

# **Отчёт по лабораторной работе 4**

**Архитектура компьютеров и операционные системы**

Абдулфазова Лейла Али гызы

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задания</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Программа Hello world! . . . . .	9
4.2	Транслятор NASM . . . . .	10
4.3	Расширенный синтаксис командной строки NASM . . . . .	11
4.4	Компоновщик LD . . . . .	11
4.5	Запуск исполняемого файла . . . . .	12
4.6	Задание для самостоятельной работы . . . . .	12
<b>5</b>	<b>Выводы</b>	<b>14</b>

## Список иллюстраций

4.1	Создан каталог для работы и файл для программы . . . . .	9
4.2	Редактирование файла hello.asm . . . . .	10
4.3	Трансляция программы . . . . .	11
4.4	Трансляция программы с дополнительными опциями . . . . .	11
4.5	Компоновка программы . . . . .	12
4.6	Компоновка программы . . . . .	12
4.7	Запуск программы . . . . .	12
4.8	Скопировал файл . . . . .	12
4.9	Редактирование файла lab4.asm . . . . .	13
4.10	Тестирование программы lab4.asm . . . . .	13

## Список таблиц

# 1 Цель работы

Целью работы является освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

## 2 Задания

1. Изучить основы языка Ассемблера
2. Изучить и рассмотреть на практике процесс сборки программы
3. Выполнить задание по программе
4. Подготовить отчет и загрузить на GitHub

### 3 Теоретическое введение

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня, таких как C/C++, Perl, Python и пр. Заметим, что получить полный доступ к ресурсам компьютера в современных архитектурах нельзя, самым низким уровнем работы прикладной программы является обращение напрямую к ядру операционной системы. Именно на этом уровне и работают программы, написанные на ассемблере. Но в отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввёл программист. Таким образом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора.

В процессе создания ассемблерной программы можно выделить четыре шага:

- Набор текста программы в текстовом редакторе и сохранение её в отдельном файле. Каждый файл имеет свой тип (или расширение), который определяет назначение файла. Файлы с исходным текстом программ на языке ассемблера имеют тип asm.
- Трансляция — преобразование с помощью транслятора, например nasm, текста программы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную дополнительную информацию, созданную транслятором. Тип объектного файла — o, файла листинга — lst.

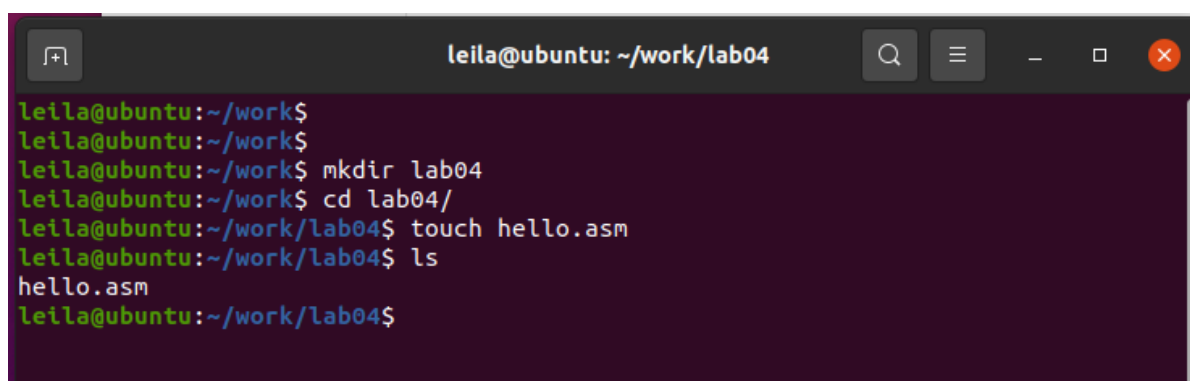
- Компоновка или линковка — этап обработки объектного кода компоновщиком (ld), который принимает на вход объектные файлы и собирает по ним исполняемый файл. Исполняемый файл обычно не имеет расширения. Кроме того, можно получить файл карты загрузки программы в ОЗУ, имеющий расширение map.
- Запуск программы.



