**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4**

*дисциплина: Архитектура Вычислительных Систем*

Студент: Сидоренко Максим Алексеевич Группа: НБИбд-02-22

**МОСКВА** 2022 г.

**Цель работы:** Освоение процедуры компиляции и сборки программ, написанных на ассемблере **NASM. Лабораторная работа**

**Ход работы:**

1. **Программа hello world! 3.4.**

* Рассмотрим пример простой программы на языке ассемблера **NASM**. Традиционно первая программа выводит приветственное сообщение Hello world! на экран.
* Создадим каталог для работы с программами на языке ассемблера **NASM:**



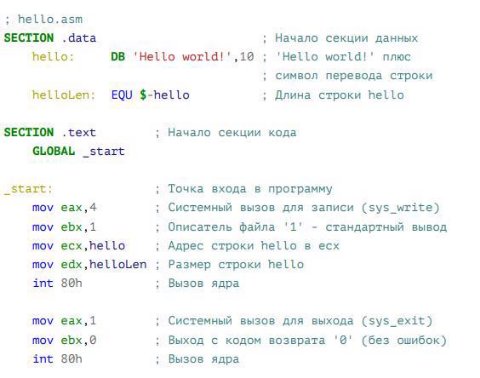
* Перейдем в созданный каталог

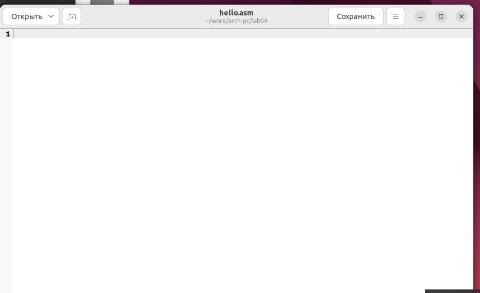


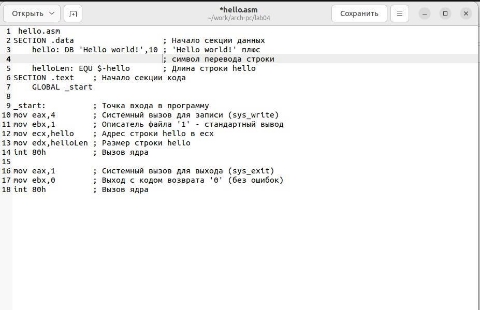
* Создадим текстовый файл с именем hello.asm
* Откроем этот файл с помощью любого текстового редактора, например, gedit



* Введем в него следующий текст:







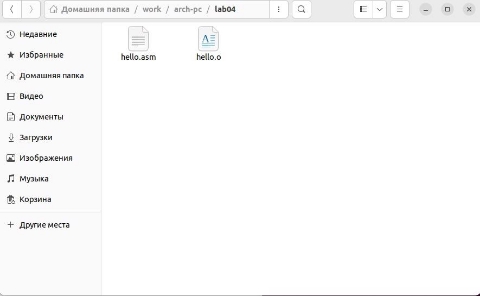
* В отличие от многих современных высокоуровневых языков программирования, в ассемблерной программе каждая команда располагается на отдельной строке. Размещение нескольких команд на одной строке недопустимо. Синтаксис ассемблера **NASM** является чувствительным к регистру, т.е. есть разница между большими и малыми буквами

1. **Транслятор NASM 4.3.2.**

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



* 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 проверем, что объектный файл был создан.\*\*
* (**Вопрос:Какое имя имеет объектный файл?. Ответ: hello.o)**
* **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**



1. **Расширенный синтаксис командной строки NASM 4.3.3.**

* Полный вариант командной строки nasm выглядит следующим образом:



* Выполним следующую команду:

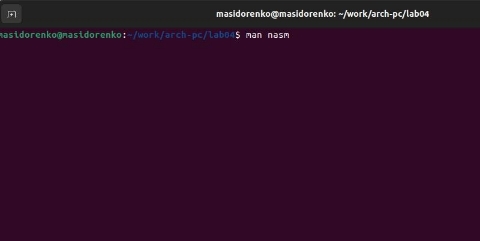


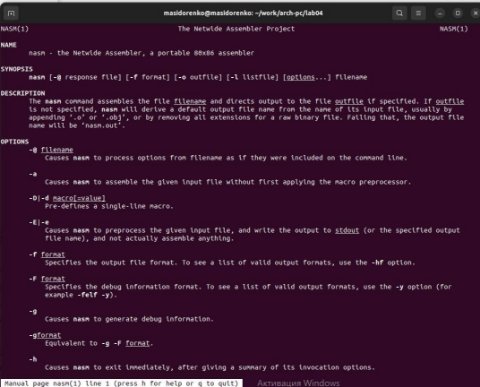
* Данная команда скомпилирует исходный файл hello.asm в obj.o (опция - o позволяет задать имя объектного файла, в данном случае obj.o), при этом формат выходного файла будет elf, и в него будут включены символы для отладки (опция -g), кроме того, будет создан файл листинга list.lst (опция -l).
* С помощью команды ls проверем, что файлы были созданы

