

Отчёт по лабораторной работе №5

Дисциплина: Архитектура Компьютера

Иванов Сергей Владимирович

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Выводы	14

Список иллюстраций

2.1	Midnight Kommander	5
2.2	Создание папки и файла lab5-1.asm	6
2.3	Редактируем lab5-1.asm	6
2.4	Трансляция и проверка исполняемого файла	7
2.5	Скачиваем файл	7
2.6	Копирование файла в каталог	8
2.7	Создание копии файла	8
2.8	Редактируем программу	9
2.9	Создание и проверка исполняемого файла	9
2.10	Замена подпрограммы	9
2.11	Копируем и редактируем lab5-3.asm	10
2.12	Проверяем программу	10
2.13	Копируем и редактируем lab5-4.asm	12
2.14	Проверяем программу	12

1 Цель работы

Целью работы является приобретение практических навыков работы в Midnight Commander и освоение инструкций языка ассемблера `mov` и `int`.

2 Выполнение лабораторной работы

Откроем Midnight Commander и перейдем в каталог созданный при выполнении лабораторной работы №4.(Рис. 2.1)

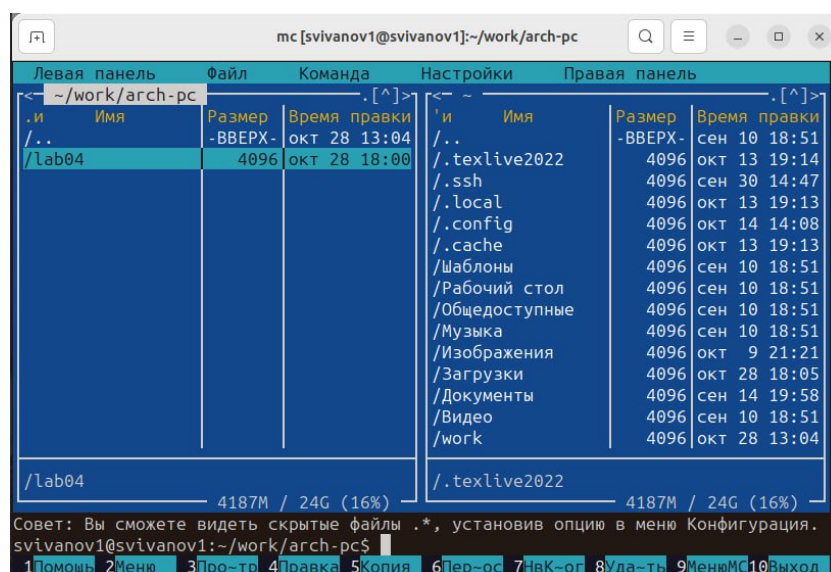


Рис. 2.1: Midnight Kommander

Создадим папку lab05, а в ней файл lab5-1.asm (Рис. 2.2)

Левая панель	Файл	Команда	Настройки	Права
~ /work/arch-pc/lab05				
.и		Имя	Размер	Время правки
/..		-ВВЕРХ-	ноя 10 20:46	
lab5-1.asm		0	ноя 10 20:49	

Рис. 2.2: Создание папки и файла lab5-1.asm

Откроем файл lab5-1.asm и отредактируем его. (Рис. 2.3)

```

GNU nano 7.2 /home/svivanov1/work/arch-pc/lab05/lab5-1.asm
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите 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
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 ; Дескриптор файла 0 - стандартный ввод
mov ecx, buf1 ; Адрес буфера под вводимую строку
mov edx, 80 ; Длина вводимой строки
int 80h ; Вызов ядра
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра

```

Рис. 2.3: Редактируем lab5-1.asm

Оттранслируем текст программы в объектный файл. Выполним компоновку объектного файла и запустим исполняемый файл. На запрос вводим ФИО. (Рис. 2.4)

```

svivanov1@svivanov1:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
svivanov1@svivanov1:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
svivanov1@svivanov1:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Иванов Сергей Владимирович
svivanov1@svivanov1:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Иванов Сергей Владимирович

```

Рис. 2.4: Трансляция и проверка исполняемого файла

Скачаем файл `in_out.asm` со страницы курса в ТУИС. (Рис. 2.5)

