Лабораторная работа №5

Петлин Артём Дмитриевич

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

1. Создайте копию файла lab5-1.asm. Внесите 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.

2. Получите исполняемый файл и проверьте его работу. На приглашение ввести строку введите свою фамилию.
3. Создайте 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

4. Создайте исполняемый файл и проверьте его работу.

3 Теоретическое введение

3.1 Основы работы с Midnight Commander.

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.

3.2 Структура программы на языке ассемблера NASM

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:

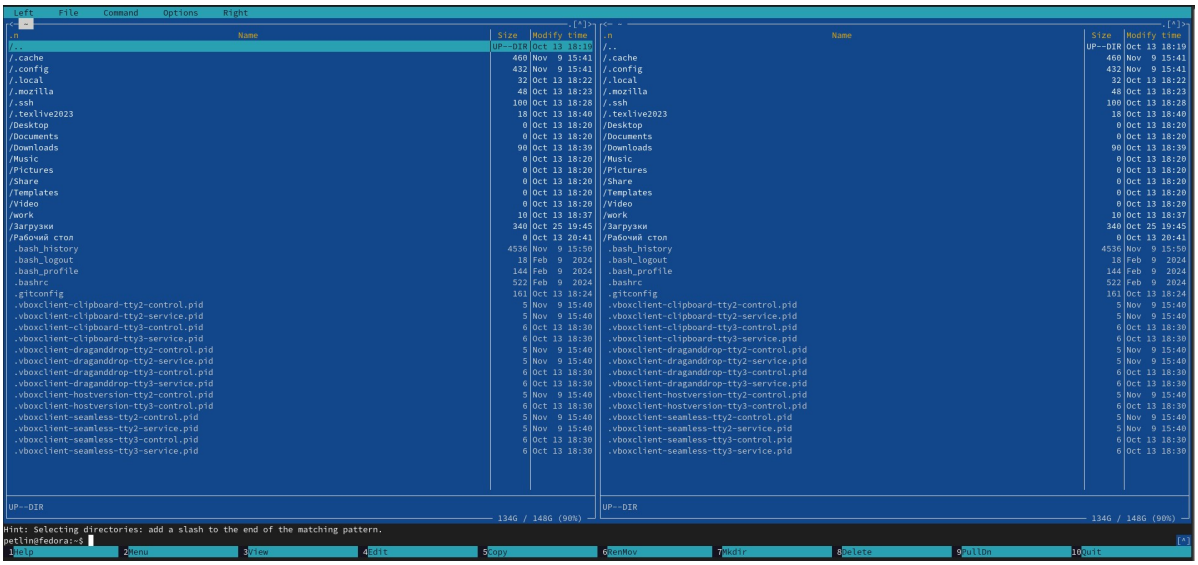
- DB (define byte) — определяет переменную размером в 1 байт; - DW (define word)

— 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 (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.

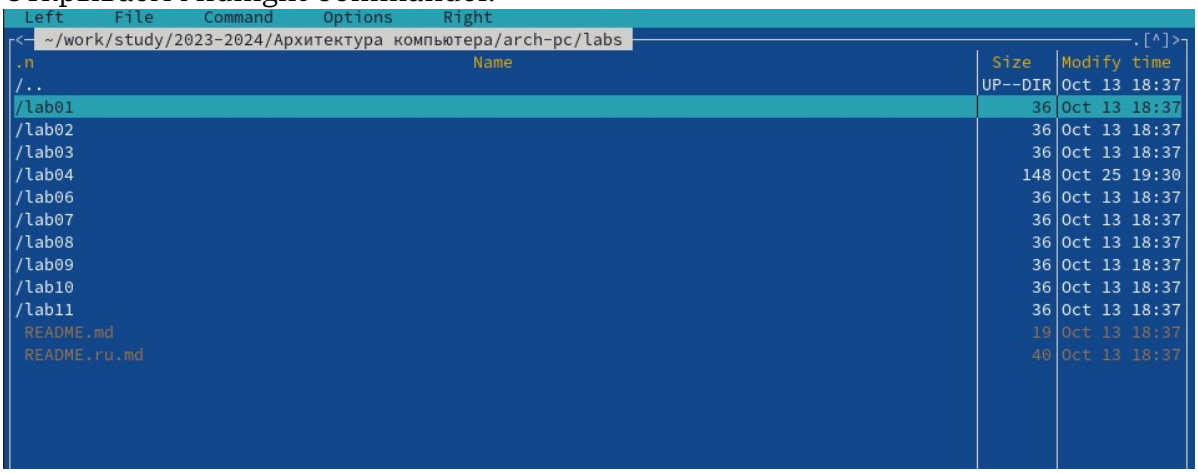
3.3 Элементы программирования

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).
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.*

