы

```
mc [mkuzhakhov@mkuzhakhov]:~/work/arch-pc/lab06
lab6-1.asm [----] 56 L:[ 1+11 12/ 29] *(743 /1801b) 0010 0x00A
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ;
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных 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
mov ecx, buf1 ; Адрес буфера под вводимую строку
mov edx, 80 ; Длина вводимой строки
int 80h
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h
```

Рис. 2. Текст программы lab6-1.asm

Затем оттранслируем текст программы lab6-1.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл. Затем на запрос программы введем свои ФИО

```
mkuzhakhov@mkuzhakhov lab06]$ nasm -f elf lab6-1.asm
lab6-1.asm:21: warning: label alone on a line without a colon might
mkuzhakhov@mkuzhakhov lab06]$ ls
lab6-1.asm  lab6-1.o
mkuzhakhov@mkuzhakhov lab06]$ rm lab6-1.o
mkuzhakhov@mkuzhakhov lab06]$ nasm -f elf lab6-1.asm
mkuzhakhov@mkuzhakhov lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
mkuzhakhov@mkuzhakhov lab06]$ ./lab6-1
Введите строку:
mkuzhakhov
mkuzhakhov@mkuzhakhov lab06]$
```

Рис. 3. Программа lab6-1

Скачаем файл `in_out.asm` со страницы курса в ТУИС, чтобы в дальнейшем упростить нашу жизнь, и переместим его в каталог с файлом `lab6-1.asm`

