

# **Отчёт по лабораторной работе 4**

**Архитектура компьютеров и операционные системы**

Дрожжанова А.Д. НБИбд-01-23

# 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	Выполнение заданий для самостоятельной работы. . . . .	12
<b>5</b>	<b>Выводы</b>	<b>14</b>
<b>6</b>	<b>Источники</b>	<b>15</b>

## Список иллюстраций

4.1	Создание файла . . . . .	9
4.2	Программа hello.asm . . . . .	10
4.3	Трансляция программы . . . . .	11
4.4	Компановка программы . . . . .	11
4.5	Запуск программы . . . . .	12
4.6	Копирование файла . . . . .	12
4.7	Программа lab4.asm . . . . .	13
4.8	Проверка 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.

Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — машинные коды. До появления языков ассемблера программистам приходилось писать программы, используя только лишь машинные коды, которые были крайне сложны для запоминания, так как представляли собой числа, записанные в двоичной или шестнадцатеричной системе счисления. Преобразование или трансляция команд с языка ассемблера в исполняемый машинный код осуществляется специальной программой транслятором — Ассемблер

Программы, написанные на языке ассемблера, не уступают в качестве и скорости программам, написанным на машинном языке, так как транслятор

просто переводит мнемонические обозначения команд в последовательности бит (нулей и единиц).

В нашем курсе будет использоваться ассемблер NASM (Netwide Assembler). NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64. Типичный формат записи команд NASM имеет вид:

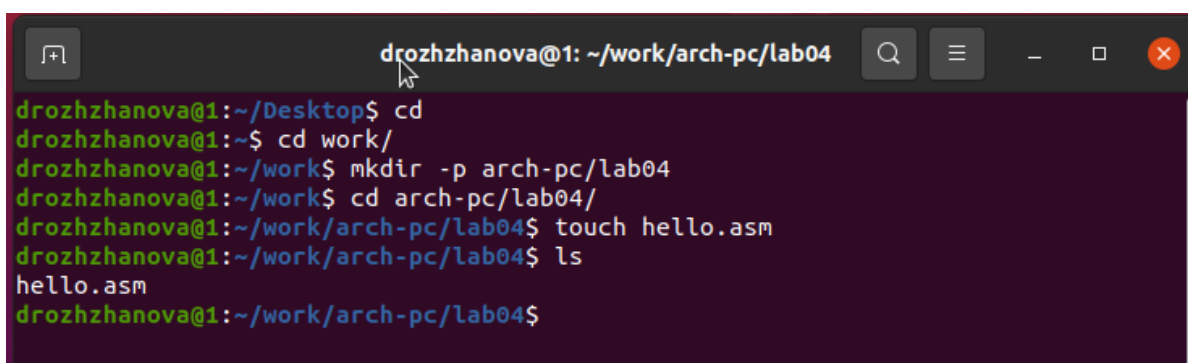
[метка:] мнемокод [операнд {, операнд}] [; комментарий]

Здесь мнемокод — непосредственно мнемоника инструкции процессору, которая является обязательной частью команды. Операндами могут быть числа, данные, адреса регистров или адреса оперативной памяти. Метка — это идентификатор, с которым ассемблер ассоциирует некоторое число, чаще всего адрес в памяти. Т.о. метка перед командой связана с адресом данной команды.



