

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

### ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4 \_\_\_\_\_

дисциплина:     Архитектура компьютера

Студент: Мамова Эрланда

Группа:    НКАбд-04-25

МОСКВА

2025 г.

## Содержание

Снимки экрана.	3
1. Цель работы.	4
2. Порядок выполнения лабораторной работы	4
2.1. Программа Hello world!	4
2.2. Транслятор NASM	5
2.3. Расширенный синтаксис командной строки NASM	6
2.4. компоновщик LD	7
2.5. Запуск исполняемого файла	7
2.6. Задание для самостоятельной работы.	8
3. Вывод	9

## Снимки экрана.

1. Создание текстового файла.	5
2. Текст файла.	5
3. Компиляция текста	6
4. Компиляция файла hello.asm в obj.o	6
5. Передача файла на обработку компоновщику.	7
6. Передача файла на обработку компоновщику 2.	7
7. Запуск файла.	8
8. Копирование файла.	8
9. Редактор.	8
10. Изменение файла.	8
11. Вывод ФИ с помощью исполняемого файла.	9
12. Копирование файлов в репозиторий	9
13. Загрузка файлов на Github	9

## 1. Цель работы.

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

## 2. Порядок выполнения лабораторной работы

### 2.1. Программа Hello world!

Рассмотрим пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит приветственное сообщение Hello world! на экран.

Создайте каталог для работы с программами на языке ассемблера NASM:

```
mkdir -p ~/work/arch-pc/lab04
```

Перейдите в созданный каталог

```
cd ~/work/arch-pc/lab04
```

Создайте текстовый файл с именем hello.asm

```
touch hello.asm
```

откройте этот файл с помощью любого текстового редактора, например, gedit

```
gedit hello.asm
```

и введите в него следующий текст:

```
; 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
```

```
mov eax,1 ; Системный вызов для выхода (sys_exit)
```

mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)

int 80h ; Вызов ядра

В отличие от многих современных высокоуровневых языков программирования, в ассемблерной программе каждая команда располагается на отдельной строке. Размещение нескольких команд на одной строке недопустимо.

Создала каталог для работы с программами на языке ассемблера NASM и перешла в него. Создала текстовый файл с именем hello.asm и открыла этот файл с помощью текстового редактора gedit

```
etmamova@DESKTOP-I3F0IIH:~$ mkdir -p ~/work/arch-pc/lab04
etmamova@DESKTOP-I3F0IIH:~$ cd ~/work/arch-pc/lab04
etmamova@DESKTOP-I3F0IIH:~/work/arch-pc/lab04$ touch hello.asm
etmamova@DESKTOP-I3F0IIH:~/work/arch-pc/lab04$ gedit hello.asm
Command 'gedit' not found, but can be installed with:
```

1. Создание текстового файла.

ввела в него текст

```
1 ; hello.asm
2 SECTION .data          ; Начало секции данных
3     hello: DB 'Hello world!', 10 ; 'Hello world!' плюс символ перевода строки
4     helloLen: EQU $-hello      ; Длина строки hello
5
6 SECTION .text          ; Начало секции кода
7     GLOBAL _start
8
9 _start:                ; Точка входа в программу
10    mov eax, 4          ; Системный вызов для записи (sys_write)
11    mov ebx, 1          ; Описатель файла '1' - стандартный вывод
12    mov ecx, hello      ; Адрес строки hello в ecx
13    mov edx, helloLen   ; Размер строки hello
14    int 0x80            ; Вызов ядра
15
16    mov ebx, 1          ; Системный вызов для записи (sys_write)
```

2. Текст программы.

## 2.2. Транслятор NASM

NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World» необходимо написать:

```
nasm -f elf hello.asm
```

Если текст программы набран без ошибок, то транслятор преобразует текст программы из файла hello.asm в объектный код, который запишется в файл hello.o.

Таким образом, имена всех файлов получаются из имени входного файла и расширения по умолчанию. При наличии ошибок объектный файл не создаётся, а после запуска транслятора появятся сообщения об ошибках или предупреждения. С помощью команды `ls` проверьте, что объектный файл был создан. Какое имя имеет объектный файл? NASM не запускают без параметров. Ключ `-f` указывает транслятору, что требуется создать бинарные файлы в формате ELF. Следует отметить, что формат `elf64` позволяет создавать исполняемый код, работающий под 64-битными версиями Linux. Для 32-битных версий ОС указываем в качестве формата просто `elf`. NASM всегда создаёт выходные файлы в текущем каталоге.

