**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**по лабораторной работе № 5**

*дисциплина: Архитектура компьютера*

Студент: Ниемек Яи Жак

Группа: НММБд-04-24

**МОСКВА**

2025 г.

Цель

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

Задание

1. Откройте Midnight Commander user@dk4n31:~$ mc

2. Пользуясь клавишами ↑ , ↓ и Enter перейдите в каталог ~/work/arch-pc созданный

при выполнении лабораторной работы №5.

3. С помощью функциональной клавиши F7 создайте папку lab05 и перейдите в

созданный каталог.

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

5. С помощью функциональной клавиши F4 откройте файл lab5-1.asm для

редактирования во встроенном редакторе. Как правило в качестве встроенного

редактора Midnight Commander используется редакторы nano или mcedit.

6. Введите текст программы из листинга 5.1 (можно без комментариев), сохраните

изменения и закройте файл.

7. С помощью функциональной клавиши F3 откройте файл lab5-1.asm для просмотра.

Убедитесь, что файл содержит текст программы.

8. Оттранслируйте текст программы lab5-1.asm в объектный файл. Выполните

компоновку объектного файла и запустите получившийся исполняемый файл.

выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. На запрос введите

Ваши ФИО.

9. Скачайте файл in\_out.asm со страницы курса в ТУИС.

10. Подключаемый файл in\_out.asm должен лежать в том же каталоге, что и файл с

программой, в которой он используется.

11. С помощью функциональной клавиши F6 создайте копию файла lab5-1.asm с

именем lab5-2.asm. Выделите файл lab5-1.asm, нажмите клавишу F6 , введите имя

файла lab5-2.asm и нажмите клавишу Enter

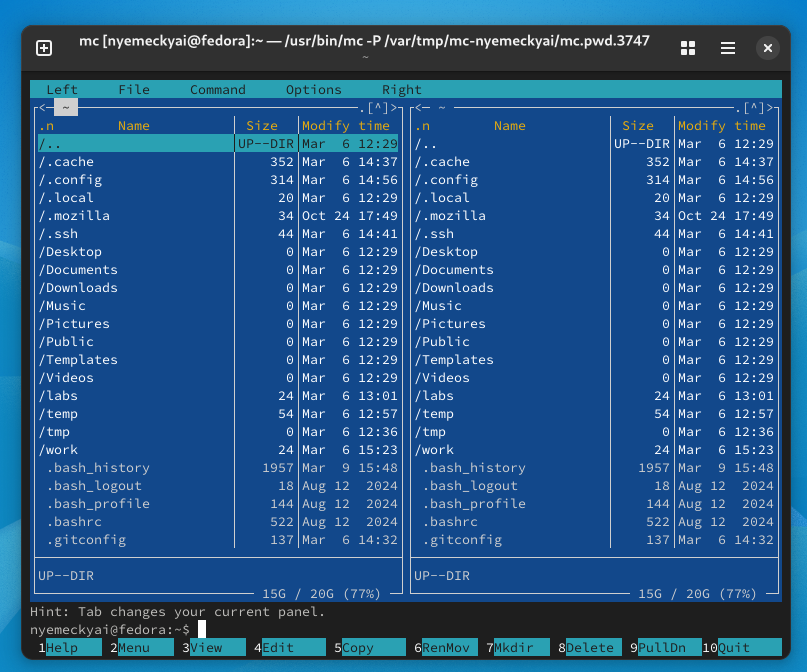
12. Исправьте текст программы в файле lab5-2.asm с использование подпрограмм из

внешнего файла in\_out.asm (используйте подпрограммы sprintLF, sread и quit) в

соответствии с листингом 5.2. Создайте исполняемый файл и проверьте его работу.

