

# **Лабораторная работа №5**

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

Россохин Олег

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Программа Hello world! . . . . .	7
<b>4</b>	<b>Задание для самостоятельной работы</b>	<b>11</b>
<b>5</b>	<b>Вопросы для самопроверки</b>	<b>13</b>
<b>6</b>	<b>Выводы</b>	<b>16</b>
	<b>Список литературы</b>	<b>17</b>

# Список иллюстраций

3.1	2	. . . . .	8
3.2	3	. . . . .	8
3.3	3	. . . . .	9
3.4	4	. . . . .	9
3.5	4.1	. . . . .	10
3.6	5	. . . . .	10
4.1	1	. . . . .	11
4.2	2	. . . . .	11
4.3	4	. . . . .	12
5.1	пример	. . . . .	14

## Список таблиц

# 1 Цель работы

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

## 2 Теоретическое введение

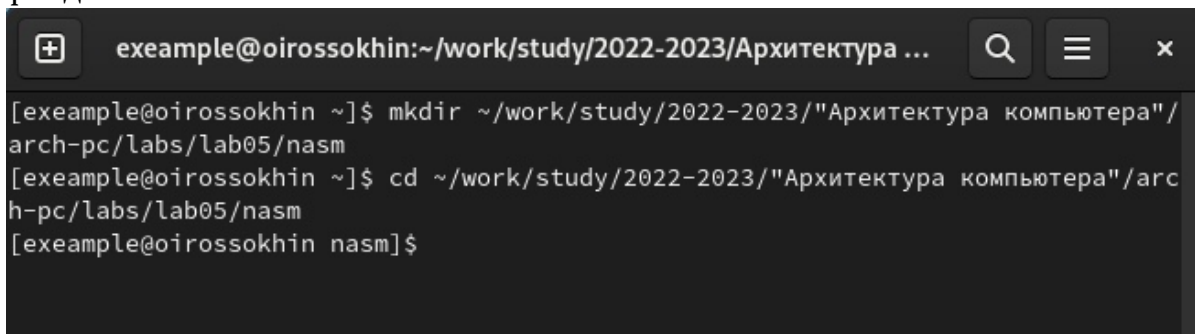
Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства. Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих дорожек на материнской (системной) плате.

Язык ассемблера (assembly language) — представление команд процессора в виде, доступном для чтения человеком. Язык ассемблера считается языком программирования низкого уровня, в противовес высокоуровневым языкам, не привязанным к конкретной реализации вычислительной системы. Программы, написанные на языке ассемблера однозначным образом переводятся в инструкции конкретного процессора и в большинстве случаев не могут быть перенесены без значительных изменений для запуска на машине с другой системой команд. Ассемблером называется программа, преобразующая код на языке ассемблера в машинный код; программа, выполняющая обратную задачу называется дизассемблером.

## 3 Выполнение лабораторной работы

### 3.1 Программа Hello world!

1. Создадим каталог для работы с программами на языке ассемблера NASM и перейдем в него:



```
exeample@oirossokhin:~/work/study/2022-2023/Архитектура ...  
[exeample@oirossokhin ~]$ mkdir ~/work/study/2022-2023/"Архитектура компьютера"/  
arch-pc/labs/lab05/nasm  
[exeample@oirossokhin ~]$ cd ~/work/study/2022-2023/"Архитектура компьютера"/arc  
h-pc/labs/lab05/nasm  
[exeample@oirossokhin nasm]$
```

я создал отдельный каталог nasm в директории lab05.

2. Создадим текстовый файл с именем hello.asm и пропишем в нем следующие команды:

