**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4**

***дисциплина: Архитектура компьютеров и операционные системы***

Студент: Ян Роман Алексеевич

Группа: НПИбд-02-23

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

**2.1 Программа Hello world!**

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:



Рис 2.1.1: Создание каталога /work/arch-pc/lab04

Перейдём в созданный каталог



Рис 2.1.2: Переход в каталог с помощью команды cd

Создадим текстовый файл с именем **hello.asm**



Рис 2.1.3: Создание текстового файла с помощью команды touch

Откроем этот файл с помощью любого текстового редактора, например, gedit



Рис 2.1.4: Откроем файл с помощью текстового редактора gedit

Введём в нём следующий текст:

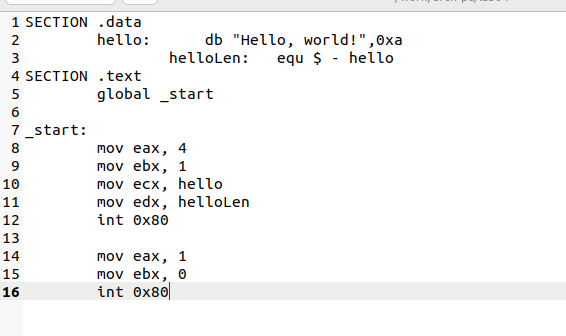


Рис 2.1.5: Демонстрация текста в файле

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.

**2.2 Транслятор NASM**

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



Рис 2.2.1: Компиляция текста с помощью команды nasm -f elf hello.asm

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* проверим, что объектный файл был создан:



Рис 2.2.2: Проверка созданного файла

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.

**2.3 Расширенный синтаксис командной строки NASM**

Полный вариант командной строки nasm выглядит следующим образом:

nasm [-@ косвенный\_файл\_настроек] [-o объектный\_файл] [-f формат\_объектного\_файла] [-l листинг] [параметры…] [–] исходный\_файл

Выполним следующую команду:



Рис 2.3.1: Компиляция исходного файла hello.asm в obj.o

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* проверим, что файлы были созданы:



Рис 2.3.2: Проверка созданных файлов

Для более подробной информации см. *man nasm*. Для получения списка форматов объектного файла см. *nasm -hf*.

**2.4 Компоновщик LD**

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



Рис 2.4.1: Передача объектного файла на обработку компоновщику

С помощью команды *ls* проверим, что исполняемый файл hello был создан:



Рис 2.4.2: Проверка созданного файла hello

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

• o – для объектных файлов;

• без расширения – для исполняемых файлов;

• map – для файлов схемы программы;

• lib – для библиотек.

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

Выполним следующую команду:



Рис 2.4.3: Создание исполняемого файла main и его проверка

Объектный файл obj.o был передан на обработку компоновщику для создания исполняемого файла main.

Формат командной строки LD можно увидеть, набрав *ld –help.* Для получения более подробной информации см. *man ld*.

**2.5 Запуск исполняемого файла**

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



Рис 2.5.1: Запуск исполняемого файла hello с помощью команды ./hello

# 3 Самостоятельна работа

*Задание№1* *В каталоге ~/work/arch-pc/lab04 с помощью команды cp создайте копию файла hello.asm с именем lab4.asm*

Создадим копию файла hello.asm с именем lab4.asmd



Рис 3.1.1: Копирование файла

*Задание№2* *С помощью любого текстового редактора внесите изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с вашими фамилией и именем.*

С помощью редактора markdown внесём изменения в текст в файле lab4.asm



Рис 3.2.1: Применение команды gedit

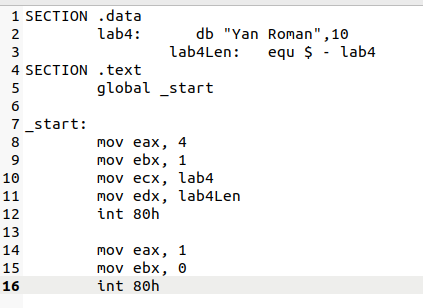


Рис 3.2.2: Демонстрация изменённого текста

*Задание№3* *Оттранслируйте полученный текст программы lab4.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл.*

Скомпилируем файл lab4.asm



Рис 3.3.1: Компиляция файла

Передадим объектный файл lab4.o на обработку компоновщику

Рис 3.3.2: Передача объектного файла на обработку компоновщику

Запустим получившийся исполняемый файл LF



Рис 3.3.3: Запуск исполняемого файла LF с помощью команды ./LF

*Задание№4* *Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/.* *Загрузите файлы на Github.*

Скопируем файлы в локальный репозиторий



Рис 3.4.1: Копирование файлов hello.asm и lab4.asm

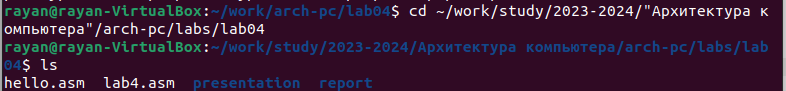


Рис 3.4.2: Проверка

Загрузим файлы на Github

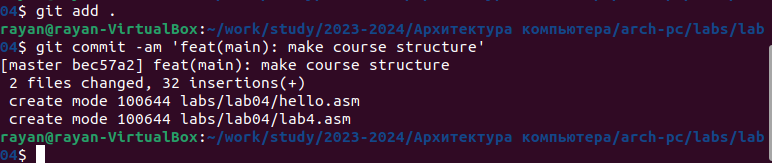


Рис 3.4.3: (1)Загрузка файлов на гитхаб

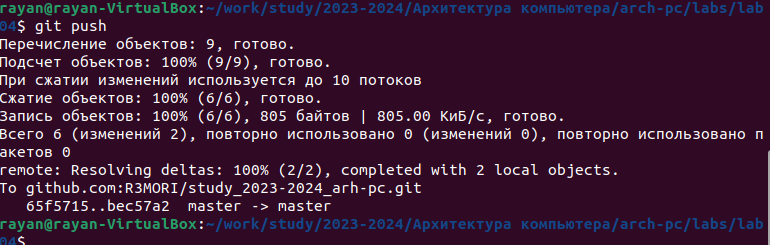


Рис 3.4.4: (2)Загрузка файлов на гитхаб

# 4 Выводы

Я освоил процедуры компиляции и сборки программ, написанных на ассемблере NASM.