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. Для активации оболочки Midnight Commander достаточно ввести в командной строке mc и нажать клавишу Enter (рис. 5.1). В Midnight Commander используются функциональные клавиши F1 — F10 , к которым привязаны часто выполняемые операции (табл. 5.1). Таблица 5.1. Функциональные клавиши Midnight Commander Функциональные клавиши Выполняемое действие F1 вызов контекстно-зависимой подсказки F2 вызов меню, созданного пользователем F3 просмотр файла, на который указывает подсветка в активной панели F4 вызов встроенного редактора для файла, на который указывает подсветка в активной панели F5 копирование файла или группы отмеченных файлов из каталога, отображаемого в активной панели, в каталог, отображаемый на второй панели F6 перенос файла или группы отмеченных файлов из каталога, отображаемого в активной панели, в каталог, отображаемый на второй панели F7 создание подкаталога в каталоге, отображаемом в активной панели F8 удаление файла (подкаталога) или группы отмеченных файлов F9 вызов основного меню программы F10 выход из программы Следующие комбинации клавиш облегчают работу с Midnight Commander: • Tab используется для переключениями между панелями; • ↑ и ↓ используется для навигации, Enter для входа в каталог или открытия файла (если в файле расширений mc.ext заданы правила связи определённых расширений файлов с инструментами их запуска или обработки); • Ctrl + u (или через меню Команда > Переставить панели ) меняет местами содержимое правой и левой панелей; • Ctrl + o (или через меню Команда > Отключить панели ) скрывает или возвращает панели Midnight Commander, за которыми доступен для работы командный интерпретатор оболочки и выводимая туда информация. • Ctrl + x + d (или через меню Команда > Сравнить каталоги ) позволяет сравнить содержимое каталогов, отображаемых на левой и правой панелях. Дополнительную информацию о Midnight Commander можно получить по команде man mc и на странице проекта [3]. 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). Таким образом, общая структура программы имеет следующий вид: SECTION .data ; Секция содержит переменные, для … ; которых задано начальное значение SECTION .bss ; Секция содержит переменные, для … ; которых не задано начальное значение SECTION .text ; Секция содержит код программы GLOBAL \_start \_start: ; Точка входа в программу … ; Текст программы mov eax,1 ; Системный вызов для выхода (sys\_exit) mov ebx,0 ; Выход с кодом возврата 0 (без ошибок) int 80h ; Вызов ядра 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 (учетверённое 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. Синтаксис директив определения данных следующий: DB [, ] [, ] Таблица 5.2. Примеры Пример Пояснение a db 10011001b определяем переменную a размером 1 байт с начальным значением, заданным в двоичной системе счисления (на двоичную систему счисления указывает также буква b (binary) в конце числа) b db ‘!’ определяем переменную b в 1 байт, инициализируемую символом ! c db “Hello” определяем строку из 5 байт d dd -345d определяем переменную d размером 4 байта с начальным значением, заданным в десятичной системе счисления (на десятичную систему указывает буква d (decimal) в конце числа) h dd 0f1ah определяем переменную h размером 4 байта с начальным значением, заданным в шестнадцатеричной системе счисления (h — hexadecimal) Для объявления неинициированных данных в секции .bss используются директивы resb, resw, resd и другие, которые сообщают ассемблеру, что необходимо зарезервировать заданное количество ячеек памяти. Примеры их использования приведены в табл. 5.3 resb Резервирование заданного числа однобайтовых ячеек string resb 20 По адресу с меткой string будет расположен массив из 20 однобайтовых ячеек (хранение строки символов) resw Резервирование заданного числа двухбайтовых ячеек (слов) count resw 256 По адресу с меткой count будет расположен массив из 256 двухбайтовых слов resd Резервирование заданного числа четырёхбайтовых ячеек (двойных слов) x resd 1 По адресу с меткой x будет расположено одно двойное слово (т.е. 4 байта для хранения большого числа) 5.2.3. Элементы программирования 5.2.3.1. Описание инструкции mov Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике. В общем виде эта инструкция записывается в виде mov dst,src Здесь операнд dst — приёмник, а src — источник. В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и непосредственные значения (const). В табл. 5.4 приведены варианты использования mov с разными операндами Тип операндов Пример Пояснение mov , mov eax,ebx пересылает значение регистра ebx в регистр eax mov , mov cx,[eax] пересылает в регистр cx значение из памяти, указанной в eax mov , mov rez,ebx пересылает в переменную rez значение из регистра ebx mov , mov eax,403045h пишет в регистр eax значение 403045h mov , mov byte[rez],0 записывает в переменную rez значение 0 ВАЖНО! Переслать значение из одной ячейки памяти в другую нельзя, для этого необходимо использовать две инструкции mov: mov eax, x mov y, eax Также необходимо учитывать то, что размер операндов приемника и источника должны совпадать. Использование слудующих примеров приведет к ошибке: • mov al,1000h — ошибка, попытка записать 2-байтное число в 1-байтный регистр; • mov eax,cx — ошибка, размеры операндов не совпадают. 5.2.3.2. Описание инструкции int Инструкция языка ассемблера intпредназначена для вызова прерывания с указанным номером. В общем виде она записывается в виде 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). После вызова инструкции int 80h выполняется системный вызов какойлибо функции ядра Linux. При этом происходит передача управления ядру операционной системы. Чтобы узнать, какую именно системную функцию нужно выполнить, ядро извлекает номер системного вызова из регистра eax. Поэтому перед вызовом прерывания необходимо поместить в этот регистр нужный номер. Кроме того, многим системным функциям требуется передавать какие-либо параметры. По принятым в ОС Linux правилам эти параметры помещаются в порядке следования в остальные регистры процессора: ebx, ecx, edx. Если системная функция должна вернуть значение, то она помещает его в регистр eax. 5.2.3.3. Системные вызовы для обеспечения диалога с пользователем Простейший диалог с пользователем требует наличия двух функций — вывода текста на экран и ввода текста с клавиатуры. Простейший способ вывести строку на экран — использовать системный вызов write. Этот системный вызов имеет номер 4, поэтому перед вызовом инструкции int необходимо поместить значение 4 в регистр eax. Первым аргументом write, помещаемым в регистр ebx, задаётся дескриптор файла. Для вывода на экран в качестве дескриптора файла нужно указать 1 (это означает «стандартный вывод», т. е. вывод на экран). Вторым аргументом задаётся адрес выводимой строки (помещаем его в регистр ecx, например, инструкцией mov ecx, msg). Строка может иметь любую длину. Последним аргументом (т.е. в регистре edx) должна задаваться максимальная длина выводимой строки. Для ввода строки с клавиатуры можно использовать аналогичный системный вызов read. Его аргументы –такие же, как у вызова write,только для «чтения» с клавиатуры используется файловый дескриптор 0 (стандартный ввод). Системный вызов exit является обязательным в конце любой программы на языке ассемблер. Для обозначения конца программы перед вызовом инструкции int 80h необходимо поместить в регистр еах значение 1, а в регистр ebx код завершения 0