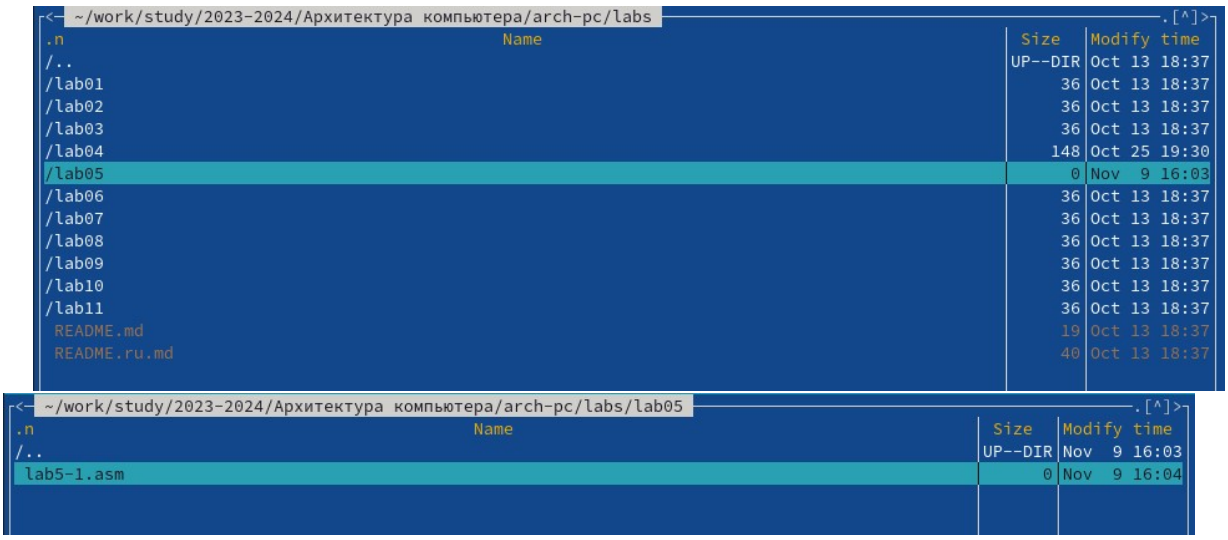
4 Выполнение лабораторной работы



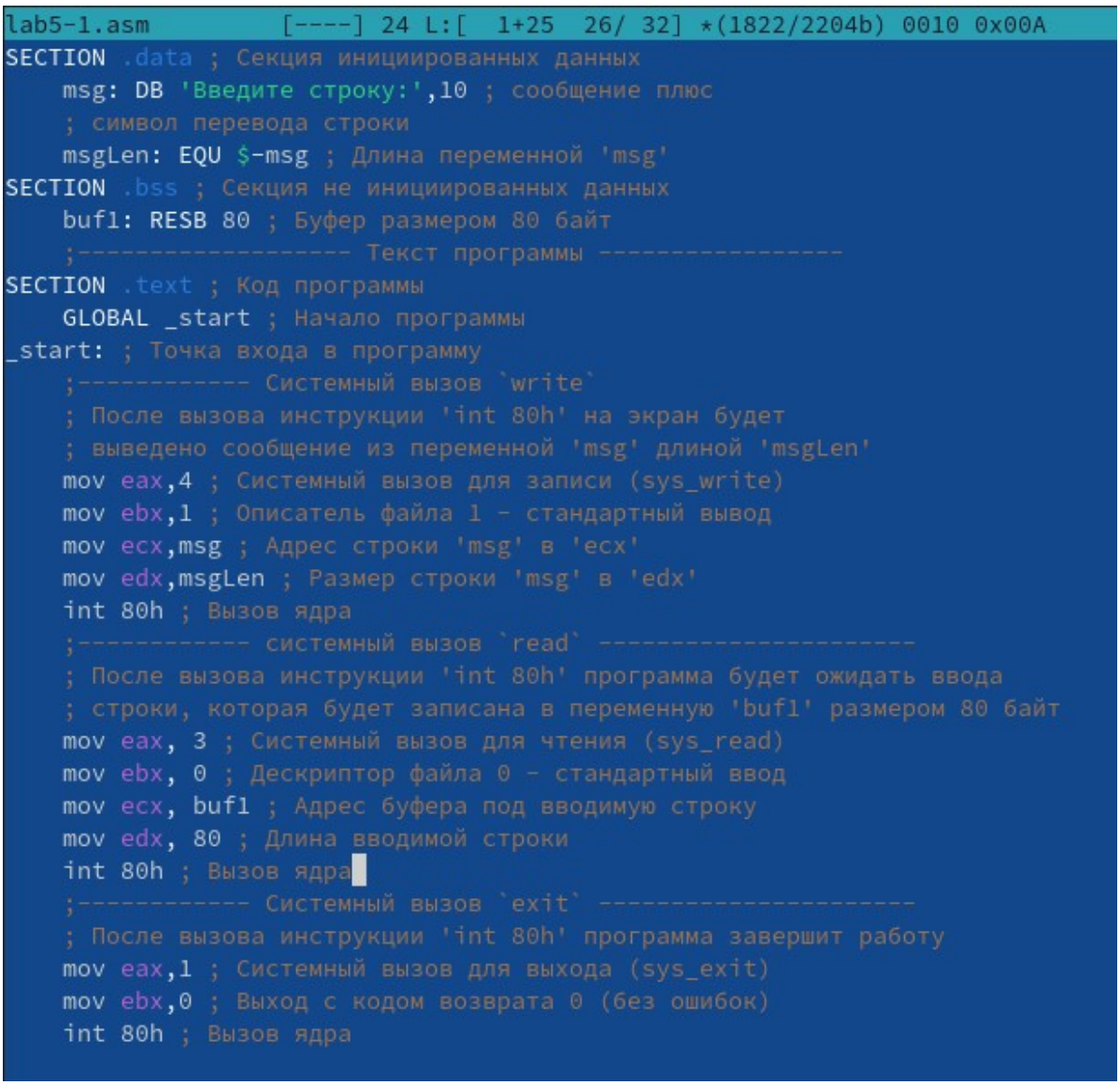
Открываем Midnight Commander.



