

Отчёт по лабораторной работе %6

Дисциплина: Архитектура компьютеров

Бабенко Роман Игоревич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	13

Список иллюстраций

2.1	Открываем Midnight Comander	6
2.2	Создание файла	7
2.3	Записываем программу в файл	7
2.4	Проверка файла	8
2.5	Выполняем программу	8
2.6	копирование 'in_out.asm'	9
2.7	Создание второго файла	9
2.8	Проверяем работу второго файла.	9
2.9	Вносим изменения в копию первого файла	10
2.10	Проверка вывода	10
2.11	Вносим изменения в копию второго файла	11
2.12	Проверка второго файла	12

Список таблиц

1 Цель работы

Приобрести практические навыки работы в Midnight Commander. Освоить инструкции языка ассемблера `mov` и `int`.

2 Выполнение лабораторной работы

Открываем Midnight Comander и переходим в нужный каталог (рис. 2.1)

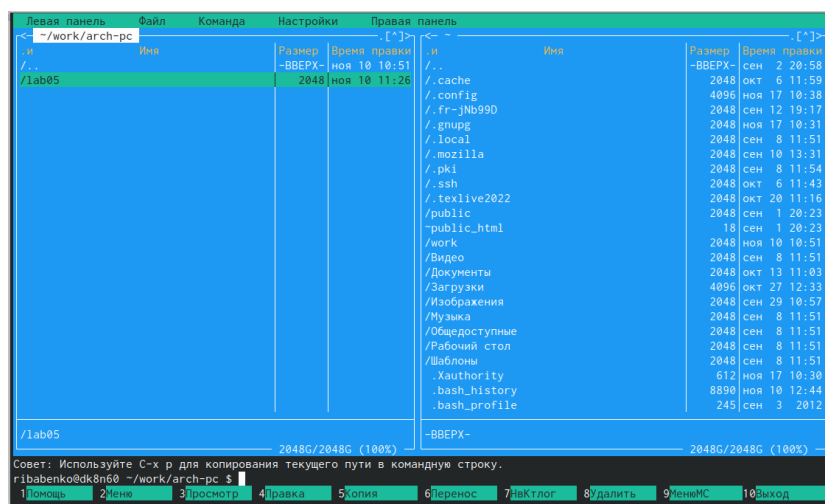


Рис. 2.1: Открываем Midnight Commander

Создаём директорию lab06 и файл lab6-1.asm (рис. 2.2)

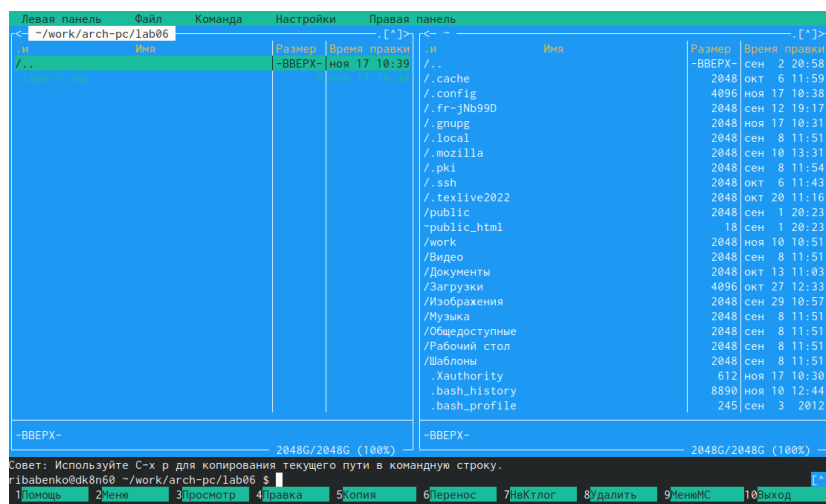


Рис. 2.2: Создание файла

Вводим текст программы из листинга 6.1, сохраняем изменения и закрываем файл (рис. 2.3)

