Отчёт по лабораторной работе №4

дисциплина: Архитектура компьютера

Баранова Анна Андреевна

Содержание

# 1 Цель работы

Цель работы - освоить процедуры компиляции и сборки программ, которые написаны на ассемблере NASM.

# 2 Задание

В ходе выполнения данной лабораторной работы необходимо изучить:

* Что такое ассемблер и язык ассемблера;
* как создавать и обрабатывать программы на языке ассемблера;
* как оформлять изображения в Markdown.

Выполнив эту работу, мы приобретём навыки работы процедуру оформления отётов с помощью легковесного языка разметки Markdown.

# 3 Теоретическое введение

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный 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 и пр. Заметим, что получить полный доступ к ресурсам компьютера в современных архитектурах нельзя, самым низким уровнем работы прикладной программы является обращение напрямую к ядру операционной системы. Именно на этом уровне и работают программы, написанные на ассемблере. Но в отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввёл программист. Таким образом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора.

Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — машинные коды. До появления языков ассемблера программистам приходилось писать программы, используя только лишь машинные коды, которые были крайне сложны для запоминания, так как представляли собой числа, записанные в двоичной или шестнадцатеричной системе счисления. Преобразование или трансляция команд с языка ассемблера в исполняемый машинный код осуществляется специальной программой транслятором — Ассемблер.

В процессе создания ассемблерной программы можно выделить четыре шага:

* 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.
* Трансляция — преобразование с помощью транслятора, например nasm, текста про- граммы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную допол- нительную информацию, созданную транслятором. Тип объектного файла — o, файла листинга — lst.
* 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 расширения. Кроме того, можно получить файл карты загрузки программы в ОЗУ, имеющий расширение map.
* 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

## 4.1 Программа Hello world!

Создадим каталог для работы с программами на языке ассемблера NASM (рис. 1).

Рис. 1: Создание каталога для работы с программами на языке ассемблера NASM

Рис. 1: Создание каталога для работы с программами на языке ассемблера NASM

Перейдём в созданный каталог (рис. 2).

Рис. 2: Переход в созданный каталог

Рис. 2: Переход в созданный каталог

Создадим текстовый файл с именем hello.asm (рис. 3).

Рис. 3: Создание текстового файлв с именем hello.asm

Рис. 3: Создание текстового файлв с именем hello.asm

Откроем этот файл с помощью текстового редактора gedit (рис. 4).

Рис. 4: Открытие файла с помощью текстового редактора gedit

Рис. 4: Открытие файла с помощью текстового редактора gedit

Введём в него следующий текст (рис. 5).

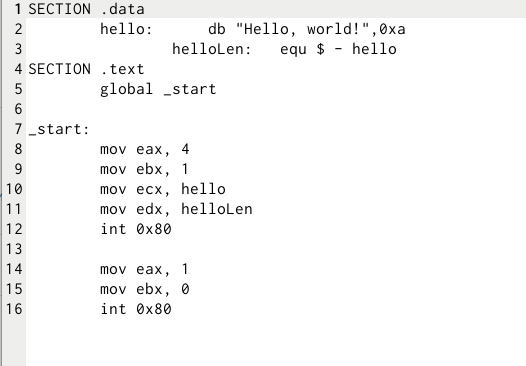


Рис. 5: Ввод текста

## 4.2 Транслятор NASM

Скомпилируем приведённый выше текст программы «Hello World» (рис. 6).

Рис. 6: Компиляция текста

Рис. 6: Компиляция текста

С помощью команды ls проверим, что объектный файл был создан (рис. 7).

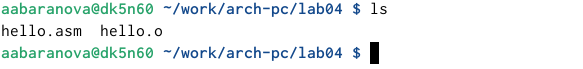


Рис. 7: Проверка создания объектного файла

## 4.3 Расширенный синтаксис командной строки NASM

Скомпилируем 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 (рис. 8).

Рис. 8: Компиляция файла hello.asm в obj.o

Рис. 8: Компиляция файла hello.asm в obj.o

С помощью команды ls проверим, что файлы были созданы (рис. 9).



Рис. 9: Проверка создания файлов

Для более подробной информации используем man nasm (рис. 10), (рис. 11).

Рис. 10: Получение более подробной информации

Рис. 10: Получение более подробной информации

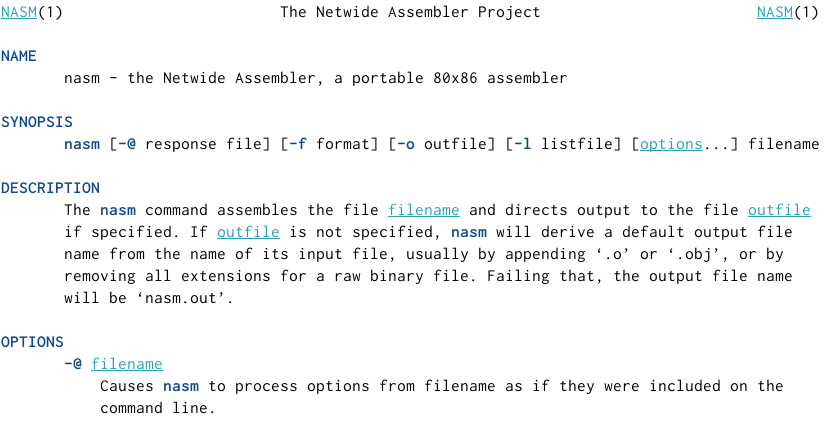


Рис. 11: Получение более подробной информации

Для получения списка форматов объектного файла используем nasm -hf (рис. 12).

