

Отчёт по лабораторной работе №5

**Создание и процесс обработки программ на языке ассемблера
NASM**

Пашутина Анна Алексеевна

Содержание

1 Цель работы	5
2 Выполнение лабораторной работы	6
3 Выводы	12

Список иллюстраций

2.1	Используем команду mc	6
2.2	Проверяем, что mc работает. Должно открыться окно Midnight Com- mander	6
2.3	Переход в папку	6
2.4	Создаем папку	7
2.5	Переходим в папку lab05 и создаем файл	7
2.6	Открываем файл в редакторе	7
2.7	Записываем код, который нам дали	8
2.8	Проверяем, что мы все записали	8
2.9	Проверка наличия и компиляция	8
2.10	Проверка файлов	9
2.11	Копируем lab5-1.asm в lab5-2.asm, используя клавишу f5	9
2.12	Проверяем наличие lab5-2.asm	9
2.13	Открываем файл, записываем в него код и импортируем файл in_out.asm	10
2.14	Компилируем файл lab5-2.asm	10
2.15	Редактируем первый файл, чтобы соответствовать заданию	10
2.16	Компиляция и проверка работы отредактированного файла lab5- 1.asm	11
2.17	Редактируем второй файл, чтобы соответствовать заданию	11
2.18	Компиляция и проверка работы отредактированного файла lab5- 2.asm	11

Список таблиц

1 Цель работы

Познакомиться с синтаксисом языка программирования **ассемблер**

2 Выполнение лабораторной работы

1) Откроем Midnight Commander



Рисунок 2.1: Используем команду mc

2) Проверяем, что команда mc сработала верно

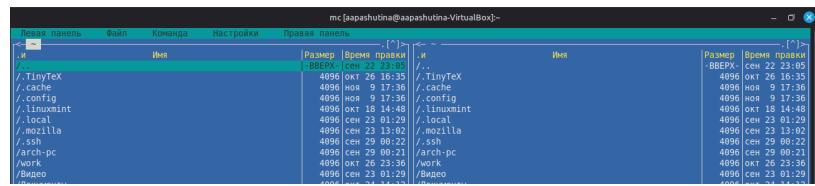


Рисунок 2.2: Проверяем, что mc работает. Должно открыться окно Midnight Commander