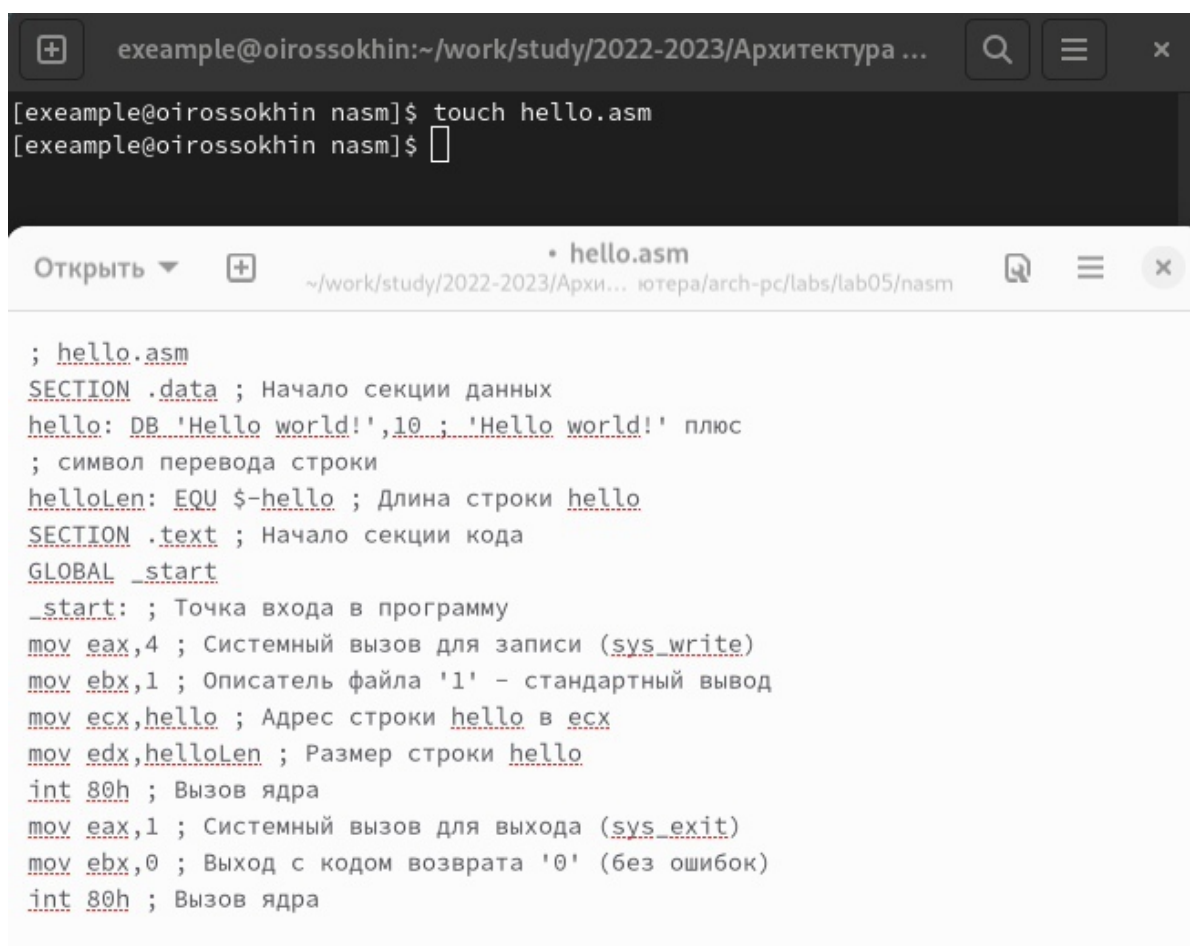


Рис. 3.1: 2

Далее для компиляции файла пропишем в терминале команду а также сразу проверим, создался ли в папке результат компиляции:

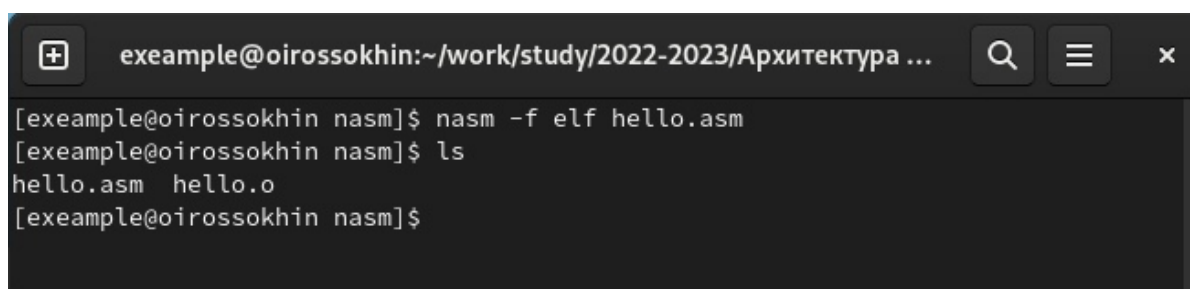
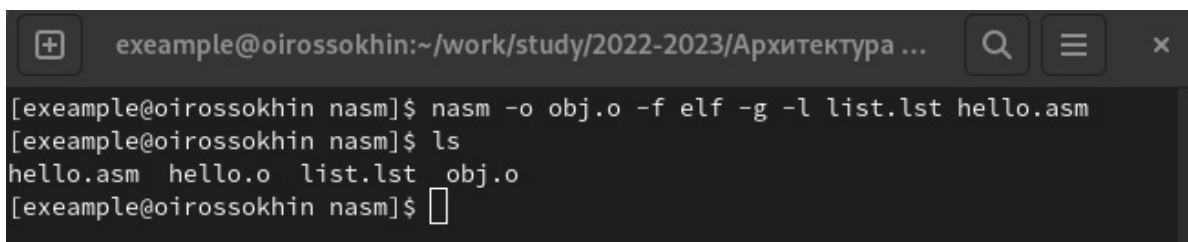


