Отчёт по лабораторной работе №5. Создание и процесс обработки программ на языке ассемблера NASM

дисциплина: Архитектура компьютера

Ибатулина Дарья Эдуардовна

Содержание

# 1 Цель работы

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

# 2 Задание

1. Ознакомиться с теоретическим введением к работе;
2. Создать каталог для программ на языке ассемблера;
3. Написать простейшую программу, выводящую на экран “Hello, world!”!;
4. Выполнить компоновку объектного файла, запустить исполняемый файл;
5. Загрузить полученные файлы на Github;
6. Сделать вывод по лабораторной работе;
7. Оформить отчёт в Markdown.

# 3 Теоретическое введение

Основными функциональными элементами любой ЭВМ являются память, процессор и периферийные устройства. Через общую шину они взаимодействуют друг с другом. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих дорожек на материнской (системной) плате.

Задачи процессора - обработка информации и координация работы всех узлов компьютера. В состав центрального процессора входят:

* арифметико-логическое устройство (АЛУ);
* устройство управления (УУ);
* регистры.

Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. Большинство команд в программах написанных на ассемблере используют регистры в качестве операндов. Практически все команды представляют собой преобразование данных, хранящихся в регистрах процессора, это например пересылка данных между регистрами или между регистрами и памятью, преобразование (арифметические или логические операции) данных хранящихся в регистрах.

Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита. К примеру, RAX - 64-битные, EAX - 32-битные.

Таким образом можно отметить, что вы можете написать в своей программе, например, такие команды (mov – команда пересылки данных на языке ассемблера):

mov ax, 1 mov eax, 1

Обе команды поместят в регистр AX число 1. Разница будет заключаться только в том, что вторая команда обнулит старшие разряды регистра EAX, то есть после выполнения второй команды в регистре EAX будет число 1. А первая команда оставит в старших разрядах регистра EAX старые данные. И если там были данные, отличные от нуля, то после выполнения первой команды в регистре EAX будет какое-то число, но не 1. А вот в регистре AX будет число 1.

Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). ОЗУ — это быстродействующее энергозависимое запоминающее устройство, которое напрямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно работает в текущий момент. ОЗУ состоит из одинаковых пронумерованных ячеек памяти. Номер ячейки памяти — это адрес хранящихся в ней данных.

В состав ЭВМ также входят периферийные устройства, которые можно разделить на: - устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных (жёсткие диски, твердотельные накопители, магнитные ленты); - устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой.

В основе вычислительного процесса лежит **принцип программного управления**. Это значит, что компьютер выполняет алгоритм (последовательность команд). Коды команд состоят из нулей и единиц. При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется **командным циклом процессора**. В самом общем виде он заключается в следующем:

1. формирование адреса в памяти очередной команды;
2. считывание кода команды из памяти и её дешифрация;
3. выполнение команды;
4. переход к следующей команде.

Данный алгоритм позволяет выполнить хранящуюся в ОЗУ программу.

Язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора. В современных компьютерах пользователь не получает прямого доступа к процессору, обращение происходит к ядру.

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.

NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64.

Типичный формат записи команд NASM имеет вид:

[метка:] мнемокод [операнд {, операнд}] [; комментарий]

Здесь ***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. Т.о. метка перед командой связана с адресом данной команды.

Допустимыми символами в метках являются буквы, цифры, а также следующие символы: *, $, #, @,~,. и ?. Начинаться метка или идентификатор могут с буквы, .,*  и ?. Перед идентификаторами, которые пишутся как *зарезервированные слова*, нужно писать $, чтобы компилятор трактовал его верно (так называемое экранирование). Максимальная длина идентификатора 4095 символов.

Программа на языке ассемблера также может содержать ***директивы*** — инструкции, не переводящиеся непосредственно в машинные команды, а управляющие работой транслятора. Например, директивы используются для определения данных (констант и переменных) и обычно пишутся большими буквами.

Программа на языке Ассемблера состоит из следующих уровней:

* текст программы;
* трансляция;
* компоновка (линковка);
* запуск программы.

