Лабораторная работа №4

Архитектура вычислительных систем

Заболотная Кристина Александровна

Содержание

# 1 Цель работы

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

# 2 Задание

1. В каталоге ~/work/arch-pc/lab04 с помощью команды cp создадим копию файла hello.asm с именем lab4.asm.
2. С помощью любого текстового редактора внесим изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с моим фамилией и именем.
3. Оттранслируем полученный текст программы lab5.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл.
4. Скопируем файлы hello.asm и lab5.asm в свой локальный репозиторий в каталог ~/work/study/2022-2023/“Архитектура компьютера”/archpc/labs/lab05/. Загрузим файлы на Github.

# 3 Теоретическое введение

Язык ассемблера (assembly language, сокращённо asm) — машинноориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям,что позволяет получить к ним более полный доступ, нежели в языках высокого уровня, таких как C/C++, Perl, Python. Мнемокод — непосредственно мнемоника инструкции процессору, которая является обязательной частью команды. Операндами могут быть числа, данные, адреса регистров или адреса оперативной памяти. Метка — это идентификатор, с которым ассемблер ассоциирует некоторое число, чаще всего адрес в памяти. Архитектура ЭВМ. В процессе создания ассемблерной программы можно выделить четыре шага: 1. Набор текста программы в текстовом редакторе и сохранение её в отдельном файле. Каждый файл имеет свой тип (или расширение), который определяет назначение файла. Файлы с исходным текстом программ на языке ассемблера имеют тип asm. 2. Трансляция — преобразование с помощью транслятора, например nasm, текста программы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную дополнительную информацию, созданную транслятором. Тип объектного файла — o, файла листинга — lst. 3. Компоновка или линковка — этап обработки объектного кода компоновщиком (ld), который принимает на вход объектные файлы и собирает по ним исполняемый файл. Исполняемый файл обычно не имеет расшире- ния. Кроме того, можно получить файл карты загрузки программы в ОЗУ, имеющий расширение map. 4. Запуск программы. Конечной целью является работоспособный исполняемый файл. Ошибки на предыдущих этапах могут привести к некорректной работе программы, поэтому может присутствовать этап отладки программы при помощи специальной программы — отладчика. При нахождении ошибки необходимо провести коррекцию программы, начиная с первого шага.

# 4 Выполнение лабораторной работы

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 1)

1. Создадим каталог для работы с программами на языке ассемблера NASM, перейдем в созданный каталог. Ссылка на иллюстрацию (рис. 1)

Рис. 1: 41.png

Рис. 1: 41.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 2)

1. Создадим текстовый файл с именем hello.asm. Ссылка на иллюстрацию (рис. 2)

Рис. 2: 42.png

Рис. 2: 42.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 3)

1. Откроем этот файл с помощью текстового редактора gedit. Ссылка на иллюстрацию (рис. 3)

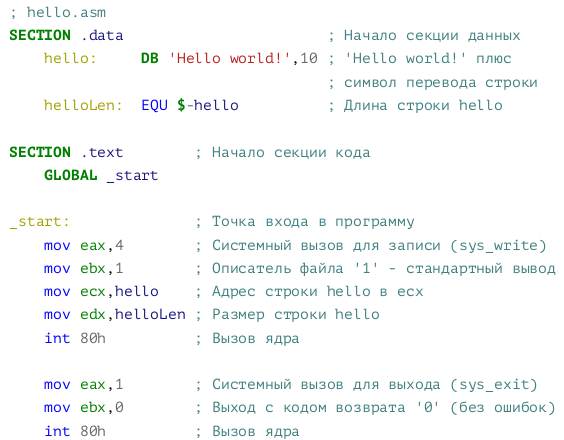


Рис. 3: 44.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 4)

1. Введём в него следующий текст. Ссылка на иллюстрацию (рис. 4)

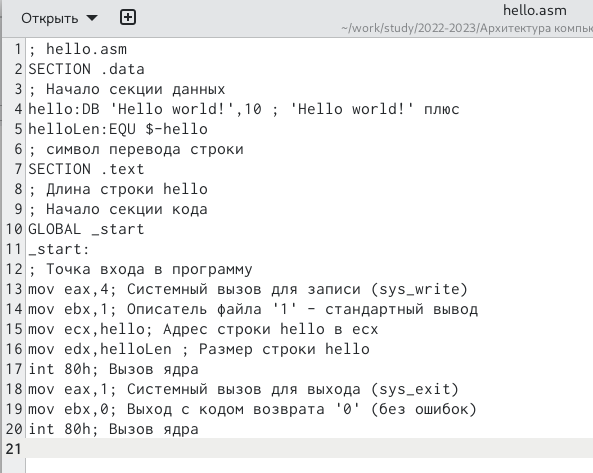


Рис. 4: 45.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 5)