3) Переходим в папку work/arch-pc, где ранее работали

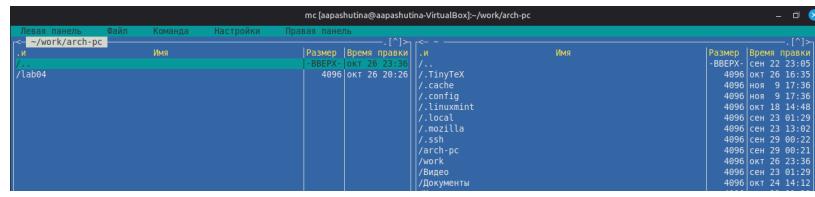


Рисунок 2.3: Переход в папку

4) Создаем папку lab05

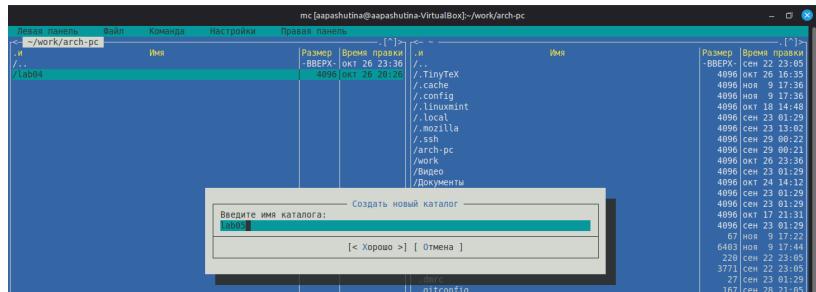


Рисунок 2.4: Создаем папку

5) Переходим в эту папку и создаем файл **lab5-1.asm**

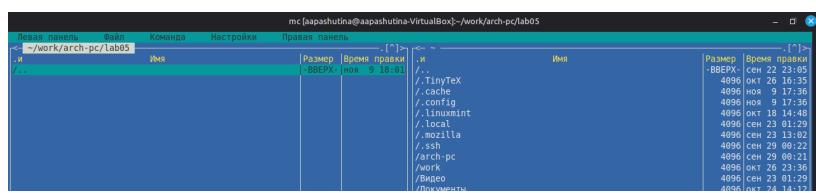


Рисунок 2.5: Переходим в папку lab05 и создаем файл

6) Открываем файл lab5-1.asm

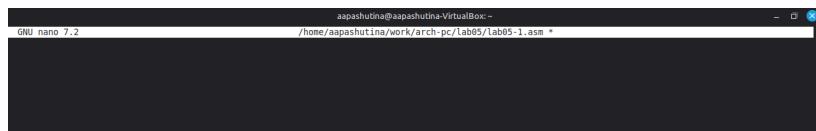


Рисунок 2.6: Открываем файл в редакторе

7) Записываем в этот файл код, который нам дали

```

GNU nano 7.2
/home/aapashutina/work/arch-pc/lab05/lab05-1.asm lab05-1.asm
; Программа вывода сообщения на экран и ввода строки с клавиатуры
; ..... Объявление переменных .....
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; Сообщение для вывода плюс
; символ перевода строки
msglen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных данных
buf: RESB 80 ; Буфер размером 80 байт

; ..... Текст программы .....
SECTION .text ; Код программы
GLOBAL start ; Начало программы
start: ; Точка входа в программу
        ; Системный вызов write
        mov eax,4
        ; Системный вызов для записи (sys write)
        mov ebx,1
        ; Дескриптор файла 1 - стандартный вывод
        mov ecx,buf
        ; 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' на экран будет ожидать ввода
        ; строки, которая будет записана в переменную 'buf' размером 80 байт
        mov eax,3
        ; Системный вызов для чтения (sys read)
        mov ebx,0
        ; Дескриптор файла 0 - стандартный ввод
        mov ecx,buf
        ; 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
; ..... Прочитано 36 строк ...

```

Рисунок 2.7: Записываем код, который нам дали

- 8) С помощью клавиши **f3** открываем файл для просмотра, чтобы убедиться, что мы все записали

```

mc [aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05]
/home/aapashutina/work/arch-pc/lab05/lab05-1.asm lab05-1.asm
; Программа вывода сообщения на экран и ввода строки с клавиатуры
; ..... Объявление переменных .....
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; Сообщение для вывода плюс
; символ перевода строки
msglen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных данных
buf: RESB 80 ; Буфер размером 80 байт

; ..... Текст программы .....
SECTION .text ; Код программы
GLOBAL start ; Начало программы
start: ; Точка входа в программу
        ; Системный вызов write
        mov eax,4
        ; Системный вызов для записи (sys write)
        mov ebx,1
        ; Дескриптор файла 1 - стандартный вывод
        mov ecx,buf
        ; 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 'buf' размером 80 байт
        mov eax,3
        ; Системный вызов для чтения (sys read)
        mov ebx,0
        ; Дескриптор файла 0 - стандартный ввод
        mov ecx,buf
        ; 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

```

Рисунок 2.8: Проверяем, что мы все записали

- 9) Проверяем, что файл в папке есть, компилируем файл и запускаем, чтобы проверить, как работает код, который мы вставили

```

aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab05-1.asm
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab05-1 lab05-1.o
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$ ls
lab05-1 lab05-1.asm lab05-1.asm.save lab05-1.o
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$ ./lab05-1
Введите строку:
pppppppp
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$ 

```

Рисунок 2.9: Проверка наличия и компиляция

- 10) Проверяем, что файл скомпилировался и создался объектный файл

```
apashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$ ls
lab05-1.asm  lab05-1.asm.o  lab05-1.asm.save  lab05-1.o
apashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$
```

