

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4.

дисциплина: Архитектура компьютера

Студент: Доронин Никита Максимович

Группа: НКАбд-02-23

Цель работы

Целью лабораторной работы №4 является освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

Ход выполнения работы

Программа Hello World!

Создал каталог для работы с программами на языке ассемблера NASM.

Перешёл в созданный каталог.

Произвёл компиляцию текста программы с помощью транслятора NASM: `nasm -f elf hello.asm`.

Проверил, что объектный файл был создан: `ls`.

□ □ □ Произвёл компоновку объектного файла с помощью команды `ld -m elf_i386 hello.o -o hello`.

Проверил, что исполняемый файл был создан: `ls`.

Запустил созданный исполняемый файл: `./hello`.

□

Задание для самостоятельной работы

В каталоге с помощью команды `cp` создал копию файла `hello.asm` с именем `name.asm`.

С помощью текстового редактора внёс изменения в текст программы в файле `name.asm` так, чтобы вместо "Hello world!" на экран выводилась строка с моими фамилией и именем.

Оттранслировал полученный текст программы `name.asm` в объектный файл. Выполнил компоновку объектного файла и запустил получившийся исполняемый файл.

□

Скопировал файлы `hello.asm` и `name.asm` в локальный репозиторий. Загрузил файлы на Github.

□

Вывод

В результате выполнения лабораторной работы №4 я овладел процедурой компиляции и сборки программ, написанных на ассемблере NASM. В процессе выполнения лабораторной работы я написал программу "Hello World!", а также выполнил задание для самостоятельной работы, где внёс изменения в исходный код программы, чтобы она выводила мои фамилию и имя. С помощью транслятора NASM я оттранслировал программу в объектный файл и произвёл компоновку, что позволило мне запустить созданный исполняемый файл. Осуществив копирование файлов в локальный репозиторий и их загрузку на Github, я завершил этап работы.

Контрольные вопросы

Основные отличия ассемблерных программ от программ на языках высокого уровня:

Язык: Ассемблерные программы пишутся на ассемблере, что является ближе к машинному коду, в то время как высокоуровневые языки программирования (например, Python, Java) предлагают абстракции более высокого уровня. Читаемость: Ассемблерные программы, как

правило, более трудны для понимания, поскольку напрямую отображают машинный код процессора, а высокоуровневые языки предоставляют абстракции, которые более легко читаются человеком. Портатбельность: Программы на высокоуровневых языках, как правило, более портируемы, так как абстракции языка позволяют обеспечить более высокий уровень независимости от аппаратного обеспечения.

Отличие инструкции от директивы на языке ассемблера:

Инструкции (мнемоники): Определяют операции, которые должен выполнить процессор. Директивы: Не являются инструкциями, а указывают ассемблеру, как обрабатывать текст программы (например, где разместить данные в памяти, какие метки использовать и т. д.)

Основные правила оформления программ на языке ассемблера:

Все метки должны быть уникальными. Программа должна содержать раздел `.data` для переменных и `.text` для кода. Каждая команда должна начинаться с новой строки. Ассемблерные инструкции обычно чувствительны к регистру.

Этапы получения исполняемого файла:

Трансляция: Программа на языке ассемблера транслируется в машинный код.

Компоновка: Исполняемый файл создается путем комбинирования одного или нескольких объектных файлов. Назначение этапа трансляции:

Преобразует ассемблерный код в машинный код, который может выполняться на процессоре.

Назначение этапа компоновки: Комбинирует объектные файлы и статические библиотеки в единый исполняемый файл, который может быть запущен на выполнение.

Какие файлы могут создаваться при трансляции программы, какие из них создаются по умолчанию:

Объектный файл: Создается всегда и содержит машинный код и данные, но не содержит информации о том, где в памяти должны располагаться эти данные.

Выполняемый файл: Содержит как машинный код и данные, так и информацию о том, где эти данные должны быть размещены в памяти. Создается только в процессе компоновки.

Форматы файлов для `nasm` и `ld`:

Форматы объектных файлов для `nasm`: `elf32`, `elf64`, `win32`, `win64`, `aout`, `aoutb`, `coff`, `ieee`, `macho32`, `macho64`, `obj`, `rdf`, `srec`, `stabs`, `sym`, `bin`. Форматы исполняемых файлов для `ld`: `elf32`, `elf64`, `elfi386`, `elfx86_64`, `elfi1om`, `elfk1om`, `aout`, `aoutb`, `coff`, `ieee`, `macho32`, `macho64`, `aouti386`, `i386linux`, `win32`, `win64`, `xcoff`, `mmo`, `ieee132`, `ieee1473`, `srec`, `tekhex`, `binary`.