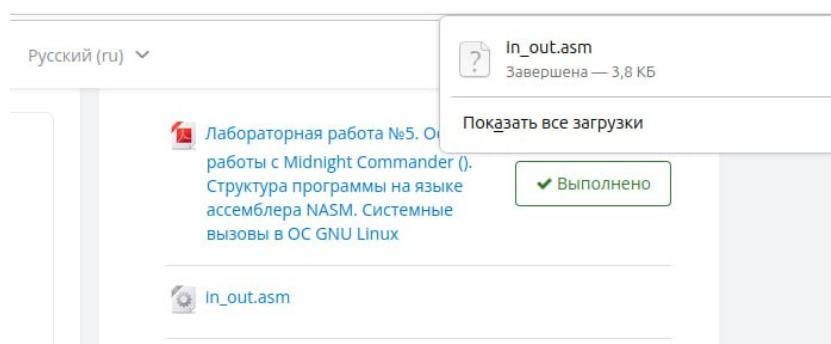


Рис. 2.5: Скачиваем файл

Скопируем файл `in_out.asm` в каталог с файлом `lab5-1.asm` с помощью функциональной клавиши `F5`. (Рис. 2.6)



Рис. 2.6: Копирование файла в каталог

Создадим копию файла lab5-1.asm с именем lab5-2.asm (Рис. 2.7)

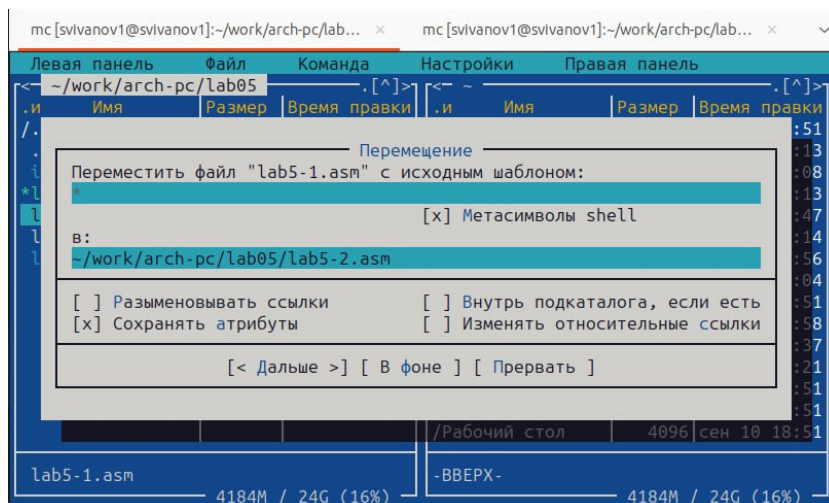
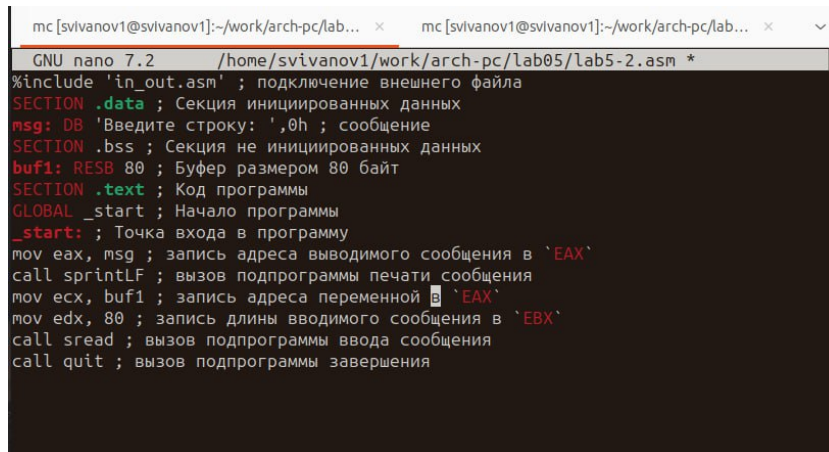