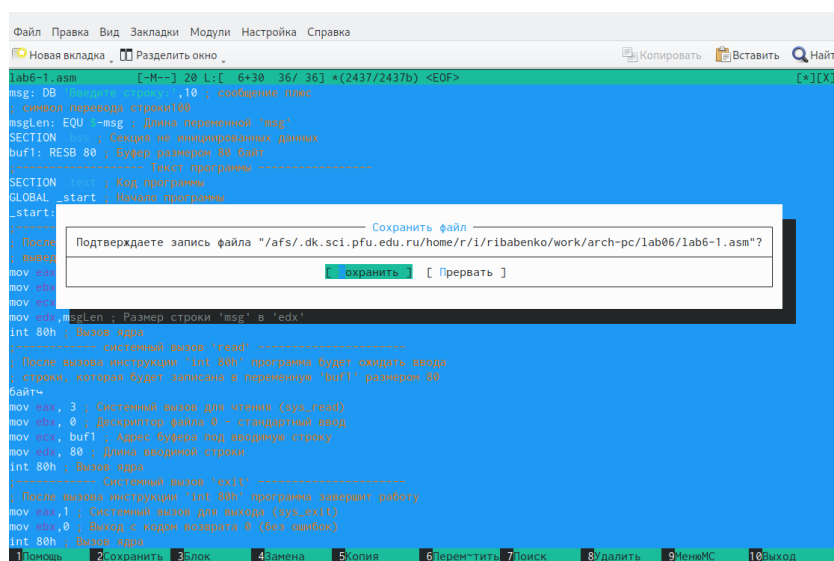


Рис. 2.3: Записываю программу в файл

Убеждаемся, что файл содержит текст программы (рис. 2.4)

```

/afs/.dk.sci.pfu.edu.ru/home/r/i/ribabenko/work/arch-pc/lab06/lab6-1.asm
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки100
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 ;Descriptor файла 0 - стандартный ввод
mov ecx,buf1 ; Адрес буфера под вводимую строку
mov edx,80 ; Длина вводимой строки
int 80h ; Вызов ядра

```

Рис. 2.4: Проверка файла

Оттранслируем текст программы в объектный файл и выполняем компоновку. Запускаем исполняемый файл. На запрос вводим ФИО(ribabenko). (рис. 2.5)

```

ribabenko@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
ribabenko@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
ribabenko@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-1
Введите строку:
ribabenko
ribabenko@dk8n60 ~/work/arch-pc/lab06 $ █

```

Рис. 2.5: Выполняем программу

Скачиваем файл 'in_out.asm' и копируем его в каталог, в котором лежит файл с программой. (рис. 2.6)

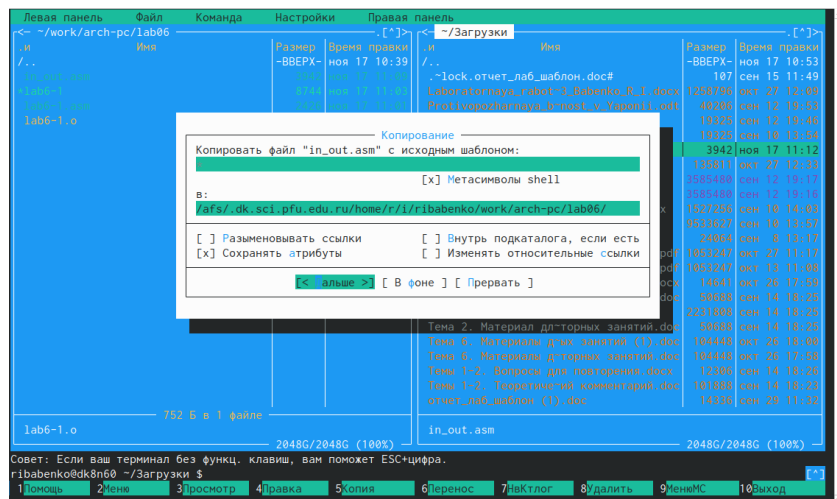


Рис. 2.6: копирование 'in_out.asm'