## 4 Выполнение лабораторной работы

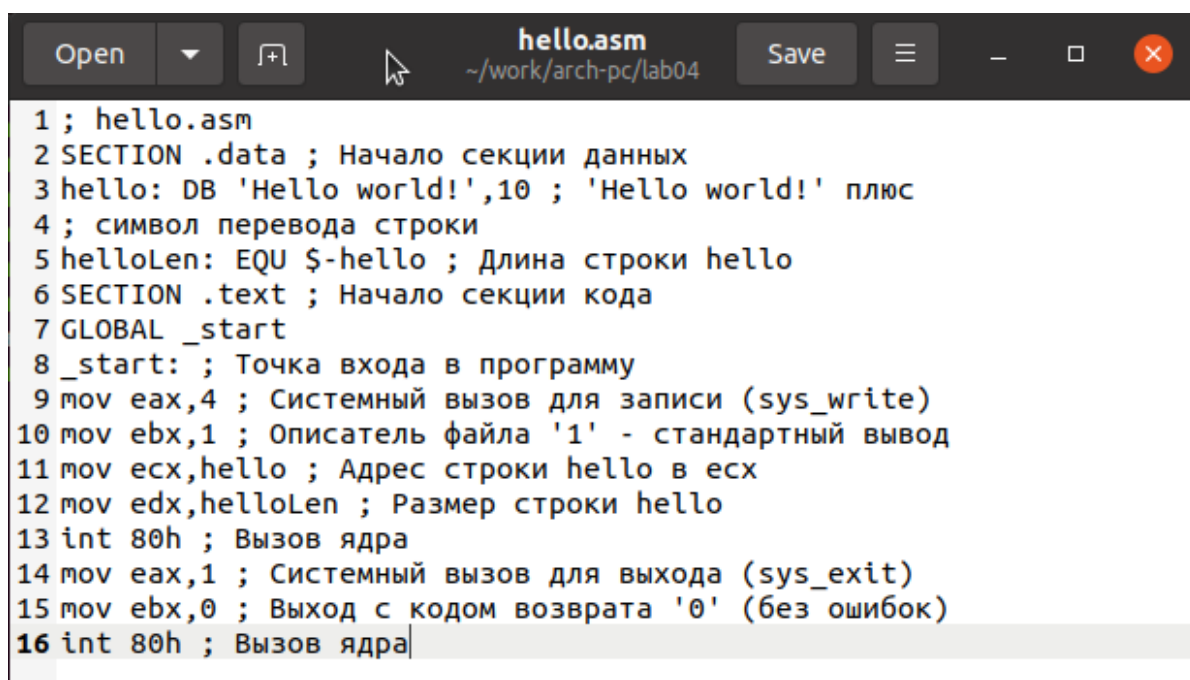
1. Создала каталог lab04 командой mkdir, перешла в него с помощью команды cd, создала файл hello.asm.



```
drozhzhanova@1: ~/work/arch-pc/lab04
drozhzhanova@1:~/Desktop$ cd
drozhzhanova@1:~$ cd work/
drozhzhanova@1:~/work$ mkdir -p arch-pc/lab04
drozhzhanova@1:~/work$ cd arch-pc/lab04/
drozhzhanova@1:~/work/arch-pc/lab04$ touch hello.asm
drozhzhanova@1:~/work/arch-pc/lab04$ ls
hello.asm
drozhzhanova@1:~/work/arch-pc/lab04$
```

Рис. 4.1: Создание файла

2. Открыла файл в gedit и написала код программы по заданию.



```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для записи (sys_write)
10 mov ebx,1 ; Описатель файла '1' - стандартный вывод
11 mov ecx,hello ; Адрес строки hello в ecx
12 mov edx,helloLen ; Размер строки hello
13 int 80h ; Вызов ядра
14 mov eax,1 ; Системный вызов для выхода (sys_exit)
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16 int 80h ; Вызов ядра
```

Рис. 4.2: Программа hello.asm

```
; 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

3. Транслировала файл командой `nasm` с опцией `-f`. Ключ `-f` указывает транслятору, что требуется создать бинарные файлы в формате ELF.

Получился объектный файл `hello.o`

4. Транслировала файл командой `nasm` с дополнительными опциями : `-o`, `-g`, `-l` Опция `-o` позволяет задать имя объектного файла. Опция `-g` добавляет отладочную информацию. Опция `-l` создает файл листинг.

Получился файл листинга `list.lst`, объектный файл `obj.o`, в программу добавилась отладочная информация.

```
drozhzhanova@1:~/work/arch-pc/lab04$ gedit hello.asm
drozhzhanova@1:~/work/arch-pc/lab04$ nasm -f elf hello.asm
drozhzhanova@1:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
drozhzhanova@1:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.a
sm
drozhzhanova@1:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
drozhzhanova@1:~/work/arch-pc/lab04$
```

Рис. 4.3: Трансляция программы

5. Выполнила компоновку командой `ld` и получила исполняемый файл.
6. Еще раз выполнила компоновку для объектного файла `obj.o` и получила исполняемый файл `main`.

```
drozhzhanova@1:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
drozhzhanova@1:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o
drozhzhanova@1:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
drozhzhanova@1:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
drozhzhanova@1:~/work/arch-pc/lab04$
```

Рис. 4.4: Компоновка программы

7. Запустила исполняемые файлы.