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



С помощью функциональной клавиши F7 создаём папку lab05 и переходим в созданный каталог. Пользуясь строкой ввода и командой touch создаём файл lab5-1.asm



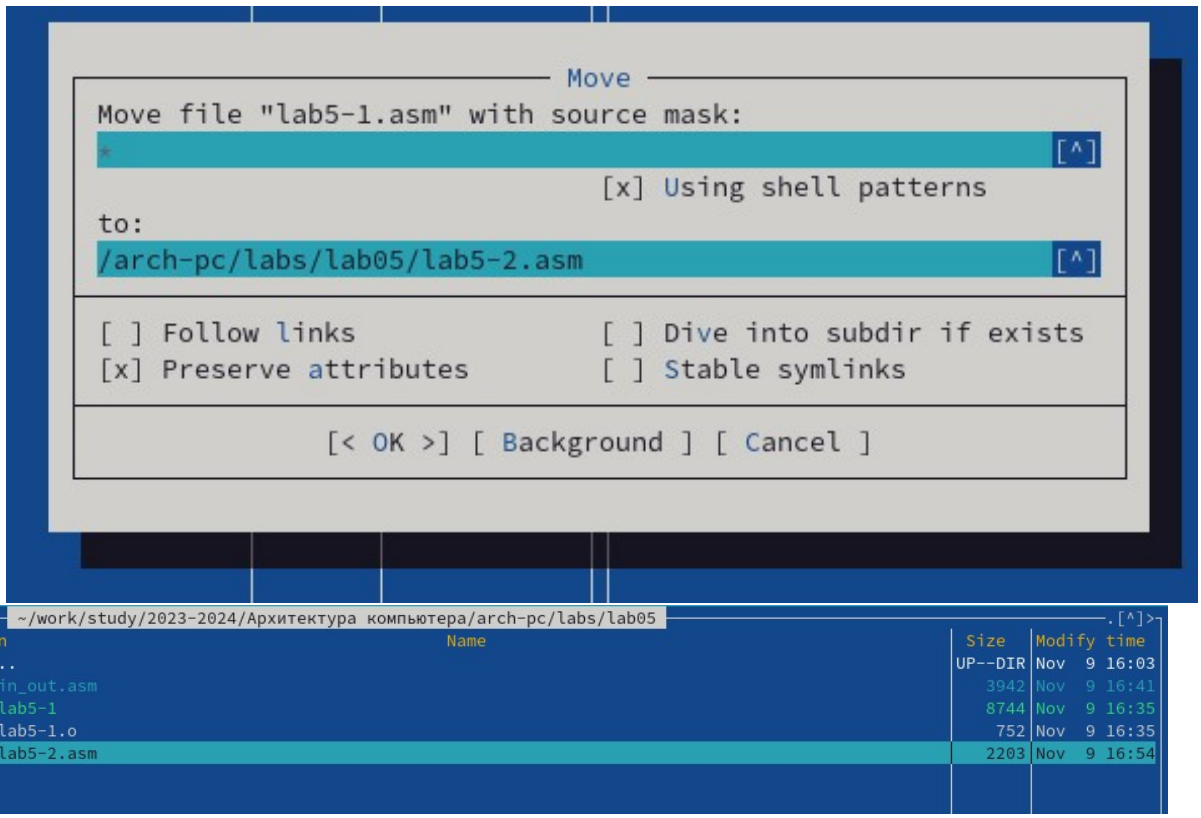
С помощью функциональной клавиши F4 открываем файл lab5-1.asm для редактирования во встроенном редакторе и вводим предложенный текст программы, сохраняем изменения и закрываем файл.

```
/home/petlin/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/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
```