Изображение выглядит как текст, Шрифт, число, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Контент, сгенерированный ИИ, может содержать ошибки.

**Ответы на вопросы для самопроверки**

**1. Каково назначение mc?**

**Midnight Commander (mc)** – это файловый менеджер для терминала в Linux, напоминающий Norton Commander. Он упрощает работу с файлами и каталогами, предоставляя **двухпанельный интерфейс** и поддержку горячих клавиш.

Основные возможности:

* Копирование, перемещение, удаление файлов и каталогов.
* Просмотр и редактирование файлов.
* Работа с архивами как с обычными каталогами.
* Доступ к удалённым файловым системам (FTP, SFTP).
* Настраиваемые горячие клавиши и макросы.

Запуск:

mc

**2. Какие операции с файлами можно выполнить как с помощью команд bash, так и с помощью меню (комбинаций клавиш) mc? Приведите несколько примеров.**

| **Операция** | **Команда в bash** | **Горячая клавиша в mc** |
| --- | --- | --- |
| Копирование | cp file1 file2 | F5 |
| Перемещение | mv file1 file2 | F6 |
| Удаление | rm file | F8 |
| Создание папки | mkdir folder | F7 |
| Просмотр файла | cat file / less file | F3 |
| Редактирование | nano file / vim file | F4 |

Пример использования Midnight Commander:

1. Открываем mc:
2. mc
3. Наводим курсор на файл.
4. Нажимаем F5, указываем путь назначения → **Файл копируется**.