## 4 Выполнение лабораторной работы

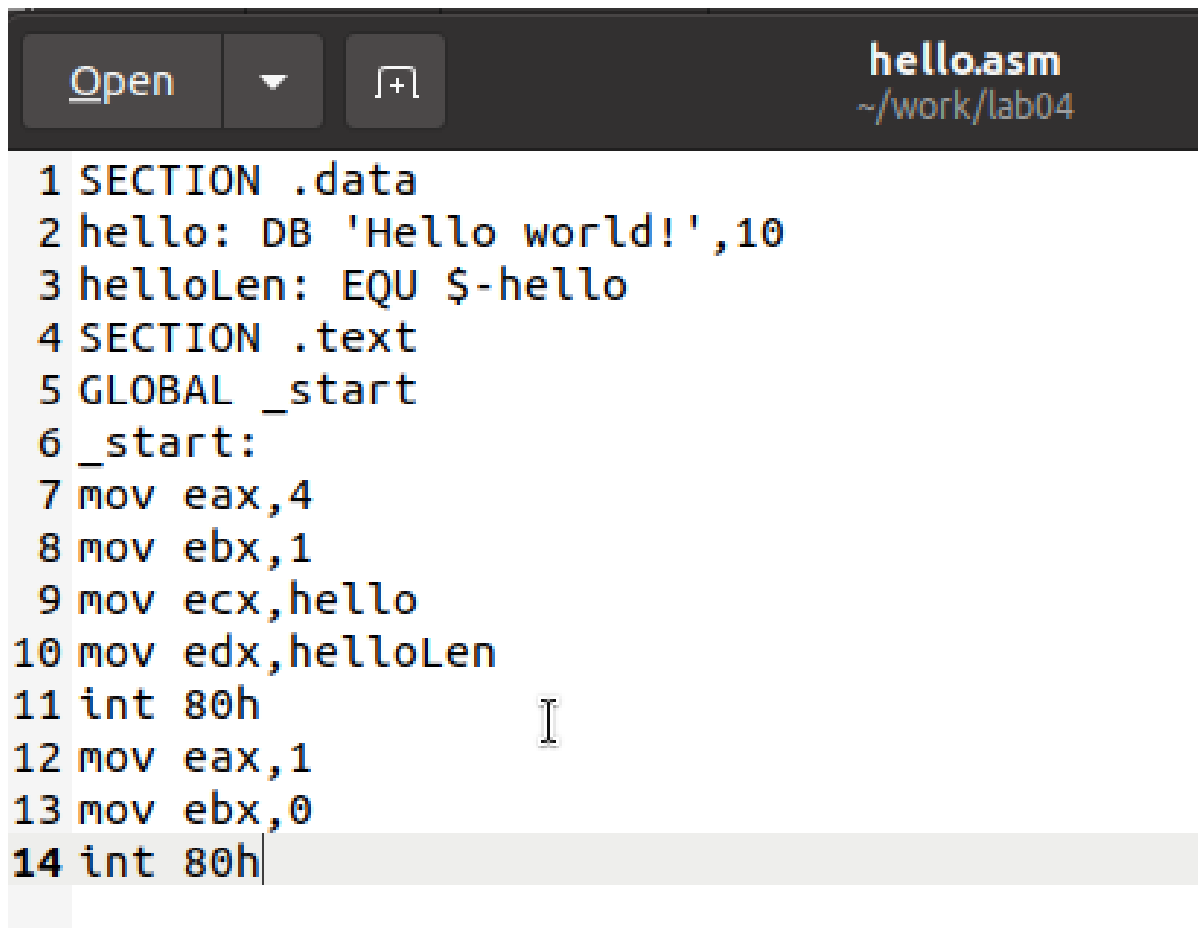
### 4.1 Программа Hello world!

Создала каталог с названием lab04 используя команду `mkdir`. Затем перешла в созданный каталог, воспользовавшись командой `cd`. Внутри каталога создала файл с именем `hello.asm`, где написала программу.