Рисунок 2.10: Проверка файлов

- 11) Копируем содержимое первого файла в другой файл с названием lab5-2.asm

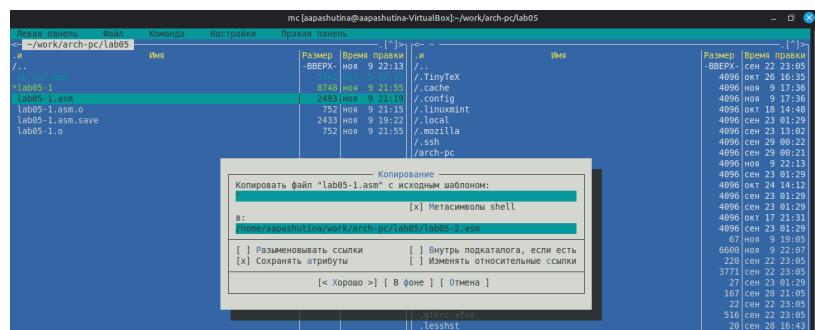


Рисунок 2.11: Копируем lab5-1.asm в lab5-2.asm, используя клавишу f5

- 12) Проверяем, что файл lab5-2.asm создался

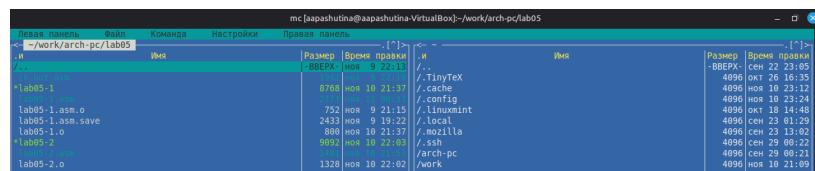


Рисунок 2.12: Проверяем наличие lab5-2.asm

- 13) Открываем файл lab5-2.asm и записываем в него код, который нам дали, предварительно скачиваем файл **in_out.asm**

```

lab05-2.asm (~\work\arch-pc\lab05)

;-----+
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----+
%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:
    mov eax, msg               ; Точка входа в программу
    call sprintLF              ; запись адреса выводимого сообщения в `EAX`
    mov ecx, buf1               ; вызов подпрограммы печати сообщения
    mov edx, 80                 ; запись адреса переменной в `EAX`
    mov eax, 80                 ; запись длины вводимого сообщения в `EBX`]

```

Рисунок 2.13: Открываем файл, записываем в него код и импортируем файл **in_out.asm**

- 14) Компилируем файл lab5-2.asm, проверяем, что объектный файл создался и смотрим, чтобы файл **in_out.asm** был в папке с файлом, в который мы его импротруем

```

aapashutina@apashutina-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
aapashutina@apashutina-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
aapashutina@apashutina-VirtualBox:~/work/arch-pc/lab05$ ls
in_out.asm lab5-1.asm lab5-1.asm.o lab5-1.asm.save lab5-1.o lab5-2 lab5-2.asm lab5-2.o
aapashutina@apashutina-VirtualBox:~/work/arch-pc/lab05$ 

```

Рисунок 2.14: Компилируем файл lab5-2.asm

- 15) Открываем первый файл **lab5-1.asm** и вносим в него изменения, чтобы он возвращал нам нашу **Фамилию и Имя**

```

GNU nano 7.2                               /home/aapashutina/work/arch-pc/lab05/lab5-1.asm
;-----+
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----+
;-----+ Определение переменных
;-----+
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h   ; сообщение
msgLen: EDX $msg ; Длина переменной `msg`

SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт
buf1_size: EDX 80

SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start:
    mov eax, msg               ; Точка входа в программу
    call sprintLF              ; запись адреса выводимого сообщения в `EAX`
    mov ecx, buf1               ; вызов подпрограммы печати сообщения
    mov edx, msgLen             ; запись адреса переменной в `EAX`
    mov eax, buf1_size           ; запись длины вводимого сообщения в `EBX`]

;-----+ 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, buf1_size ; Длина вводимой строки
int 80h ; Вызов ядра

mov esi, eax

```

Рисунок 2.15: Редактируем первый файл, чтобы соответствовать заданию

16) Компилируем файл **lab5-1.asm** и проверяем его работу

```
aapashutina@aapashutina-VirtualBox:~/Рабочий стол$ cd  
aapashutina@aapashutina-VirtualBox:~$ cd ~/work/arch-pc/lab05  
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab05-1.asm  
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab05-1 lab05-1.o  
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$ ./lab05-1  
Введите строку: Pashutina Anna  
Pashutina Anna  
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$
```

Рисунок 2.16: Компиляция и проверка работы отредактированного файла **lab5-1.asm**

17) Открываем файл **lab5-2.asm** и вносим в него изменения, чтобы он также возвращал нам нашу **Фамилию и Имя**

```
mc [aapashutina@aapashutina-VirtualBox]:~/work/arch-pc/lab05  
GNU nano 7.2  
/home/aapashutina/work/arch-pc/lab05/lab05-2.asm  
Программа вывода сообщения на экран и ввода строки с клавиатуры  
-----  
.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:  
    mov eax, msg ; запись адреса выводимого сообщения в `EAX`  
    call sprint ; вызов подпрограммы печати сообщения  
  
    mov ecx, buf1 ; запись адреса переменной в `EAX`  

```

Рисунок 2.17: Редактируем второй файл, чтобы соответствовать заданию

18) Компилируем файл **lab5-2.asm** и проверяем его работу

```
aapashutina@aapashutina-VirtualBox:~/Рабочий стол$ cd  
aapashutina@aapashutina-VirtualBox:~$ cd ~/work/arch-pc/lab05  
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab05-2.asm  
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab05-2 lab05-2.o  
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$ ./lab05-2  
Введите строку: Pashutina Anna  
Pashutina Anna  
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab05$
```

Рисунок 2.18: Компиляция и проверка работы отредактированного файла **lab5-2.asm**

3 Выводы

Мы познакомились с простыми командами для ввода и вывода слов на языке
Ассемблер