Рис. 3.2: 3

Как мы видим, появился новый файл *hello.o*.



3. Далее выполним следующую команду для компиляции файла *hello.asm* в *obj.o* и проверим, что файлы были созданы:

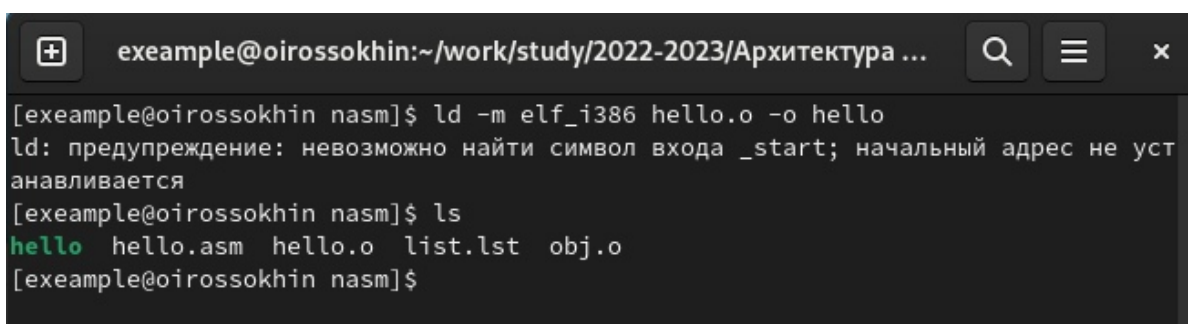


```
exeample@oirossokhin:~/work/study/2022-2023/Архитектура ...  
[exeample@oirossokhin nasm]$ nasm -o obj.o -f elf -g -l list.lst hello.asm  
[exeample@oirossokhin nasm]$ ls  
hello.asm hello.o list.lst obj.o  
[exeample@oirossokhin nasm]$
```

Рис. 3.3: 3

Данна команда скомпилировала файл *hello.asm* в *obj.o* при помощи опции *-o*, при этом формат файла *elf*, в него включены символы для отладки с помощью опции *-g*, а также, создан файл листинга *list.lst* с помощью опции *-l*.

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

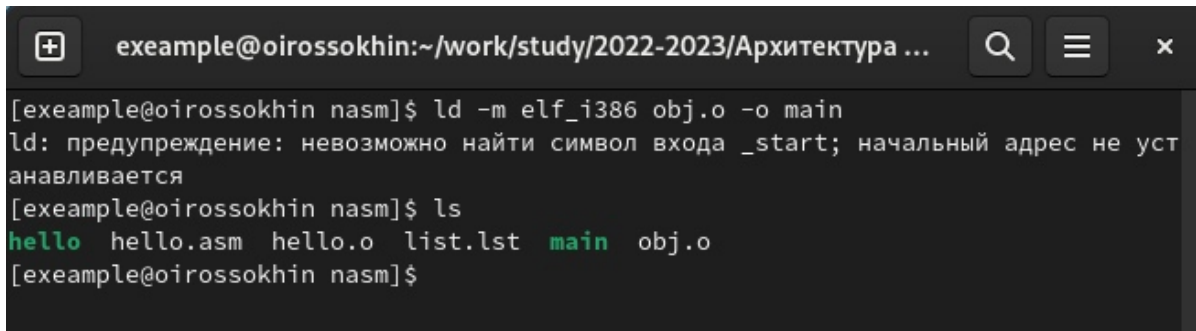


```
exeample@oirossokhin:~/work/study/2022-2023/Архитектура ...  
[exeample@oirossokhin nasm]$ ld -m elf_i386 hello.o -o hello  
ld: предупреждение: невозможно найти символ входа _start; начальный адрес не уст  
анавливается  
[exeample@oirossokhin nasm]$ ls  
hello hello.asm hello.o list.lst obj.o  
[exeample@oirossokhin nasm]$
```

Рис. 3.4: 4

Несмотря на ошибку, файл создался.