# 4 Выполнение лабораторной работы

Создаём каталог для разработки программ на языке ассемблера Nasm и переходим в него. (рис. 1)

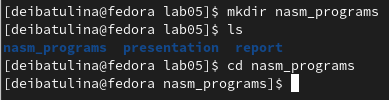


Рис. 1: Создание каталога для программ на Ассемблере

Создаём текстовый файл с помощью команды *touch* и открываем его в текстовом редакторе gedit, вносим изменения в текст программы, сохраняем (рис. 2), (рис. 3).

Рис. 2: Создание файла

Рис. 2: Создание файла

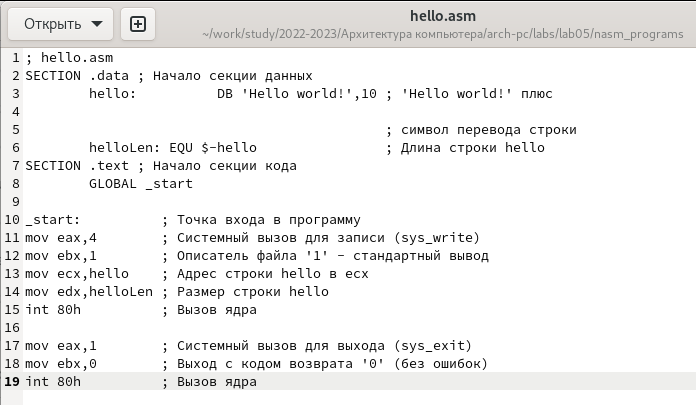


Рис. 3: Открытие в текстовом редакторе

Компилируем данный текст программы, проверяем корректность выполненных действий (рис. 4).

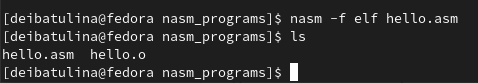


Рис. 4: Компиляция текста программы

Выполняем следующую команду (рис. 5):

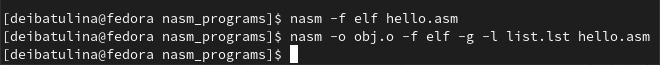


Рис. 5: Создание новых файлов

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).

Проверяем корректность выполненных действий (рис. 6).

Рис. 6: Проверка корректности выполненных действий

Рис. 6: Проверка корректности выполненных действий

Передаём объектный файл на обработку компоновщику, проверяем корректность выполненных действий (рис. 7).

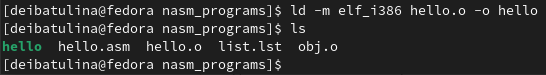


Рис. 7: Передача объектного файла на обработку компоновщику

Задаём имя исполняемого файла (рис. 8). Имя исполняемого файла - main, объектный файл, из которого он собран, имеет имя obj.o.

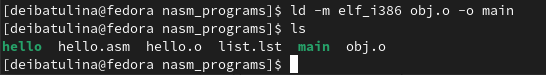


Рис. 8: Задание имени исполняемого файла

Запускаем программу (рис. 9).

Рис. 9: Запуск программы

Рис. 9: Запуск программы

Переходим к выполнению заданий для самостоятельной работы. Первым делом создаём копию файла hello.asm, созданного при выполнении лабораторной работы, в папке для разработки программ (рис. 10).

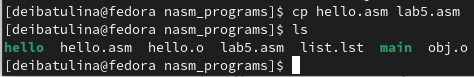


Рис. 10: Создание файла для новой программы

Далее редактируем данный файл с помощью текстового редактора gedit (рис. 11), (рис. 12).