1. Выполним команду, которая скомпилирует исходный файл hello.asm в obj.o, при этом формат выходного файла будет elf, и в него будут включены символы для отладки, кроме того, будет создан файл листинга list.lst. С помощью команды ls проверим, что файлы были созданы. Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику. С помощью команды ls проверим, что исполняемый файл hello был создан. Ссылка на иллюстрацию (рис. 5)

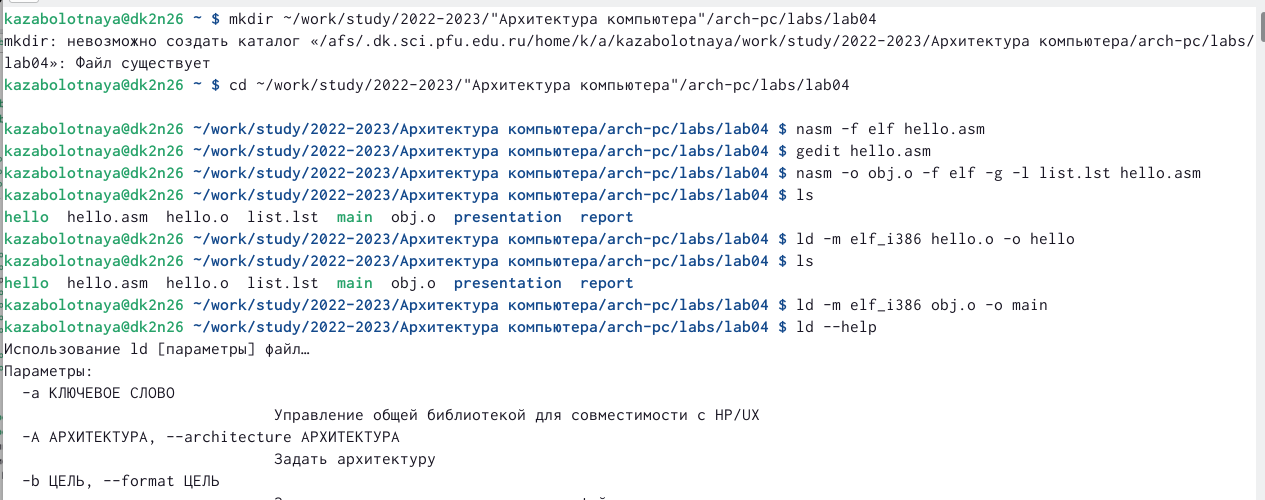


Рис. 5: 43.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 6)

1. Запустим исполняемый файл набрав в командной строке: ./hello. Ссылка на иллюстрацию (рис. 6)

Рис. 6: 46.png

Рис. 6: 46.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 7)

1. В каталоге ~/work/arch-pc/lab04 с помощью команды cp создадим копию файла hello.asm с именем lab4.asm. Ссылка на иллюстрацию (рис. 7)

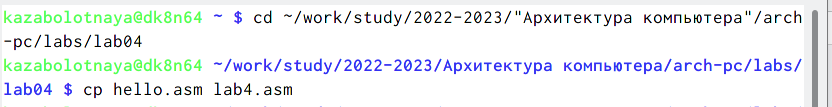


Рис. 7: 47.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 8)

1. С помощью любого текстового редактоcdра внесем изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с фамилией и именем. Оттранслируем полученный текст программы lab4.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл. Ссылка на иллюстрацию (рис. 8)

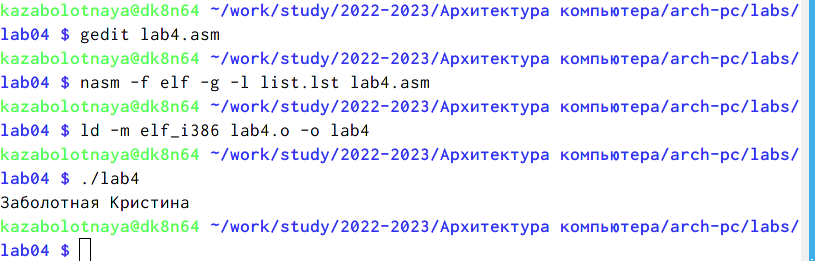


Рис. 8: 48.png

# 5 Выводы

В ходе выполнения данной лабораторной работы была освоена процедура компиляции и сборки программ, написанных на ассемблере NASM.

# Список литературы