

# **Отчёта по лабораторной работе 4**

**НММбд-04-24**

Ракутуманандзара Цантамписедрана Сарубиди

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>13</b>
	<b>Список литературы</b>	<b>14</b>

# Список иллюстраций

4.1	рис 1 . . . . .	8
4.2	рис 2 . . . . .	8
4.3	рис3 . . . . .	9
4.4	рис4 . . . . .	9
4.5	рис 5 . . . . .	9
4.6	рис 6 . . . . .	10
4.7	рис 7 . . . . .	10
4.8	рис 8 . . . . .	10
4.9	рис 9 . . . . .	10
4.10	рис 10 . . . . .	11
4.11	рис 11 . . . . .	11
4.12	рис 12 . . . . .	11
4.13	рис 13 . . . . .	11
4.14	рис 14 . . . . .	12
4.15	рис 15 . . . . .	12

## **Список таблиц**

# 1 Цель работы

Целью данной лабораторной работы является освоение процедуры оформления отчетов с помощью легковесного языка разметки Markdown.

## 2 Задание

1. Установка необходимого ПО
2. Заполнение отчета по выполнению лабораторной работы №4 с помощью языка разметки Markdown
3. Задание для самостоятельной работы

## 3 Теоретическое введение

Markdown - легковесный язык разметки, созданный с целью обозначения форматирования в простом тексте, с максимальным сохранением его читаемости человеком, и пригодный для машинного преобразования в языки для продвинутых публикаций. Внутритекстовые формулы делаются аналогично формулам LaTeX. В Markdown вставить изображение в документ можно с помощью непосредственного указания адреса изображения. Синтаксис Markdown для встроенной ссылки состоит из части [link text], представляющей текст гиперссылки, и части (file-name.md) – URL-адреса или имени файла, на который дается ссылка. Markdown поддерживает как встраивание фрагментов кода в предложение, так и их размещение между предложениями в виде отдельных огражденных блоков. Огражденные блоки кода — это простой способ выделить синтаксис для фрагментов кода.

## 4 Выполнение лабораторной работы

### ##1 Создание программы Hello world!

я буду использовать `mkdir` для создания указанного каталога, а затем с помощью команды `cd` войду в каталог, в котором буду работать, там я создам пустой текстовый файл «hello.asm» с помощью команды `touch` (рис 1).

```
tsanta@tsanta-VirtualBox:~$ mkdir -p ~/work/os-intro/lab04
tsanta@tsanta-VirtualBox:~$ cd ~/work/os-intro/lab04
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ touch hello.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$
```

Рис. 4.1: рис 1

Я открываю созданный файл в текстовом редакторе и заполню файл, вставив в него программу вывода «Hello word!» (рис 2).

```
; hello.asm
SECTION .data                ; Начало секции данных
    hello:DB 'Hello world!',10 ; 'Hello world!' плюс
                                ; символ перевода строки
    helloLen: EQU $-hello     ; Длина строки hello

SECTION .text                ; Начало секции кода
    GLOBAL _start

_start:                      ; Точка входа в программу
    mov eax,4                ; Системный вызов для записи (sys_write)
    mov ebx,1                ; Описатель файла '1' - стандартный вывод
    mov ecx,hello            ; Адрес строки hello в ecx
    mov edx,helloLen         ; Размер строки hello
    int 80h                  ; Вызов ядра
```

Рис. 4.2: рис 2

### ##2 Транслятором NASM



Я преобразую текст программы в вывод «Hello world!» в объектный код с помощью транслятора NASM с помощью команды `nasm -f elf hello.asm`. Далее проверяю правильность выполнения команды с помощью команды `ls` и, как показано, файл `hello.o` создан (рис3).

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ nasm -f elf hello.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ ls
```

Рис. 4.3: рис3

### ##3 Расширенным синтаксисом командной строки NASM

Я введу команду, которая скомпилирует файл `hello.asm` в файл `obj.o`, и файл будет содержать символы отладки (переключатель `-g`), также используя переключатель `-l` будет создан файл листинга `list.lst`. Далее проверяю с помощью команды `ls` правильность выполнения команды (рис 4)

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ nasm -o obj.o -f elf -g -l list.
lst hello.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ ls
hello.asm list.lst obj.o
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$
```

Рис. 4.4: рис4

### ##4 компоновщиком LD

Я передам объектный файл `hello.o` для обработки компоновщиком LD для создания исполняемого файла `hello`. Далее я использую команду `ls`, чтобы проверить правильность выполнения команды (рис 5).