С помощью функциональной клавиши F3 открываем файл lab5-1.asm для просмотра. Убеждаемся, что файл содержит текст программы.

```
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ nasm -f elf lab5-1.asm
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ls
lab5-1  lab5-1.asm  lab5-1.o
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ./lab5-1
Введите строку:
Петлин Артём Дмитриевич
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$
```

Оттранслируем текст программы lab5-1.asm в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл. На запрос вводим своё ФИО.



Скачиваем файл in_out.asm и переносим его в тот же каталог, где лежит и файл с программой, в которой он используется. С помощью функциональной клавиши F6 создаём копию файла lab5-1.asm с именем lab5-2.asm.

```
/home/petlin/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05/lab5-2.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 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
```

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

```
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ nasm -f elf lab5-2.asm
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ls
in_out.asm  lab5-1  lab5-1.o  lab5-2  lab5-2.asm  lab5-2.o
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ./lab5-2
Введите строку:
Петлин Артём Дмитриевич
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$
```

Создайте исполняемый файл и проверьте его работу.

```
lab5-2.asm          [BM--]  9 L:[  1+ 9  10/ 14] *(600 / 997b) 0115 0x073
%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
    call quit ; вызов подпрограммы завершения
```

В файле lab5-2.asm заменяем подпрограмму sprintLF на sprint.

```
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ nasm -f elf lab5-2.asm
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ls
in_out.asm  lab5-1  lab5-1.o  lab5-2  lab5-2.asm  lab5-2.o
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ls
in_out.asm  lab5-1  lab5-1.o  lab5-2  lab5-2.asm  lab5-2.o
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ./lab5-2
Введите строку: Петлин Артём Дмитриевич
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$
```

Создаём исполняемый файл и проверяем его работу. Замечаем разницу, что sprintLF переводит ввод на следующую строку, в отличии от sprint, где фраза “Введите строку” и ввод с клавиатуры находяться на одной строке.

5 Задание для самостоятельной работы.

```
lab5-3.asm          [----] 24 L:[  1+37  38/ 38] *(2277/2277b) <EOF>
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

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

    ;------------ 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
```

Создаём копию файла lab5-1.asm (lab5-3.asm). Вносим 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.

```
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ nasm -f elf lab5-3.asm
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ld -m elf_i386 -o lab5-3 lab5-3.o
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ./lab5-3
Введите строку:
Петлин
Петлин
```

Получаем исполняемый файл и проверяем его работу. На приглашение ввести строку вводим свою фамилию.

```
lab5-4.asm          [----]  0 L:[  1+16  17/ 17] *(1034/1034b) <EOF>
%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
    mov eax, buf1
    call sprint
    call quit ; вызов подпрограммы завершения
```

Создайте копию файла lab5-2.asm (lab5-4.asm). Исправляем 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.

```
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ nasm -f elf lab5-4.asm
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ld -m elf_i386 -o lab5-4 lab5-4.o
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$ ./lab5-4
Введите строку:
Петлин
Петлин
petlin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab05$
```

Создаём исполняемый файл и проверяем его работу.

6 Выводы

Мы приобрели практические навыки работы в Midnight Commander, освоили инструкции языка ассемблера mov и int.

Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.

2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.

3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander. org/.

4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.

5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 c. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.

6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 c. — ISBN 978-1491941591.

7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.

8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 c. — ISBN 9781784396879.

9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.

10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.

11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.

12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.

13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.

14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).

16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).