Лабораторная работа №5

Создание и процесс обработки программ на языке ассемблера NASM

Россохин Олег

Содержание

# 1 Цель работы

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

# 2 Теоретическое введение

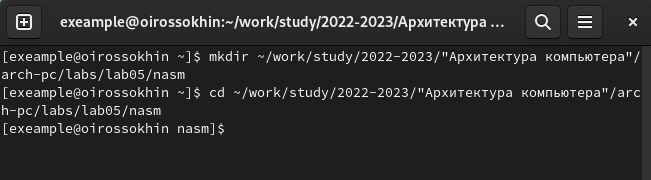
Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства. Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, 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.

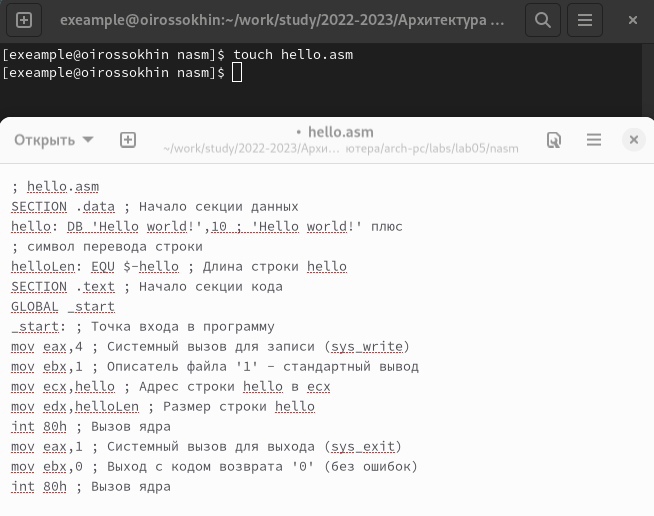
# 3 Выполнение лабораторной работы

## 3.1 Программа Hello world!

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

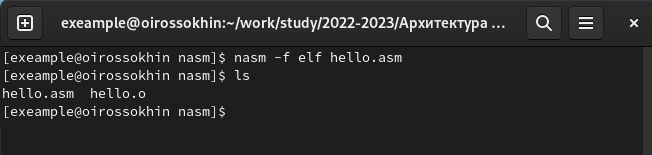
 я создал отдельный каталог nasm в директории lab05.

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



2

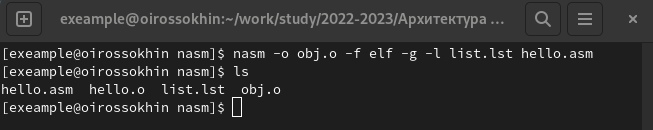
Далее для компиляции файла пропишем в терминале команду а также сразу проверим, создался ли в папке результат компиляции:



3

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

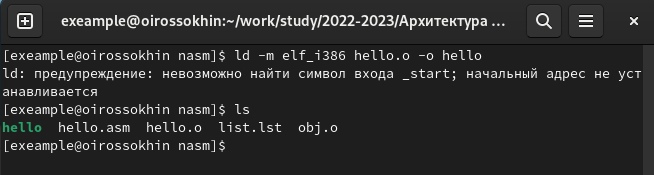
**3.** Далее выполним следующую команду для компиляции файла *hello.asm* в *obj.o* и проверим, что файлы были созданы:



3

Данна команда скомпилировала файл *hello.asm* в *obj.o* при помощи опции -о, 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.

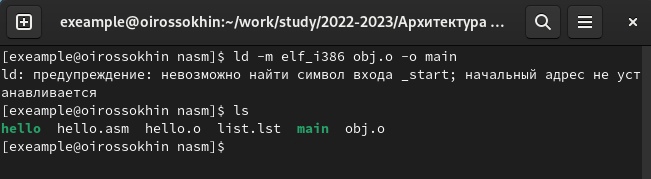
**4.** Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компановщику:



4

Несмотря на ошибку, файл создался.