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ ld -m elf_i386 hello.o -o hello
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ ls
hello hello.asm hello.o list.lst obj.o
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$
```

Рис. 4.5: рис 5

Теперь я создам файл с именем «`main`» с помощью данной команды. Объектный файл, из которого собран этот исполняемый файл, имеет имя `obj.o` (рис 6)

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ ld -m elf_i386 obj.o -o main
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$
```

Рис. 4.6: рис 6

### ##5 Запуск исполняемого файла

Я запущу созданный исполняемый файл hello(рис 7)

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ ./hello
Hello world!
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$
```

Рис. 4.7: рис 7

### ##6 Выполнение заданий для самостоятельной работы.

С помощью команды `cp` я создам копию файла `hello.asm` в текущем каталоге с именем `lab4.asm`(рис 8).

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ cp hello.asm lab4.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$
```

Рис. 4.8: рис 8

С помощью текстового редактора я открою файл `lab4.asm` и внесу изменения в программу, чтобы она отображала мое имя и фамилию(рис 9)

```
; lab4.asm
SECTION .data
    lab4:DB 'Цанта Ракуту',10 ; Начало секции данных
                                ; 'Цанта Ракуту' плюс
                                ; символ перевода строки
    lab4Len: EQU $-lab4 ; Длина строки hello

SECTION .text
    GLOBAL _start ; Начало секции кода

_start:
    ; Точка входа в программу
    mov eax,4 ; Системный вызов для записи (sys_write)
    mov ebx,1 ; Описатель файла '1' - стандартный вывод
    mov ecx,lab4 ; Адрес строки lab4 в ecx
    mov edx,lab4Len ; Размер строки lab4
    int 80h ; Вызов ядра
```

Рис. 4.9: рис 9

Текст программы я скомпилирую в объектный файл. С помощью команды `ls` я проверяю, что файл `lab5.o` создан(рис 10)

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ nasm -f elf lab4.asm
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o
```

Рис. 4.10: рис 10

Я передам объектный файл `lab5.o` компоновщику `LD` для обработки для создания исполняемого файла `lab5`(рис 11).

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ ld -m elf_i386 lab4.o -o lab4
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$
```

Рис. 4.11: рис 11

Когда я запускаю исполняемый файл `lab4`, мое имя и фамилия фактически отображаются на экране(рис 12)

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ ./lab4
Цанта Ракуту
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$
```

Рис. 4.12: рис 12

Я скопирую файлы `hello.asm` и `lab4.asm` в свой локальный репозиторий в каталоге `~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/report`. Я сделаю это с помощью команды `cd`(рис 13)

```
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ cp hello.asm ~/work/study/2024-2025/"Операционные системы"/os-intro/labs/lab04/report
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ cp lab4.asm ~/work/study/2024-2025/"Операционные системы"/os-intro/labs/lab04/report
tsanta@tsanta-VirtualBox:~/work/os-intro/lab04$ cd ~/work/study/2024-2025/"Операционные системы"/os-intro/labs/lab04/report
tsanta@tsanta-VirtualBox:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab04/report$ ls
bib  hello.asm  image  lab4.asm  Makefile  pandoc  report.md
tsanta@tsanta-VirtualBox:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab04/report$
```

Рис. 4.13: рис 13

Используя команду `git add` и `git commit`, я добавлю файлы в `github`(рис 14)

```
tsanta@tsanta-VirtualBox:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab04/report$ git add .
tsanta@tsanta-VirtualBox:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab04/report$ git commit -m "Add files"
[master 81689d2] Add files
17 files changed, 80 insertions(+), 30 deletions(-)
create mode 100644 labs/lab04/report/hello.asm
```

Рис. 4.14: рис 14

Я отправляю файлы на сервер с помощью команды git push(рис 15)

```
tsanta@tsanta-VirtualBox:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab04/report$ git push
Enumerating objects: 28, done.
```

Рис. 4.15: рис 15

## **5 Выводы**

При выполнении данной лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

# **Список литературы**

Архитектура ЭВМ