Рис. 2.7: Создание копии файла

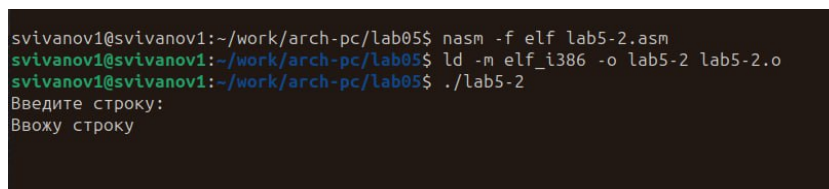
Исправим текст программы lab5-2.asm с использованием подпрограмм из внешнего файла in_out.asm (Рис. 2.8)



```
mc [svivanov1@svivanov1]:~/work/arch-pc/lab... x mc [svivanov1@svivanov1]:~/work/arch-pc/lab... x
GNU nano 7.2 /home/svivanov1/work/arch-pc/lab05/lab5-2.asm *
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение
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, msg ; запись адреса выводимого сообщения в `EAX`
call printf ; 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. 2.8: Редактируем программу

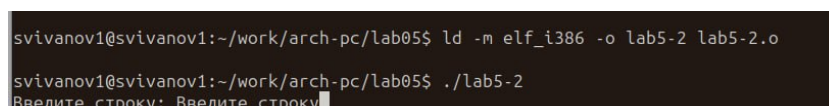
Создадим исполняемый файл и проверим его работу. Все работает правильно.
(Рис. 2.9)



```
svivanov1@svivanov1:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
svivanov1@svivanov1:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
svivanov1@svivanov1:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Ввожу строку
```

Рис. 2.9: Создание и проверка исполняемого файла

В файле lab5-2.asm заменим подпрограмму printf на print. Создадим исполняемый файл и проверим его работу. Разница в том, что теперь строка на ввод не переносится, как это было в прошлом варианте. (Рис. 2.10)



```
svivanov1@svivanov1:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
svivanov1@svivanov1:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Введите строку
```

Рис. 2.10: Замена подпрограммы

Создадим копию файла lab5-1.asm с именем lab5-3.asm. Внесем изменения в программу чтобы она выводила введенную строку на экран.(без использования внешнего файла in_out.asm) (Рис. 2.11)

```

GNU nano 7.2 /home/svivanov1/work/arch-pc/lab05/lab5-3.asm
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 ; 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
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 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,buf1 ; Адрес буфера под вводимую строку
mov edx,buf1 ; Размер строки buf1
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
[ Записано 25 строк ]
^C Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^S Позиция
^X Выход ^R ЧитФайл ^M Замена ^U Вставить ^D Выровнять ^_/ К строке

```

Рис. 2.11: Копируем и редактируем lab5-3.asm

Получим исполняемый файл и убедимся что он работает правильно. На приглашение ввести строку вводим свою фамилию. (Рис. 2.12)

```

svivanov1@svivanov1:~/work/arch-pc/lab05$ ./lab5-3
Введите строку:
Иванов
Иванов
svivanov1@svivanov1:~/work/arch-pc/lab05$

```

Рис. 2.12: Проверяем программу

Код программы из пункта 1 самостоятельной работы:

```

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
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 ; 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

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

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

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

```

Создадим копию файла lab5-1.asm с именем lab5-4.asm. Внесем изменения в программу чтобы она выводила введенную строку на экран.(с использованием внешнего файла in_out.asm) (Рис. 2.13)

```

GNU nano 7.2 /home/svivanov1/work/arch-pc/lab05/lab5-4.asm
#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 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

```

[Прочитано 18 строк]

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^С Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^/ К строке

Рис. 2.13: Копируем и редактируем lab5-4.asm

Получим исполняемый файл и убедимся что он работает правильно. (Рис. 2.14)

```

svivanov1@svivanov1:~$ nasm -f elf lab5-4.asm
nasm: fatal: unable to open input file 'lab5-4.asm' No such file or directory

svivanov1@svivanov1:~/work/arch-pc/lab05$ nasm -f elf lab5-4.asm

svivanov1@svivanov1:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-4 lab5-4.o

svivanov1@svivanov1:~/work/arch-pc/lab05$ ./lab5-4
Введите строку: Иванов Сергей Владимирович
Иванов Сергей Владимирович

svivanov1@svivanov1:~/work/arch-pc/lab05$

```

Рис. 2.14: Проверяем программу

Код программы из пункта 3 самостоятельной работы:

```

#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 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
```

3 Выводы

В результате выполнения лабораторной работы мы приобрели практические навыки работы в Midnight Commander и освоили инструкции языка ассемблера `mov` и `int`.