Лабораторная работа №4. Создание и процесс обработки программ на языке ассемблера NASM

Простейший вариант

Диана Садова Алексевна

Содержание

# 1 Цель работы

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

# 2 Задание

## 2.1 Порядок выполнения лабораторной работы

### 2.1.1 Программа Hello world!

Рассмотрим пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит приветственное сообщение Hello world! на экран.

Создайте каталог для работы с программами на языке ассемблера NASM (рис.2.1)

Создаем каталог для работы на языке NASM

Создаем каталог для работы на языке NASM

Перейдите в созданный каталог (рис.2.2)

Создаем каталог

Создаем каталог

Создайте текстовый файл с именем hello.asm (рис.2.3)(рис.2.4)

Создаем текстовый файл

Создаем текстовый файл

Проверяем его наличие

Проверяем его наличие

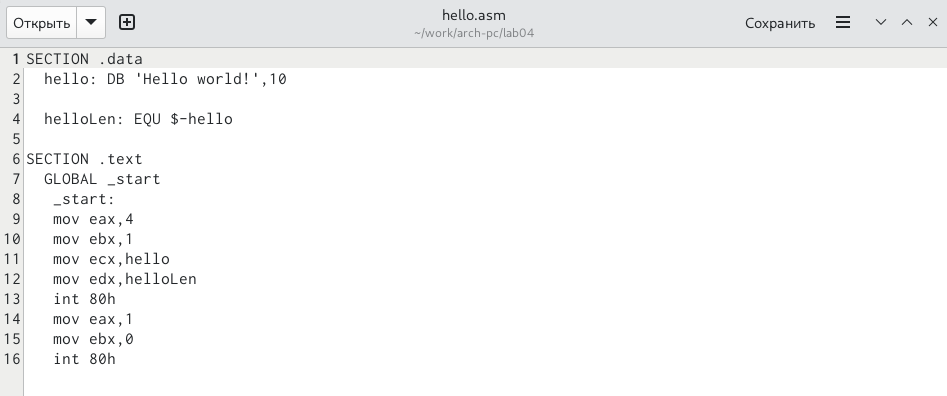
Откройте этот файл с помощью любого текстового редактора, например, gedit (рис.2.5)

Открываем текстовый файл с помощью редактора gedit

Открываем текстовый файл с помощью редактора gedit

С помощью gedit начинаем редактировать текст для дальнейшей работы.

И введите в него следующий текст (рис.2.6)



Вводим текст в hello.asm

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.

### 2.1.2 Транслятор NASM

NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World» необходимо написать (рис.2.7)

Преображаем текст в код программы

Преображаем текст в код программы

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 проверьте, что объектный файл был создан. Какое имя имеет объектный файл?

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.

### 2.1.3 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↪

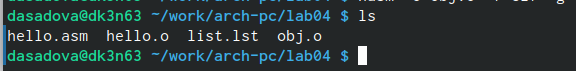
Выполните следующую команду (рис.2.8)

Вводим полный вариант командной строки nasm

Вводим полный вариант командной строки nasm

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 проверьте, что файлы были созданы (рис.2.9)



Проверяем наличие файла

Для более подробной информации см. man nasm. Для получения списка форматов объектного файла см. nasm -hf. # Теоретическое введение

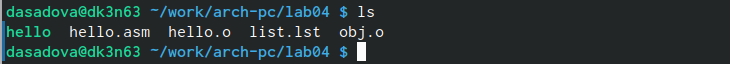
## 2.2 Компоновщик LD

Как видно из схемы на рис. 4.3, чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику (рис.2.10)

Передаем проект на обработку ком

Передаем проект на обработку ком

С помощью команды ls проверьте, что исполняемый файл hello был создан (рис.2.11)



Проверяем корректность выполненной работы

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.

Выполните следующую команду (рис.2.12)

Создаем исполнитель

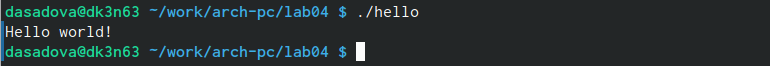
Создаем исполнитель

Какое имя будет иметь исполняемый файл? Какое имя имеет объектный файл из которого собран этот исполняемый файл?

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

### 2.2.1 Запуск исполняемого файла

Запустить на выполнение созданный исполняемый файл, находящийся в текущем каталоге, можно, набрав в командной строке (рис.2.13)



Запускаем созданный исполняемый файл

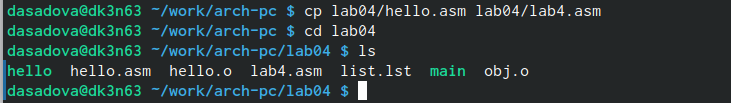
Проверя работу программы, мы получили надпись “Hello world!”записанную в текстовый и далее преобразованный файл.