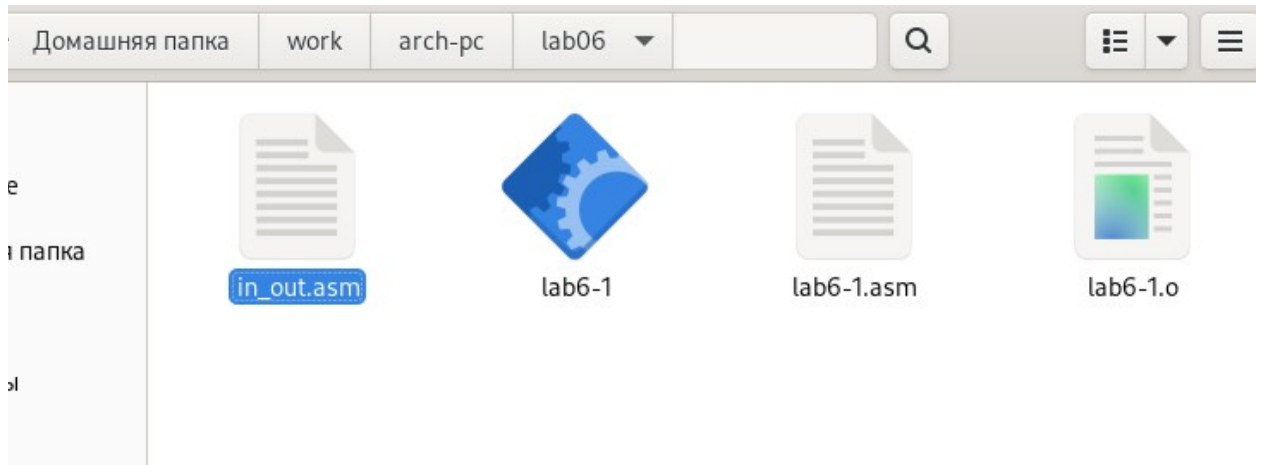


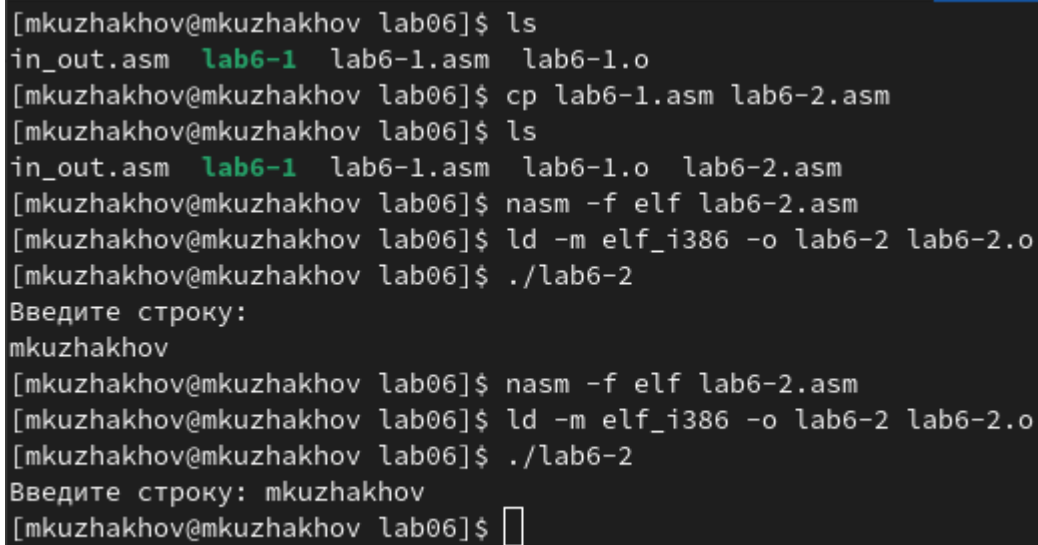
Рис. 4. Копирование файла `in_out.asm`

Создадим копию файла `lab6-1.asm` с именем `lab6-2.asm`. Затем исправим текст программы в файле `lab6-2.asm` с использованием подпрограмм из внешнего файла `in_out.asm`. Создадим исполняемый файл и проверим его работу

```
mc [mkuzhakhov@mkuzhakhov]:~/work/arch-pc/lab06 x
lab6-2.asm [-M--] 20 L: [ 1+ 0 1/ 14] *(20 / 910b)
%include 'in_out.asm'
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 sprintf ; 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. 5. Исправление текста программы

Затем в этом же файле заменим подпрограмму `sprintLF` на `sprint` и затем запустим



```
[mkuzhakhov@mkuzhakhov lab06]$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o
[mkuzhakhov@mkuzhakhov lab06]$ cp lab6-1.asm lab6-2.asm
[mkuzhakhov@mkuzhakhov lab06]$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2.asm
[mkuzhakhov@mkuzhakhov lab06]$ nasm -f elf lab6-2.asm
[mkuzhakhov@mkuzhakhov lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[mkuzhakhov@mkuzhakhov lab06]$ ./lab6-2
Введите строку:
mkuzhakhov
[mkuzhakhov@mkuzhakhov lab06]$ nasm -f elf lab6-2.asm
[mkuzhakhov@mkuzhakhov lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[mkuzhakhov@mkuzhakhov lab06]$ ./lab6-2
Введите строку: mkuzhakhov
[mkuzhakhov@mkuzhakhov lab06]$ █
```

Рис. 6. Запуск программы 2.0

И видим, что по сравнению со `sprintLF` с подпрограммой `sprint` нет переноса строки после вывода строки программы.

Порядок выполнения самостоятельной работы:

Создадим копию файла `lab6-1.asm` и назовем его `lab6-1_1.asm`. Внесем изменения в программу, так чтобы она после нашего ввода выводила напечатанную строку И запустим её

```

lab6-3.asm      [----]  4 L:[ 1+31 32/ 38] *(1681/1846b) 0056 0x0
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10 ;
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициированных 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

    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

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

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

```

Рис. 7. Изменения в программе lab6-1_1.asm

```
[mkuzhakhov@mkuzhakhov lab06]$ nasm -f elf lab6-3.asm
[mkuzhakhov@mkuzhakhov lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3
[mkuzhakhov@mkuzhakhov lab06]$ ./lab6-3
Введите строку:
mkuzhakhov
mkuzhakhov
[mkuzhakhov@mkuzhakhov lab06]$
```

Рис. 8. Работа программы lab6-3

Затем проделаем этот процесс со второй программой .

```
lab6-4.asm      [-M--] 11 L:[ 1+15 16/ 18] *(868 / 9
%include 'in_out.asm'
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 sprintf ; 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, buf1
call sprintf

call quit ; вызов подпрограммы завершения
```

Рис. 9. Изменения в программе lab6-4.asm

```
[mkuzhakhov@mkuzhakhov lab06]$ nasm -f elf lab6-4.asm
[mkuzhakhov@mkuzhakhov lab06]$ ld -m elf_i386 lab6-4.o -o lab6-4
[mkuzhakhov@mkuzhakhov lab06]$ ./lab6-4
Введите строку:
mkuzhakhov
mkuzhakhov
[mkuzhakhov@mkuzhakhov lab06]$
```

Рис. 10. Работа программы lab6-4

Вывод:

Во время выполнения лабораторной работы были приобретены практические навыки работы в Midnight Commander, а также освоены инструкции языка ассемблера `mov` и `int`.