Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Шония Ника Гигловна

Содержание

Цель работы 5

Задание 6

Теоретическое введение 7

Выполнение лабораторной работы 9

Выводы 14

Список литературы 15

Список таблиц

Список иллюстраций

Цель работы

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

Задание

1. Основы работы с mc
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла
4. Выполнение заданий для самостоятельной работы

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 размеров 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 (учетве- рённое с
DT (define ten bytes) — определяет переменную размером в 10 байт. Директивы испол
```

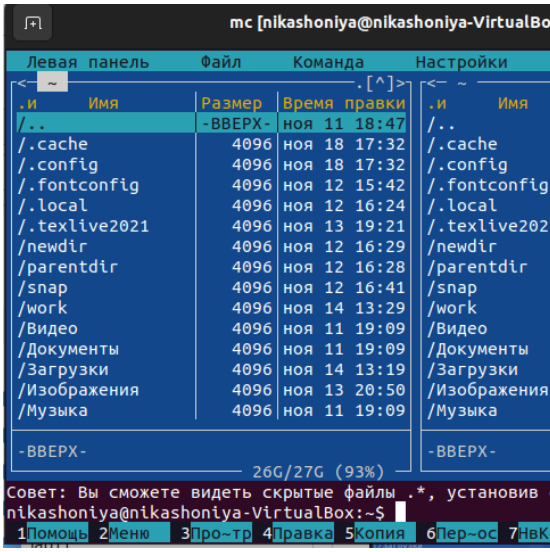
mov dst,src

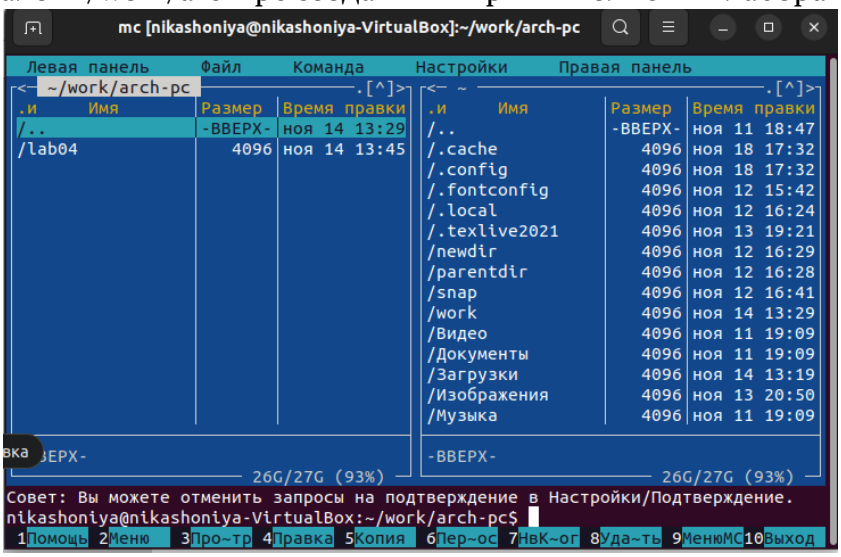
Здесь операнд dst — приёмник, а src — источник. В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и непосредственные значения (const). Инструкция языка ассемблера intпредназначена для вызова прерывания с указанным номером.

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).

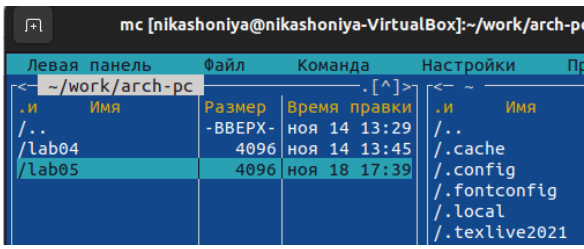
Выполнение лабораторной работы

1. Основы работы с mc Открываю Midnight Commander 

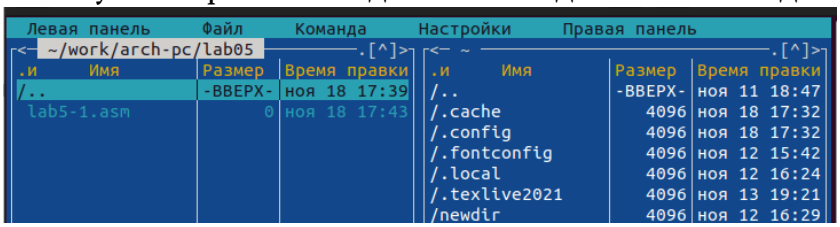
   Перехожу в каталог ~/work/arch-pc созданный при выполнении лабораторной работы №4 

   Со-

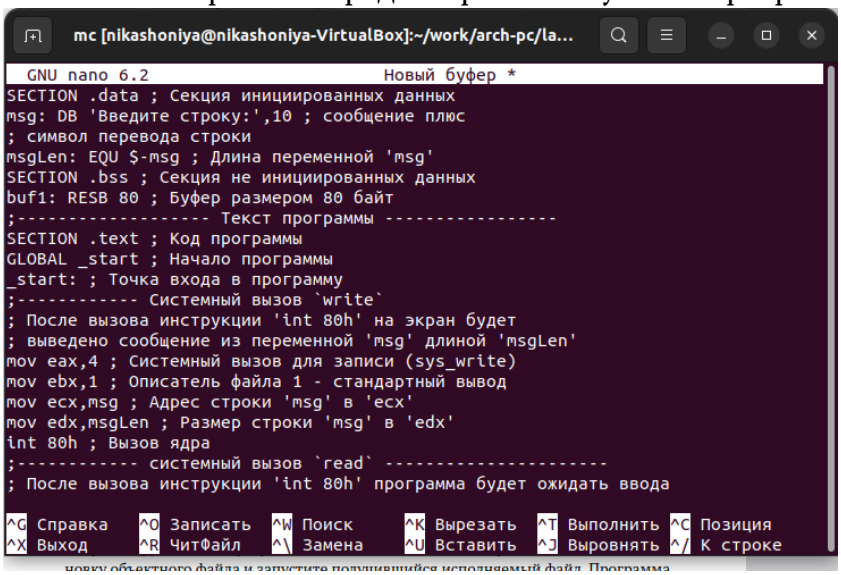
здаю папку lab05 и перехожу в созданный каталог.



Пользуясь строкой ввода и командой touch создаю файл lab5-1.asm



2. Структура программы на языке ассемблера NASM Открываю файл lab5-1.asm для редактирования во встроенном редакторе и ввожу текст программы из листинга



Открываю файл lab5-1.asm для просмотра. Убедилась, что файл содержит

текст программы