Рис. 11: Редактируем в терминале

Рис. 11: Редактируем в терминале

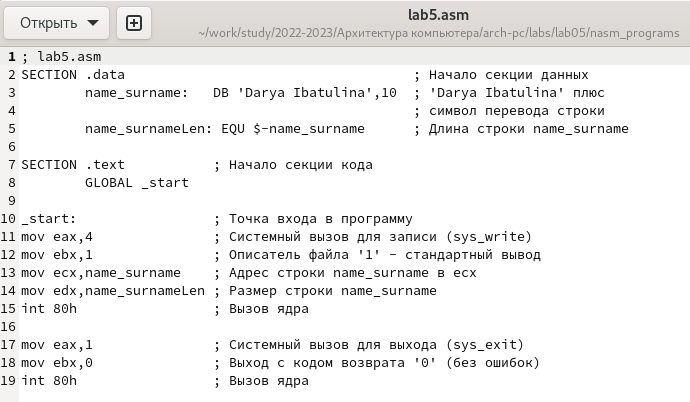


Рис. 12: Редактируем в текстовом редакторе

Компонуем и запускаем исполняемый файл (рис. 13).

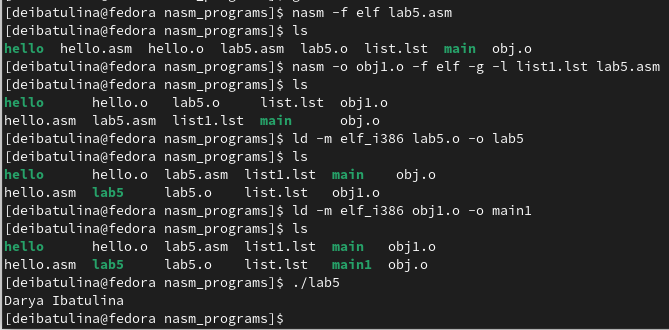


Рис. 13: Компоновка и запуск исполняемого файла

Копируем 2 файла с программами в каталог для лабораторных работ (рис. 14).

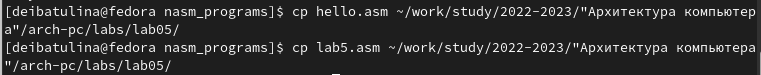


Рис. 14: Копирование файлов с программами в каталог для лабораторных работ

Проверяем корректность выполненных действий (рис. 15).

Рис. 15: Проверка корректности выполненных действий

Рис. 15: Проверка корректности выполненных действий

Отправляем файлов на GitHub посредством локального репозитория (рис. 16).

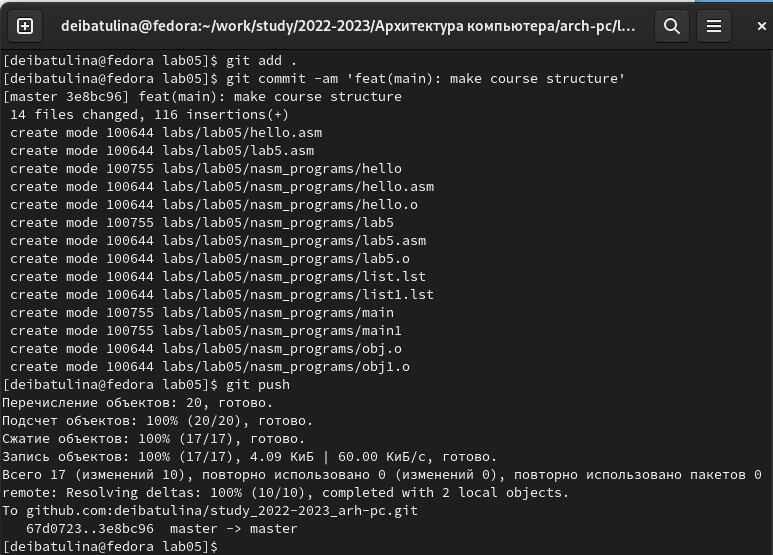


Рис. 16: Отправка файлов на GitHub посредством локального репозитория

# 5 Выводы

В ходе выполнения лабораторной работы я освоила процедуру компиляции и сборки программ, написанных на низкоуровневом языке программирования Ассемблер.

# Список литературы

* Руководство к лабораторной работе №5. Создание и процесс обработки программ на языке ассемблера NASM.