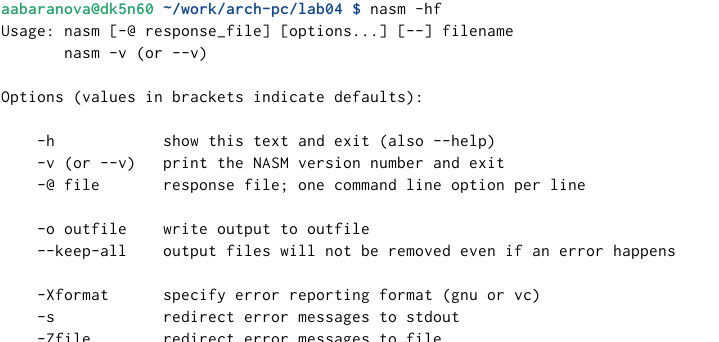


Рис. 12: Получение списка форматов объектного файла

## 4.4 Компоновщик LD

Передадим объектный файл на обработку компоновщику (рис. 13).

Рис. 13: Передача объектного файла на обработку компоновщику

Рис. 13: Передача объектного файла на обработку компоновщику

С помощью команды ls проверим, что исполняемый файл hello был создан (рис. 14).

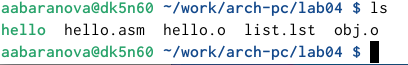


Рис. 14: Проверка создания файла

Зададим имя создаваемого исполняемого файла (рис. 15).

Рис. 15: Дача имени файлу

Рис. 15: Дача имени файлу

Для получения более подробной информации используем man ld (рис. 16), (рис. 17).

Рис. 16: Получение более подробной информации

Рис. 16: Получение более подробной информации

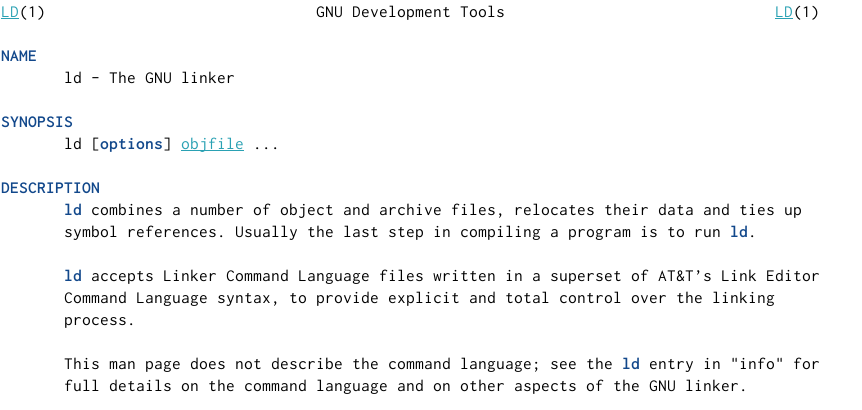


Рис. 17: Получение более подробной информации

## 4.5 Запуск исполняемого файла

Запустим на выполнение созданный исполняемый файл, находящийся в текущем каталоге (рис. 18).



Рис. 18: Запуск исполняемого файла

## 4.6 Задание для самостоятельной работы

В каталоге ~/work/arch-pc/lab04 с помощью команды cp создадим копию файла hello.asm с именем lab4.asm (рис. 19).

Рис. 19: Создание копии файла

Рис. 19: Создание копии файла

С помощью команды ls проверим, что файл lab4.asm был создан (рис. 20).

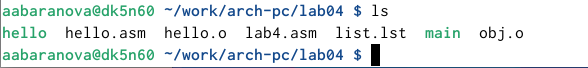


Рис. 20: Проверка создания файла

С помощью текстового редактора gedit внесём изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с моими фамилией и именем (рис. 21), (рис. 22).

Рис. 21: Открытие файла с помощью текстового редактора gedit

Рис. 21: Открытие файла с помощью текстового редактора gedit

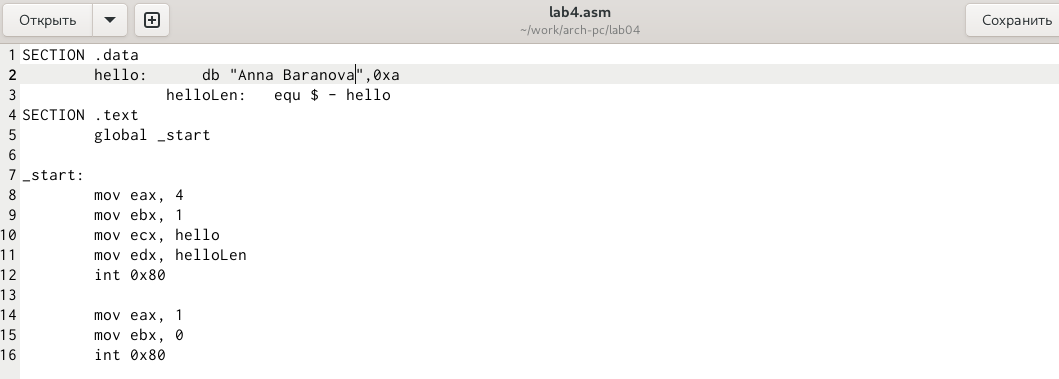


Рис. 22: Изменеие текста программы в файле lab4.asm

Скомпилируем текст файла lab4.asm (рис. 23).