Далее выполним команду:

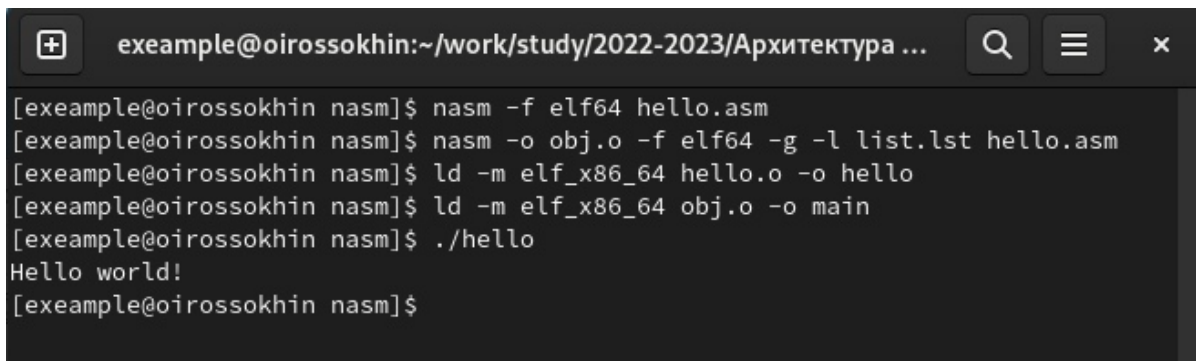
A terminal window with a dark background. The title bar shows the user 'exeample@oirossokhin' and the path '~/work/study/2022-2023/Архитектура ...'. The terminal contains the following text:

```
[exeample@oirossokhin nasm]$ ld -m elf_i386 obj.o -o main
ld: предупреждение: невозможно найти символ входа _start; начальный адрес не устанавливается
[exeample@oirossokhin nasm]$ ls
hello hello.asm hello.o list.lst main obj.o
[exeample@oirossokhin nasm]$
```

Рис. 3.5: 4.1

Видим, что имя исполняемого файла *obj.o*, а объектного файла *main*, потому что перед ним стоит опция *-o*.

5. Позже обратил внимание на разрядность дистрибутива и проделал те же операции, но для 64-бит системы. Запустим созданный исполняемый файл

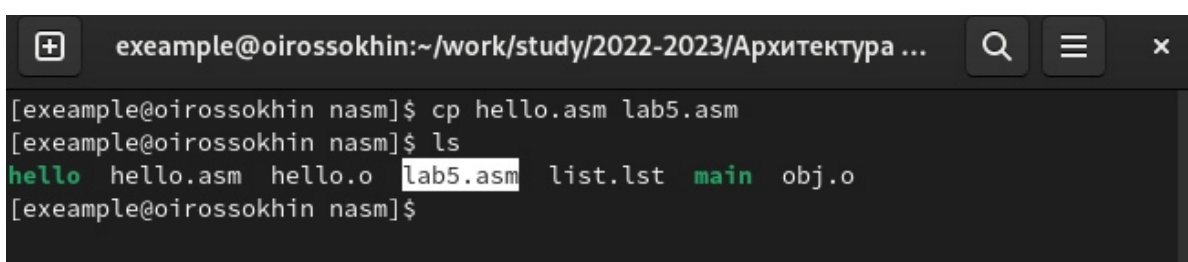
A terminal window with a dark background. The title bar shows the user 'exeample@oirossokhin' and the path '~/work/study/2022-2023/Архитектура ...'. The terminal contains the following text:

```
[exeample@oirossokhin nasm]$ nasm -f elf64 hello.asm
[exeample@oirossokhin nasm]$ nasm -o obj.o -f elf64 -g -l list.lst hello.asm
[exeample@oirossokhin nasm]$ ld -m elf_x86_64 hello.o -o hello
[exeample@oirossokhin nasm]$ ld -m elf_x86_64 obj.o -o main
[exeample@oirossokhin nasm]$ ./hello
Hello world!
[exeample@oirossokhin nasm]$
```