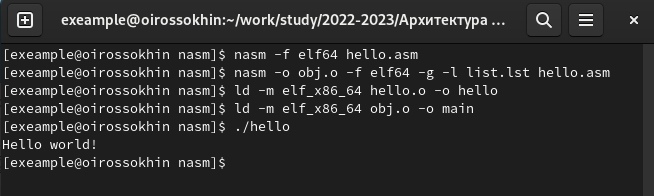
Далее выполним команду:



4.1

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

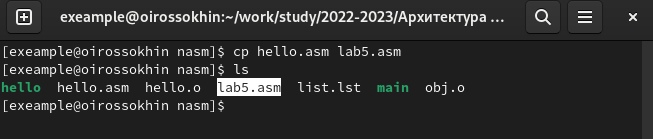
**5.** *Позже обратил внимание на разрядность дистрибутива и проделал те же операции, но для 64-бит системы.* **Запустим созданный исполняемый файл**



5

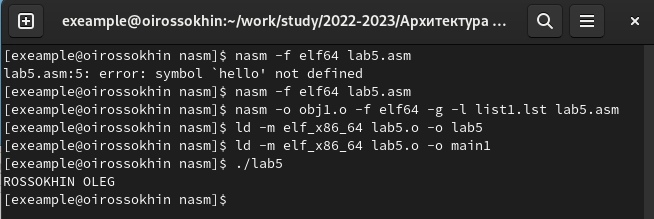
# 4 Задание для самостоятельной работы

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



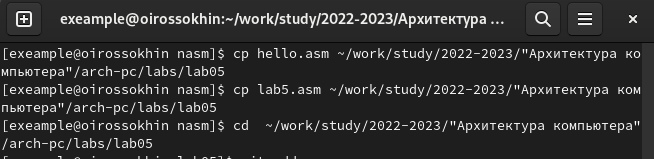
1

**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 получивший- ся исполняемый файл



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.



4

# 5 Вопросы для самопроверки

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

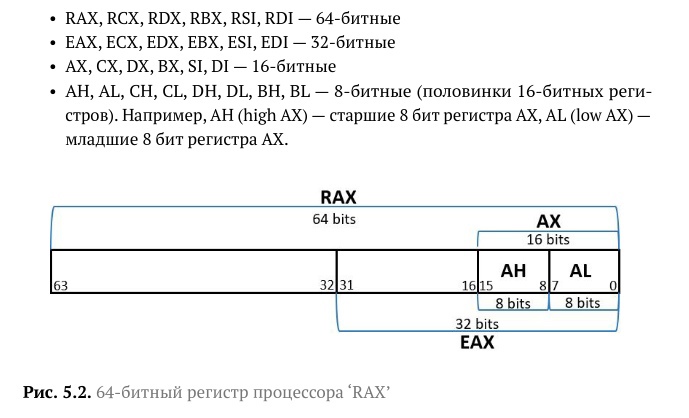
Основные отличия заключаются в том, что ассемблерная программа обращается напрямую к ядру ОС и содержит только тот код, который ввел программист.

1. *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.

1. *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.



пример

1. *Каковы этапы получения исполняемого файла?*

*a)* Набор текста программы в текстовом редакторе и сохранение её в отдельном файле.

*b)* Трансляция — преобразование с помощью транслятора, например nasm, текста программы в машинный код, называемый объектным

*c)* Компоновка — этап обработки объектного кода компоновщиком (ld), который принимает на вход объектные файлы и собирает по ним исполняемый файл.

*d)* Запуск программы.

1. *Каково назначение этапа трансляции?*

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

1. *Каково назначение этапа компоновки?*

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

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

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

1. *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 nasm, освоили её компиляцию и сборку.

# Список литературы

https://esystem.rudn.ru/pluginfile.php/1584628/mod\_resource/content/1/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%965.pdf (author: Демидова А.В.)