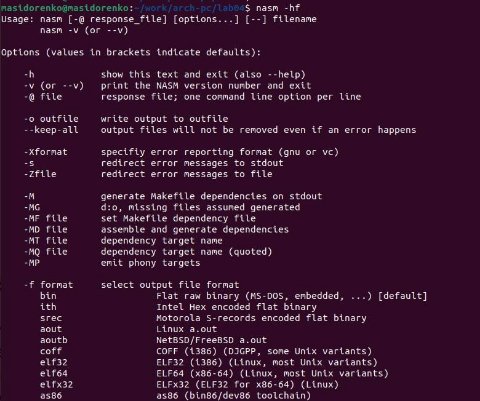


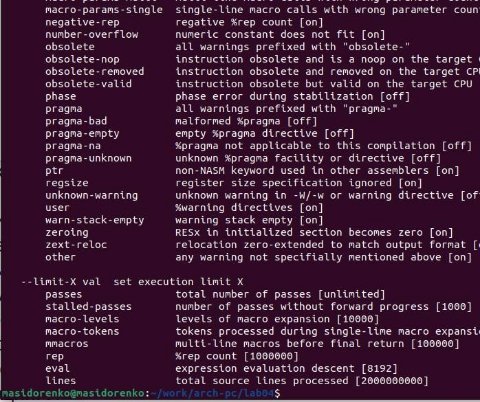
* Для более подробной информации см. man nasm. Для получения

списка форматов объектного файла см. nasm -hf.









1. **Компоновщик LD 4.4.**



* Как видно из схемы на рис, чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику:



* С помощью команды ls проверем, что исполняемый файл hello был создан.



* Компоновщик ld не предполагает по умолчанию расширений для файлов, но принято использовать следующие расширения:

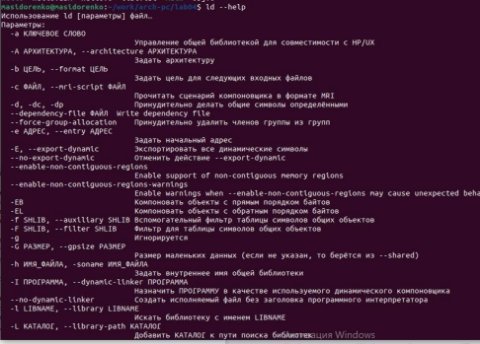


* Ключ -o с последующим значением задаёт в данном случае имя создаваемого исполняемого файла.
* Выполните следующую команду:

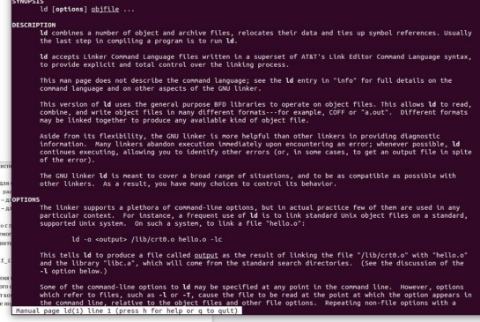




* **(Вопрос: Какое имя будет иметь исполняемый файл? Ответ: main)**
* **(Вопрос: Какое имя имеет объектный файл из которого собран этот исполняемый файл? Ответ: obj.o)**
* Формат командной строки LD можно увидеть, набрав ld –help. Для получения более подробной информации см. man ld.







1. **Запуск исполняемого файла 4.4.1.**

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



1. **Самостоятельная работа 4.5.**

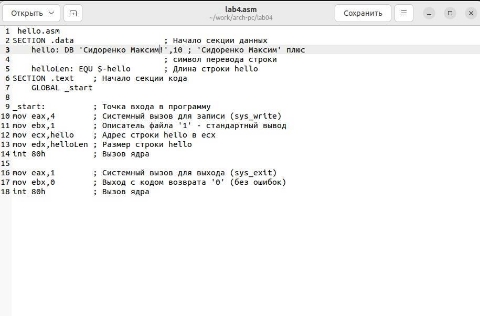
**Ход работы:**

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



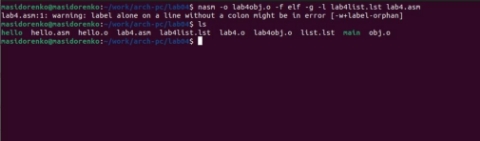
1. **С помощью любого текстового редактора внесём изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с моей фамилией и именем.**





1. **Оттранслируем полученный текст программы lab4.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл.**





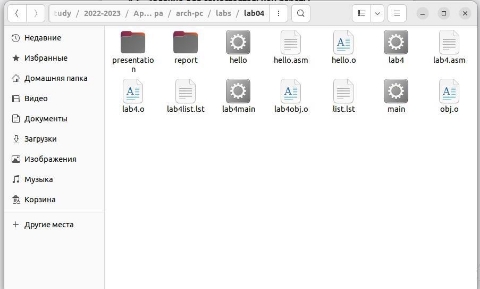


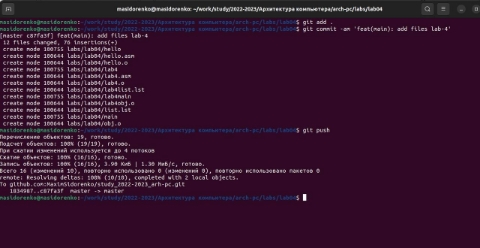


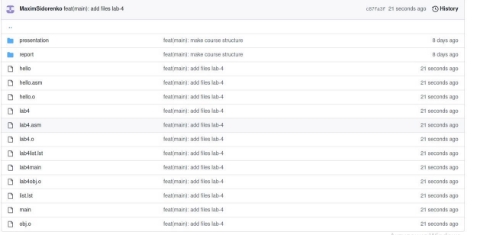


1. **Скопируем файлы hello.asm и lab4.asm в наш локальный репозиторий в каталог ~/work/study/2022- 2023/“Архитектура компьютера”/archpc/labs/lab04/. Загрузим файлы на Github.**









**Вывод: При выполнении работы я освоил процедуры компиляции и сборки программ, написанных на ассемблере NASM**

**(ссылка на github)**

**(https://github.com/MaximSidorenko/study\_2022-2023\_arh-pc)**