**3. Какова структура программы на языке ассемблера NASM?**

Программа на NASM состоит из следующих частей:

1. **Секция данных (section .data)** – хранит **инициализированные** данные.
2. **Секция bss (section .bss)** – для **неинициализированных** данных.
3. **Секция кода (section .text)** – основной код программы.

Пример структуры программы:

section .data ; Инициализированные данные

message db "Hello, world!", 0

section .bss ; Неинициализированные данные

buffer resb 256

section .text ; Основной код

global \_start ; Точка входа

\_start:

; Код программы

mov eax, 1 ; Системный вызов sys\_exit

xor ebx, ebx ; Код возврата 0

int 0x80 ; Вызов ядра

**4. Для описания каких данных используются секции bss и data в языке ассемблера NASM?**

| **Секция** | **Описание** |
| --- | --- |
| .data | Для **инициализированных** данных (константы, строки, числа). |
| .bss | Для **неинициализированных** данных (буферы, массивы, переменные). |

Пример:

section .data

num db 10 ; Инициализированная переменная

section .bss

buffer resb 100 ; Буфер на 100 байт (память выделена, но не инициализирована)

**5. Для чего используются компоненты db, dw, dd, dq и dt языка ассемблера NASM?**

Эти директивы определяют **размер переменных**:

| **Директива** | **Размер** | **Описание** |
| --- | --- | --- |
| db | 1 байт | Byte (8 бит) |
| dw | 2 байта | Word (16 бит) |
| dd | 4 байта | Double word (32 бита) |
| dq | 8 байт | Quad word (64 бита) |
| dt | 10 байт | Ten bytes (80 бит, используется для FPU) |

Пример:

section .data

a db 5 ; 8-битное число

b dw 1000 ; 16-битное число

c dd 12345678h ; 32-битное число

d dq 1122334455667788h ; 64-битное число

**6. Какое произойдёт действие при выполнении инструкции mov eax, esi?**

Инструкция:

mov eax, esi

* Копирует содержимое регистра ESI в EAX.
* Значение ESI **не изменяется**.
* Не изменяет флаги флагового регистра (EFLAGS).

Пример:

mov esi, 1234h ; Загрузить 1234h в ESI

mov eax, esi ; EAX теперь тоже 1234h

**7. Для чего используется инструкция int 80h?**

Инструкция int 0x80 используется для **системных вызовов в Linux**. Она переключает процессор в **режим ядра** и передаёт управление **системному вызову**.

Пример:

mov eax, 1 ; Код системного вызова sys\_exit

xor ebx, ebx ; Код возврата 0

int 0x80 ; Вызов ядра

Этот код завершает выполнение программы.

**Основные коды системных вызовов:**

| **Код EAX** | **Функция** | **Описание** |
| --- | --- | --- |
| 1 | sys\_exit | Завершить программу |
| 3 | sys\_read | Прочитать данные из файла |
| 4 | sys\_write | Записать данные в файл |
| 5 | sys\_open | Открыть файл |
| 6 | sys\_close | Закрыть файл |

Пример вывода строки в терминал:

section .data

msg db "Hello, world!", 10 ; Строка с переводом строки

len equ $ - msg ; Длина строки

section .text

global \_start

\_start:

mov eax, 4 ; sys\_write

mov ebx, 1 ; Файл (stdout)

mov ecx, msg ; Адрес строки

mov edx, len ; Длина строки

int 0x80 ; Вызов ядра

mov eax, 1 ; sys\_exit

xor ebx, ebx

int 0x80

🔹 Этот код выводит "Hello, world!" в консоль.

Выводы

Я приобрел практические навыки работы в Midnight Commander и освоил инструкции языка ассемблера mov и int