Рис. 3.6: 5

## 4 Задание для самостоятельной работы

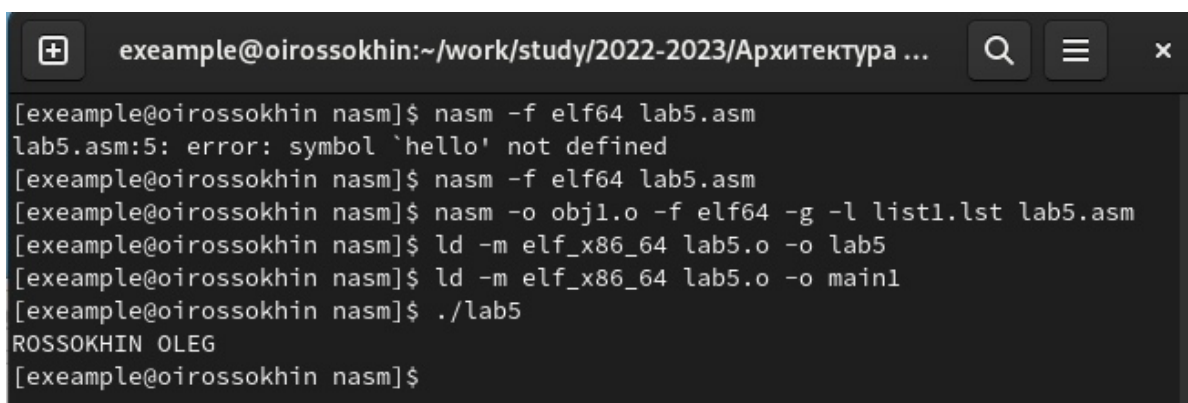
1. В каталоге ~/work/arch-pc/lab05 с помощью команды `cp` создайте копию файла `hello.asm` с именем `lab5.asm`



```
exeample@oirossokhin:~/work/study/2022-2023/Архитектура ...  
[exeample@oirossokhin nasm]$ cp hello.asm lab5.asm  
[exeample@oirossokhin nasm]$ ls  
hello  hello.asm  hello.o  lab5.asm  list.lst  main  obj.o  
[exeample@oirossokhin nasm]$
```

Рис. 4.1: 1

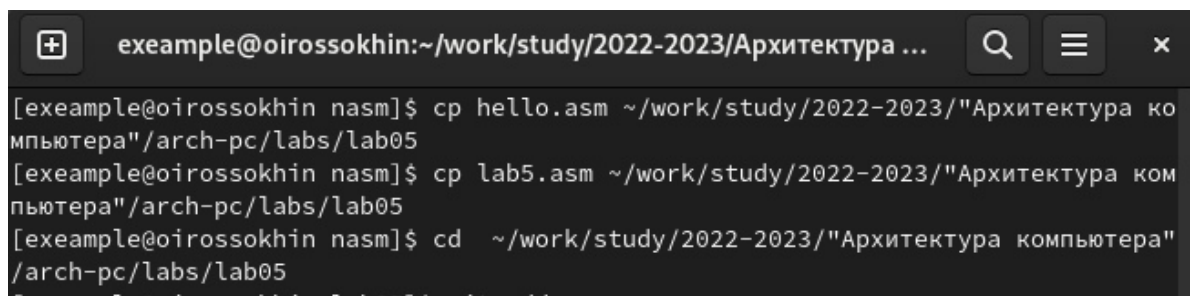
2, 3 С помощью любого текстового редактора внесите изменения в текст программы в файле `lab5.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с вашими фамилией и именем; Оттранслируйте полученный текст программы `lab5.asm` в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл



```
exeample@oirossokhin:~/work/study/2022-2023/Архитектура ...  
[exeample@oirossokhin nasm]$ nasm -f elf64 lab5.asm  
lab5.asm:5: error: symbol `hello' not defined  
[exeample@oirossokhin nasm]$ nasm -f elf64 lab5.asm  
[exeample@oirossokhin nasm]$ nasm -o obj1.o -f elf64 -g -l list1.lst lab5.asm  
[exeample@oirossokhin nasm]$ ld -m elf_x86_64 lab5.o -o lab5  
[exeample@oirossokhin nasm]$ ld -m elf_x86_64 lab5.o -o main1  
[exeample@oirossokhin nasm]$ ./lab5  
ROSSOKHIN OLEG  
[exeample@oirossokhin nasm]$
```

Рис. 4.2: 2

4. Скопируйте файлы `hello.asm` и `lab5.asm` в Ваш локальный репозиторий в каталог `~/work/study/2022-2023/“Архитектура компьютера”/arch-pc/labs/lab05/`. Загрузите файлы на Github.