A screenshot of a terminal window with a dark purple background. The window title bar shows 'leila@ubuntu: ~/work/lab04'. The terminal text shows the following sequence of commands and output:

```
leila@ubuntu:~/work$  
leila@ubuntu:~/work$  
leila@ubuntu:~/work$ mkdir lab04  
leila@ubuntu:~/work$ cd lab04/  
leila@ubuntu:~/work/lab04$ touch hello.asm  
leila@ubuntu:~/work/lab04$ ls  
hello.asm  
leila@ubuntu:~/work/lab04$
```

Рис. 4.1: Создан каталог для работы и файл для программы



```
1 SECTION .data
2 hello: DB 'Hello world!',10
3 helloLen: EQU $-hello
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax,4
8 mov ebx,1
9 mov ecx,hello
10 mov edx,helloLen
11 int 80h
12 mov eax,1
13 mov ebx,0
14 int 80h
```

Рис. 4.2: Редактирование файла hello.asm

## 4.2 Транслятор NASM

Использовала транслятор NASM, чтобы преобразовать текст программы в объектный код. Если текст программы был написан без ошибок, то транслятор преобразовал текст программы из файла hello.asm в объектный код и сохраняет его в файле hello.o.

Транслировала файл с помощью команды `nasm` и получила объектный файл hello.o.

```
leila@ubuntu:~/work/lab04$ nasm -f elf hello.asm
leila@ubuntu:~/work/lab04$ ls
hello.asm  hello.o
leila@ubuntu:~/work/lab04$
```

Рис. 4.3: Трансляция программы

## 4.3 Расширенный синтаксис командной строки NASM

Полный вариант командной строки для трансляции с использованием NASM выглядит следующим образом

```
nasm [-@ косвенный_файл_настроек] [-o объектный_файл] [-f формат_объектного_файла] [-l листинг] [параметры...] [--] исходный_файл
```

Транслировала файл с помощью команды `nasm` и использовала дополнительные опции. С опцией `-l` получила файл листинга с именем `list.lst`, с опцией `-f` получила объектный файл с именем `obj.o`, а с опцией `-g` в программу добавилась отладочная информация.

```
leila@ubuntu:~/work/lab04$
leila@ubuntu:~/work/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
leila@ubuntu:~/work/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
leila@ubuntu:~/work/lab04$
```

Рис. 4.4: Трансляция программы с дополнительными опциями

## 4.4 Компоновщик LD

Для получения исполняемой программы, объектный файл нужно передать на обработку компоновщику.

Выполнила команду `ld` и получила исполняемый файл с именем `hello` из объектного файла `hello.o`.

```
leila@ubuntu:~/work/lab04$ ld -m elf_i386 hello.o -o hello
leila@ubuntu:~/work/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o
leila@ubuntu:~/work/lab04$
```

Рис. 4.5: Компоновка программы

Еще раз выполнена команда `ld` для объектного файла `obj.o` и получен исполняемый файл с именем `main`

```
leila@ubuntu:~/work/lab04$
leila@ubuntu:~/work/lab04$ ld -m elf_i386 obj.o -o main
leila@ubuntu:~/work/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
leila@ubuntu:~/work/lab04$
```

Рис. 4.6: Компоновка программы

## 4.5 Запуск исполняемого файла

Запустила исполняемые файлы.

```
leila@ubuntu:~/work/lab04$
leila@ubuntu:~/work/lab04$ ./hello
Hello world!
leila@ubuntu:~/work/lab04$ ./main
Hello world!
leila@ubuntu:~/work/lab04$
```