Рис. 23: Компиляция текста

Рис. 23: Компиляция текста

Скомпилируем исходный файл lab4.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 list1.lst (рис. 24).

Рис. 24: Компиляция файла lab4.asm в obj.o

Рис. 24: Компиляция файла lab4.asm в obj.o

Передадим объектный файл на обработку компоновщику (рис. 25).

Рис. 25: Передача объектного файла на обработку компоновщику

Рис. 25: Передача объектного файла на обработку компоновщику

Зададим имя создаваемого исполняемого файла (рис. 26).

Рис. 26: Дача имени файлу

Рис. 26: Дача имени файлу

С помощью команды ls проверим, что нужные файлы были создан (рис. 27).

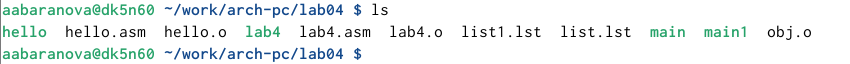


Рис. 27: Проверка создания файлов

Запустим получившийся исполняемый файл (рис. 28).



Рис. 28: Запуск получившегося исполняемого файла

Скопируем файлы hello.asm и lab4.asm в мой локальный репозиторий в каталог ~/work/study/2024-2025/“Архитектура компьютера”/arch-pc/labs/lab04/ (рис. 29), (рис. 30).

Рис. 29: Копирование файла hello.asm

Рис. 29: Копирование файла hello.asm

Рис. 30: Копирование файла lab4.asm

Рис. 30: Копирование файла lab4.asm

Загрузим файлы на Github (рис. 31).

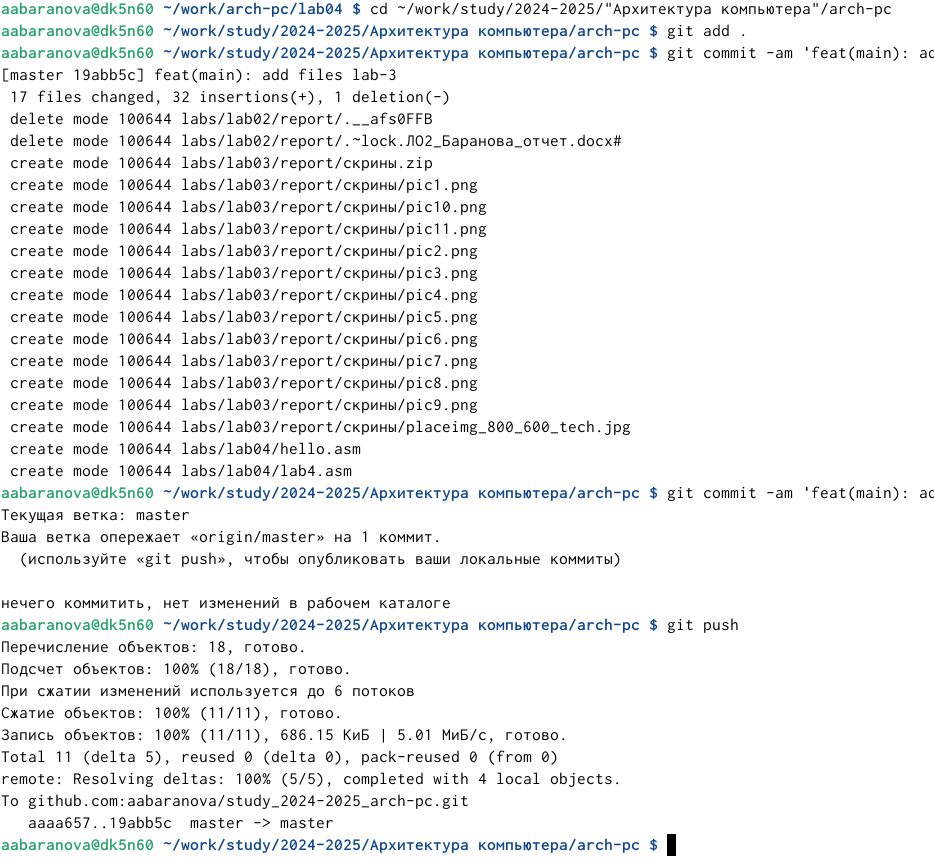


Рис. 31: Загрузка файлов на Github

# 5 Выводы

В ходе выполнения данной лабораторной работы были освоены процедуры компиляции и сборки программ, которые написаны на ассемблере NASM