```
drozhzhanova@1:~/work/arch-pc/lab04$  
drozhzhanova@1:~/work/arch-pc/lab04$ ./hello  
Hello world!  
drozhzhanova@1:~/work/arch-pc/lab04$ ./main  
Hello world!  
drozhzhanova@1:~/work/arch-pc/lab04$
```

Рис. 4.5: Запуск программы

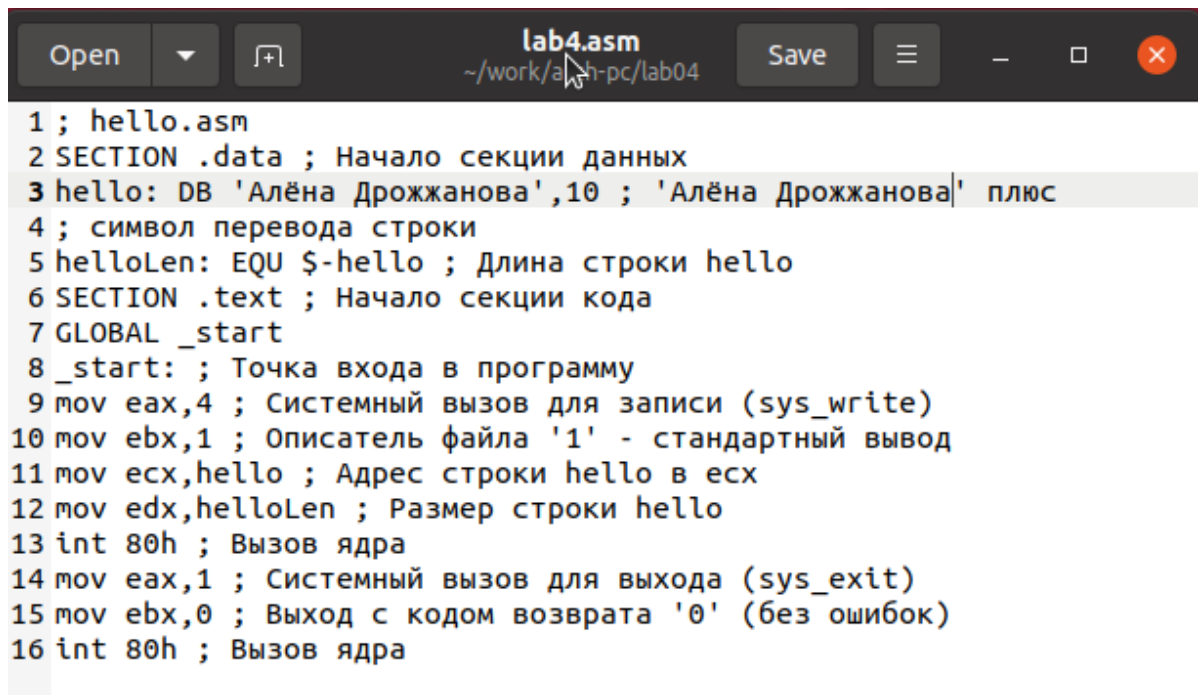
## 4.1 Выполнение заданий для самостоятельной работы.

1. Скопировала программу в файл lab4.asm.

```
drozhzhanova@1:~/work/arch-pc/lab04$  
drozhzhanova@1:~/work/arch-pc/lab04$ cp hello.asm lab4.asm  
drozhzhanova@1:~/work/arch-pc/lab04$ ls  
hello hello.asm hello.o lab4.asm list.lst main obj.o  
drozhzhanova@1:~/work/arch-pc/lab04$
```

Рис. 4.6: Копирование файла

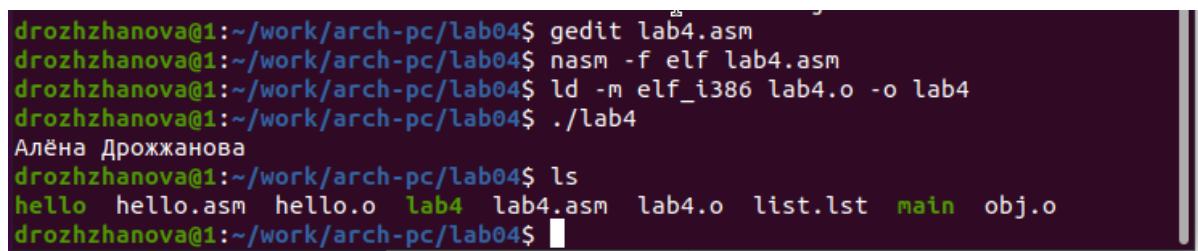
2. Изменила сообщение Hello world на свое имя.



```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Алёна Дрожжанова',10 ; 'Алёна Дрожжанова' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для записи (sys_write)
10 mov ebx,1 ; Описатель файла '1' - стандартный вывод
11 mov ecx,hello ; Адрес строки hello в ecx
12 mov edx,helloLen ; Размер строки hello
13 int 80h ; Вызов ядра
14 mov eax,1 ; Системный вызов для выхода (sys_exit)
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16 int 80h ; Вызов ядра
```

Рис. 4.7: Программа lab4.asm

- Оттранслировала полученный текст программы lab4.asm в объектный файл. Выполнила компоновку объектного файла и запустила получившийся исполняемый файл.



```
drozhzhanova@1:~/work/arch-pc/lab04$ gedit lab4.asm
drozhzhanova@1:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
drozhzhanova@1:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
drozhzhanova@1:~/work/arch-pc/lab04$ ./lab4
Алёна Дрожжанова
drozhzhanova@1:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o
drozhzhanova@1:~/work/arch-pc/lab04$
```

Рис. 4.8: Проверка программы lab4.asm

- Загрузила файлы на github.

## 5 Выводы

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

## **6 Источники**

1. Архитектура ЭВМ - Материалы курса