Создаём копию lab6-1 и записываем в неё листинг 6.2. (рис. 2.7)

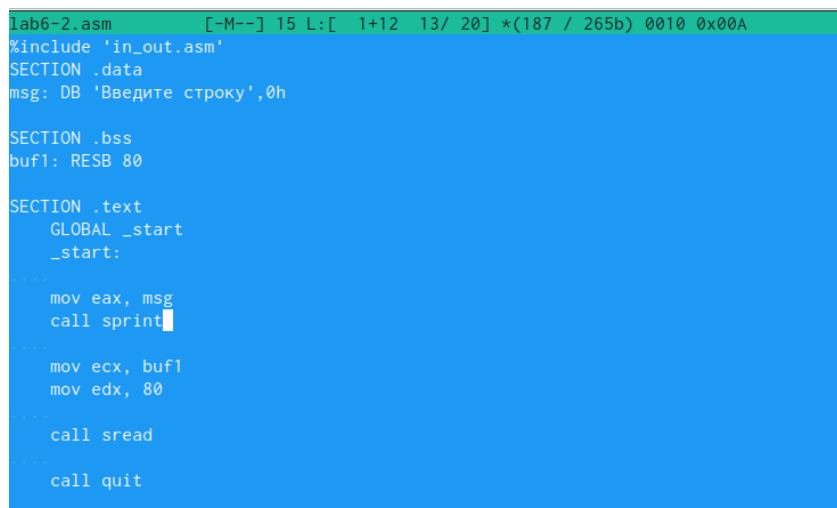


Рис. 2.7: Создание второго файла

Проверяем работу второго файла. (заменяли `sprintf` на `sprint`) При второй команде ввод текста происходит именно в данную строку.(рис. 2.8)

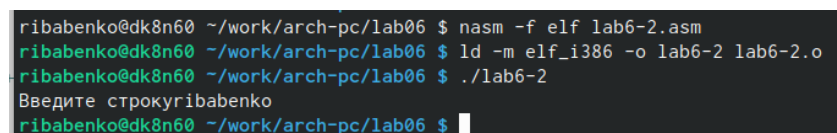


Рис. 2.8: Проверяем работу второго файла.

#Задания для самостоятельной работы

Создаём копию первого файла и вносим изменения в программу (добавляем вывод). (рис. 2.9)

```
GNU nano 0.3 /ats/.dk.sc1.ptu.edu.ru/pome/g/1/gibabenko
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 ;Descriptor 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
mov ebx,1
mov ecx,buf1
mov edx,80
int 80h

Сохранить изменённый буфер?
Y Да
N Нет ^C Отмена
```

Рис. 2.9: Вносим изменения в копию первого файла

Проверяем работоспособность программы (рис. 2.10)

```
ribabenko@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
ribabenko@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-3
Введите строку:
Бабенко
Бабенко
ribabenko@dk8n60 ~/work/arch-pc/lab06 $
```

Рис. 2.10: Проверка вывода

Создаём копию второго файла и вносим изменения в программу (добавляем вывод с использованием внешнего файла). (рис. 2.11)

```
GNU nano 6.3
#include 'in_out.asm
SECTION .data
msg: DB 'Введите строку'
msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprintLF

mov ecx, buf1
mov edx, 80
call sread

mov eax, buf1
mov ebx, 80
call sprint
```

Рис. 2.11: Вносим изменения в копию второго файла

Проверка вывода (рис. 2.12)

```
ribabenko@dk8n80 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
ribabenko@dk8n80 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
ribabenko@dk8n80 ~/work/arch-pc/lab06 $ ./lab6-4
Введите строку
Рома Б
Рома Б
ribabenko@dk8n80 ~/work/arch-pc/lab06 $
```

Рис. 2.12: Проверка второго файла

3 Выводы

В ходе выполнения данной лабораторной работы я приобрёл практические навыки работы в Midnight Commander и освоил инструкции языка ассемблера `mov` и `int`.