Рис. 4.7: Запуск программы

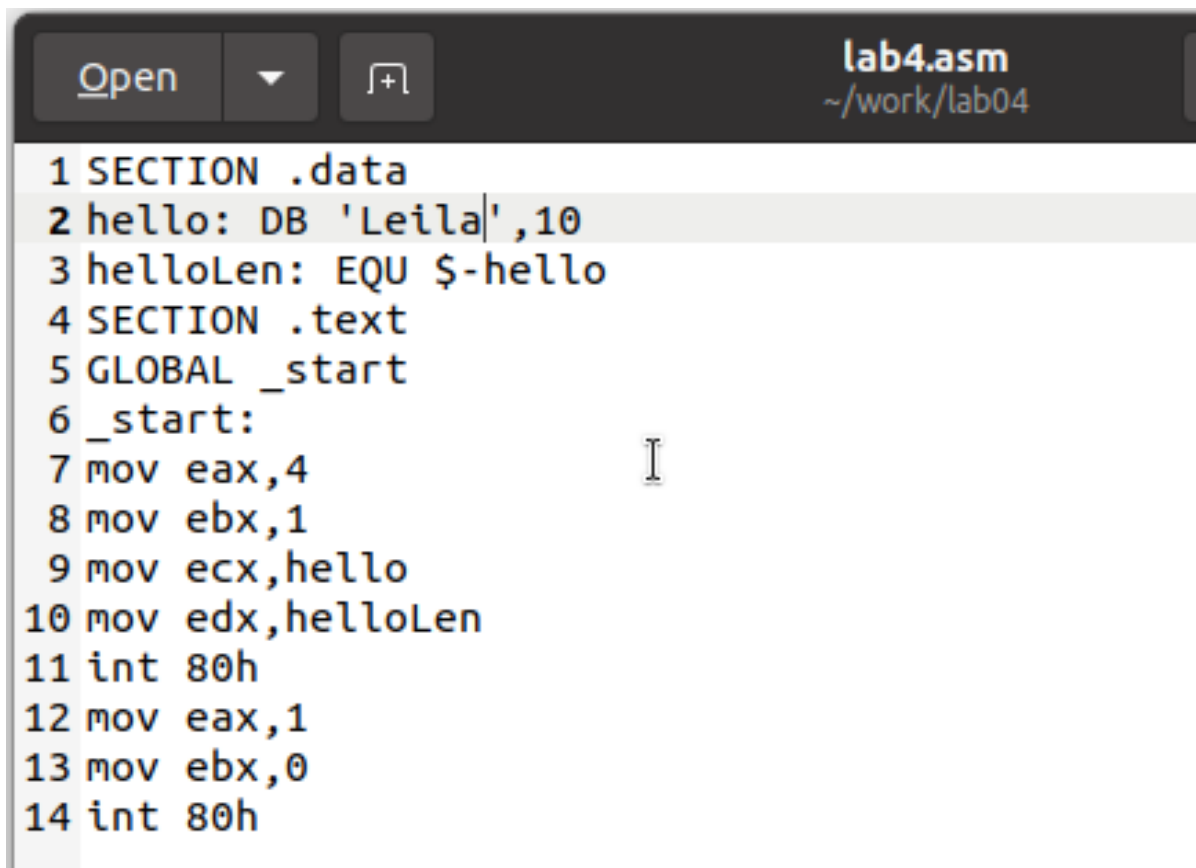
## 4.6 Задание для самостоятельной работы

Скопировала файл `hello.asm` в файл `lab4.asm`.

```
leila@ubuntu:~/work/lab04$
leila@ubuntu:~/work/lab04$ cp hello.asm lab4.asm
leila@ubuntu:~/work/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
leila@ubuntu:~/work/lab04$
```

Рис. 4.8: Скопировал файл

Изменила сообщение Hello world на свое имя.

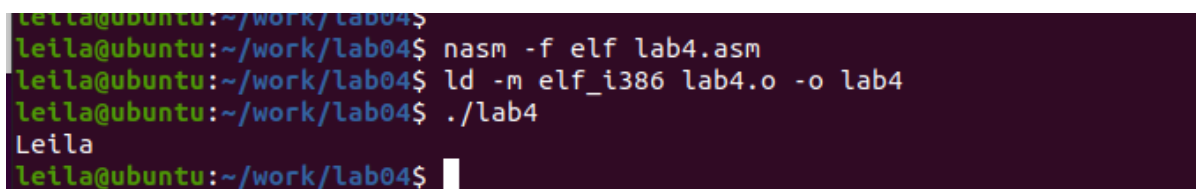


```
lab4.asm
~/work/lab04

1 SECTION .data
2 hello: DB 'Leila',10
3 helloLen: EQU $-hello
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax,4
8 mov ebx,1
9 mov ecx,hello
10 mov edx,helloLen
11 int 80h
12 mov eax,1
13 mov ebx,0
14 int 80h
```

Рис. 4.9: Редактирование файла lab4.asm

Запустила программу и проверила.



```
leila@ubuntu:~/work/lab04$
leila@ubuntu:~/work/lab04$ nasm -f elf lab4.asm
leila@ubuntu:~/work/lab04$ ld -m elf_i386 lab4.o -o lab4
leila@ubuntu:~/work/lab04$ ./lab4
Leila
leila@ubuntu:~/work/lab04$
```

Рис. 4.10: Тестирование программы lab4.asm

## 5 Выводы

Освоил процесс компиляции и сборки программ, написанных на ассемблере `nasm`.