# 3 Теоретическое введение

# 4 Выполнение лабораторной работы

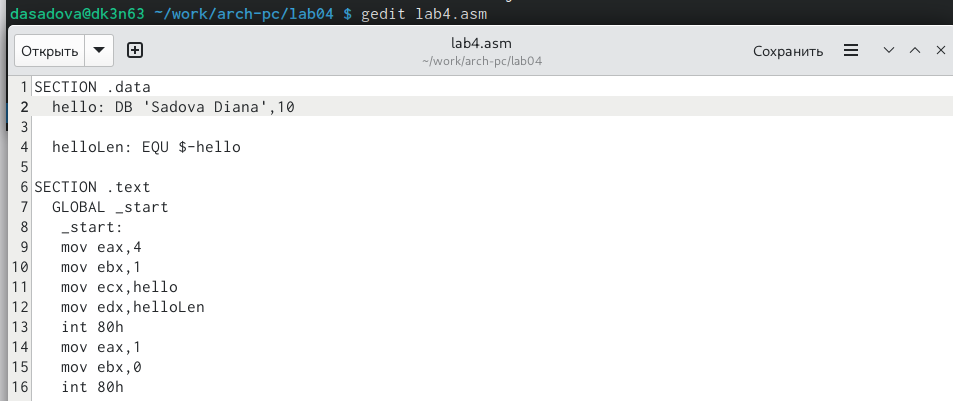
## 4.1 Задание для самостоятельной работы

### 4.1.1 В каталоге ~/work/arch-pc/lab04 с помощью команды cp создайте копию файла hello.asm с именем lab4.asm (рис.4.1)

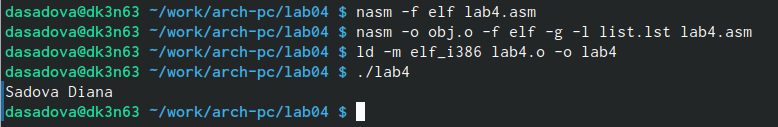


Создаем копию hello.asm с именем lab4.asm

### 4.1.2 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 (рис.4.2)(рис.4.3)

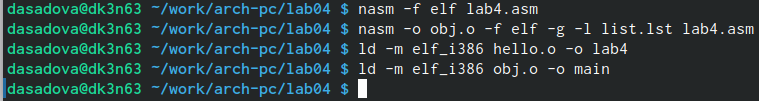


Открываем с помощью текстового редактора и вносим изменения



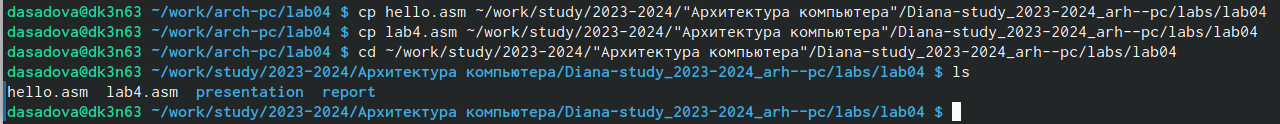
Проверяем, что изменения внесены верно

### 4.1.3 Оттранслируйте полученный текст программы lab4.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл (рис.4.4)

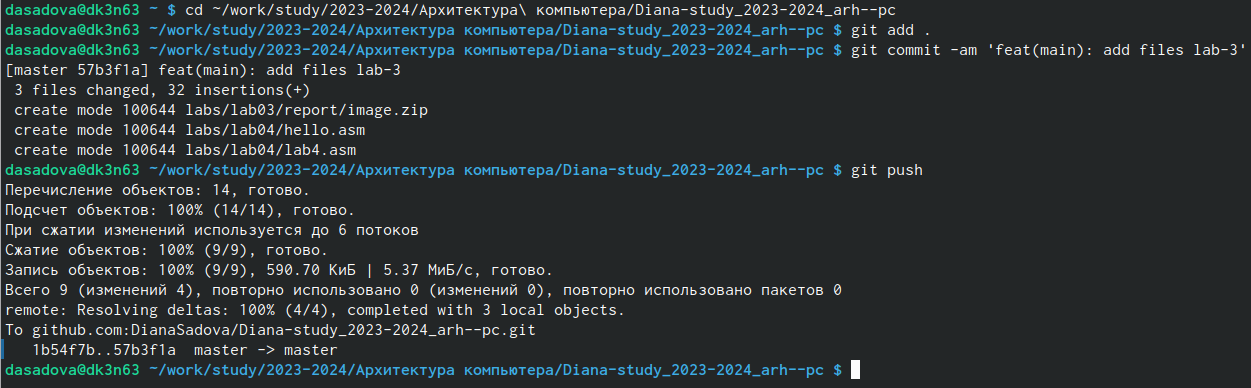


Оттранслируем полученный текст в объектный файл

### 4.1.4 Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/. Загрузите файлы на Github (рис.4.5)(рис.4.6)



Копируем файлы hello.asm и lab4.asm в локальный репозиторий



Загружаем файл на Github

# 5 Выводы

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

# Список литературы