Компилировала приведённый выше текст программы «Hello World» и проверила, что объектный файл был создан. Имя объектного файла «hello.o».

```
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ nasm -f elf hello.asm
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ ls
hello.asm hello.o
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ |
```

### 3. Компиляция текста.

#### 2.3. Расширенный синтаксис командной строки NASM

Полный вариант командной строки `nasm` выглядит следующим образом:

`nasm` `[-@` косвенный\_файл\_настроек `]` `[-o` объектный\_файл `]` `[-f` ↵  
формат\_объектного\_файла `]` `[-l` листинг `]` `[параметры...]` `[--]` исходный\_файл

Выполните следующую команду:

```
nasm -o obj.o -f elf -g -l list.lst hello.asm
```

Данная команда скомпилирует исходный файл `hello.asm` в `obj.o` (опция `-o` позволяет задать имя объектного файла, в данном случае `obj.o`), при этом формат выходного файла будет `elf`, и в него будут включены символы для отладки (опция `-g`), кроме того, будет создан файл листинга `list.lst` (опция `-l`). С помощью команды `ls` проверьте, что файлы были созданы. Для более подробной информации см. `man nasm`. Для получения списка форматов объектного файла см. `nasm -hf`.

Скомпилировала исходный файл `hello.asm` в `obj.o` с форматом `elf` и проверила, что файлы были созданы.

```
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ ls
hello.asm hello.o list.lst obj.o
```

### 4. Компиляция файла `hello.asm` в `obj.o`

## 2.4. Компоновщик LD

Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику:

```
ld -m elf_i386 hello.o -o hello
```

С помощью команды `ls` проверьте, что исполняемый файл `hello` был создан. Компоновщик `ld` не предполагает по умолчанию расширений для файлов, но принято использовать следующие расширения: • `o` – для объектных файлов; • без расширения – для исполняемых файлов; • `map` – для файлов схемы программы; • `lib` – для библиотек. Ключ `-o` с последующим значением задаёт в данном случае имя создаваемого исполняемого файла.

Выполните следующую команду:

```
ld -m elf_i386 obj.o -o main
```

Какое имя будет иметь исполняемый файл? Какое имя имеет объектный файл из которого собран этот исполняемый файл?

Передала программу на обработку компоновщику и проверила, что исполняемый файл `hello` был создан

```
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst obj.o
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ |
```

## 5. Передача файла на обработку компоновщику

Выполнила указанную команду.

```
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst main obj.o
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ |
```

## 6. Передача файла на обработку компоновщику 2

Имя исполняемого файла `main`, а имя объектного файла `obj.o`.

## 2.5. Запуск исполняемого файла

Запустить на выполнение созданный исполняемый файл, находящийся в текущем каталоге, можно, набрав в командной строке:

```
./hello
```

Запустила на выполнение созданный исполняемый файл.

```
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ ./hello
Hello world!
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ |
```

#### 7. Запуск файла.

### 2.6. Задание для самостоятельной работы.

1. В каталоге ~/work/arch-pc/lab04 с помощью команды `cp` создайте копию файла `hello.asm` с именем `lab4.asm`

Создала копию файла `hello.asm` с именем `lab4.asm` в каталоге ~/work/arch-pc/lab04

```
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ |
```

#### 8. Копирование файла.

2. С помощью любого текстового редактора внесите изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с вашими фамилией и именем.

С помощью текстового редактора внесла изменения в текст программы в файле `lab4.asm`.

```
etmamova@DESKTOP-I3F0I1H:~/work/arch-pc/lab04$ nano lab4.asm
```

#### 9. Редактор.

```
GNU nano 7.2 lab4.asm
; hello.asm
SECTION .data ; Начало секции данных
hello: DB 'Mamova Erlanda', 10|
hello: EQU $-hello
```

#### 10. Изменение файла

3. Оттранслируйте полученный текст программы `lab4.asm` в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл.

Оттранслировала полученный текст программы `lab4.asm` в объектный файл, выполните компоновку объектного файла и запустила получившийся исполняемый файл.



```

etmamova@DESKTOP-I3F0IIH:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
etmamova@DESKTOP-I3F0IIH:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
etmamova@DESKTOP-I3F0IIH:~/work/arch-pc/lab04$ ./lab4
Mamova ErLanda
etmamova@DESKTOP-I3F0IIH:~/work/arch-pc/lab04$ |

```

11. Вывод ФИ с помощью исполняемого файла.

4. Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study/2025-2026/"Архитектура компьютера"/arch-pc/labs/lab04/. Загрузите файлы на Github.

Скопировала файлы hello.asm и lab4.asm в свой локальный репозиторий

```

etmamova@DESKTOP-I3F0IIH:~/work/study/study_2025-2026_arch-pc$ cp ~/work/arch-pc/lab04/hello.asm labs/lab04/
etmamova@DESKTOP-I3F0IIH:~/work/study/study_2025-2026_arch-pc$ cp ~/work/arch-pc/lab04/lab4.asm labs/lab04/
etmamova@DESKTOP-I3F0IIH:~/work/study/study_2025-2026_arch-pc$ ls

```

12. Копирование файлов в репозиторий

Загрузила файлы на Github.

```

etmamova@DESKTOP-I3F0IIH:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git add labs/lab04/
etmamova@DESKTOP-I3F0IIH:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git commit -m "Add lab 4: Assembly programs with personal name output"
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
etmamova@DESKTOP-I3F0IIH:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git push

```

13. Загрузка файлов на Github

### 3. Вывод.

Я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.