```
/home/nikashoniya/work/arch-pc/lab05/lab5-1.asm
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
;------------------ 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 ; 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
```

Оттранслировала текст программы lab5-1.asm в объектный файл. Выполнила компоновку объектного файла и запустила получившийся исполняемый файл

```
nikashoniya@nikashoniya-VirtualBox:~$ cd ~/work/arch-pc/lab05
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Шония Ника Гигловна
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$
```

3. Подключение внешнего файла Скачала файл in_out.asm

```
.и                Имя          Размер
/..                            -ВВЕРХ-
 in_out.asm                       3942
*lab5-1                           8744
 lab5-1.asm                       2096
 lab5-1.o                          752
```

Создаю копию файла lab5-1.asm с именем lab5-2.asm

```
.и                Имя          Размер   Вре
/..                            -ВВЕРХ-   ноя
 in_out.asm                       3942   ноя
*lab5-1                           8744   ноя
 lab5-1.o                          752   ноя
 lab5-2.asm                       2096   ноя
```

Исправляю текст программы в файле lab5-2.asm с использование подпрограмм из внешнего файла in_out.asm

```
 GNU nano 6.2                                  /home/nikashoniya/work/arch-pc/lab0
%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 sprintLF ; 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
```

Создаю исполняемый файл и проверяю его работу

```
nikashoniya@nikashoniya-VirtualBox:~$ cd ~/work/arch-pc/lab05
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$ nasm
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$ ld -
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$ ./lab
Введите строку:
Шония Ника Гигловна
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$
```

4. Выполнение заданий для самостоятельной работы Создаю копию файла lab5-1.asm. Вношу 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.

```
 GNU nano 6.2                          /home/nikashoniya/work/arch-pc/lab05/lab5-
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 строки buf1 в ecx
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
```

Создаю файл и проверяю его

```
nikashoniya@nikashoniya-VirtualBox:~$ cd ~/work/arch-pc/lab05
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Шония Ника Гигловна
Шония Ника Гигловна
```

Создаю копию файла lab5-2.asm. Исправьте текст программы с использование 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.

```
%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 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
```

Создаю файл и проверяю его

```
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Шония Ника Гигловна
Шония Ника Гигловна
nikashoniya@nikashoniya-VirtualBox:~/work/arch-pc/lab05$
```

Выводы

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

Список литературы

Архитектура компьютера Мой репозиторий: https://github.com/NikaShoniya/study_2023-2024_arch-pc