A terminal window with a dark background. The title bar shows the user 'exeample@oirossokhin' and the current directory '~/work/study/2022-2023/Архитектура ...'. The terminal contains three lines of commands and their outputs: 1. '[exeample@oirossokhin nasm]\$ cp hello.asm ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab05' 2. '[exeample@oirossokhin nasm]\$ cp lab5.asm ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab05' 3. '[exeample@oirossokhin nasm]\$ cd ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab05'

```
[exeample@oirossokhin nasm]$ cp hello.asm ~/work/study/2022-2023/"Архитектура ко
мпьютера"/arch-pc/labs/lab05
[exeample@oirossokhin nasm]$ cp lab5.asm ~/work/study/2022-2023/"Архитектура ком
пьютера"/arch-pc/labs/lab05
[exeample@oirossokhin nasm]$ cd ~/work/study/2022-2023/"Архитектура компьютера"
/arch-pc/labs/lab05
```

Рис. 4.3: 4

## 5 Вопросы для самопроверки

*1. Какие основные отличия ассемблерных программ от программ на языках высокого уровня?*

Основные отличия заключаются в том, что ассемблерная программа обращается напрямую к ядру ОС и содержит только тот код, который ввел программист.

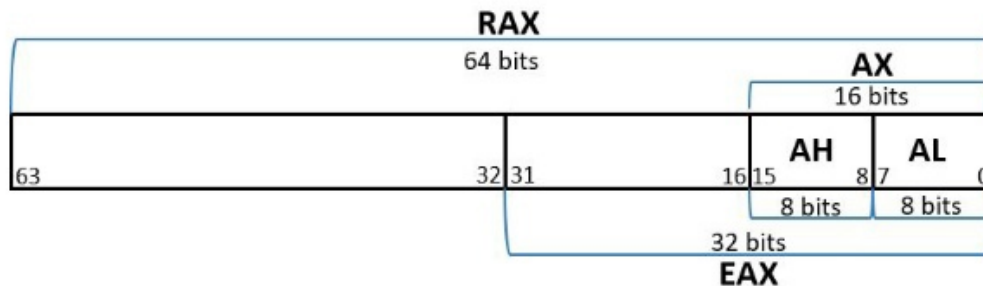
*2. В чём состоит отличие инструкции от директивы на языке ассемблера?*

Отличие состоит в том, что директива -не переводящаяся в машинную команду инструкция, а управляющая работой транслятора. Она используется для определения данных и пишется большими буквами.

*3. Перечислите основные правила оформления программ на языке ассем- блера.*

Чтобы писать программы на ассемблере, нужно знать, какие регистры процессора существуют и как их можно использовать. Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита.

- RAX, RCX, RDX, RBX, RSI, RDI — 64-битные
- EAX, ECX, EDX, EBX, ESI, EDI — 32-битные
- AX, CX, DX, BX, SI, DI — 16-битные
- AH, AL, CH, CL, DH, DL, BH, BL — 8-битные (половинки 16-битных регистров). Например, AH (high AX) — старшие 8 бит регистра AX, AL (low AX) — младшие 8 бит регистра AX.



**Рис. 5.2.** 64-битный регистр процессора 'RAX'

Рис. 5.1: пример

4. *Каковы этапы получения исполняемого файла?*

- Набор текста программы в текстовом редакторе и сохранение её в отдельном файле.
- Трансляция — преобразование с помощью транслятора, например `nasm`, текста программы в машинный код, называемый объектным
- Компоновка — этап обработки объектного кода компоновщиком (`ld`), который принимает на вход объектные файлы и собирает по ним исполняемый файл.
- Запуск программы.

5. *Каково назначение этапа трансляции?*

Трансляция нужна для преобразования текста программы в машинный код(объектный).

6. *Каково назначение этапа компоновки?*

Компоновка осуществляет обработку машинного кода компоновщиком, принимает на вход объектные файлы и собирает по ним исполняемый файл.

*7. Какие файлы могут создаваться при трансляции программы, какие из них создаются по умолчанию?*

На этапе трансляции генерируется листинг программы и объектный файл (по умолчанию)

*8. Каковы форматы файлов для `nasm` и `ld`?*

Для `nasm` - `.asm` Для `ld` - `.o`

## 6 Выводы

По итогам проделанной работы мы научились писать базовую программу вывода текста на языке ассемблера `nasn`, освоили её компиляцию и сборку.



## Список литературы

[https://esystem.rudn.ru/pluginfile.php/1584628/mod\\_resource/content/1/%D0%9B%D0%B0%D0%](https://esystem.rudn.ru/pluginfile.php/1584628/mod_resource/content/1/%D0%9B%D0%B0%D0%)

(author: Демидова А.В.)