Отчёт по лабораторной работе №4

Дисциплина: архитектура компьютера

Ермакова Анастасия Алексеевна

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

# 2 Задание

1. Программа Hello world!
2. Транслятор NASM
3. Расширенный синтаксис командной строки NASM
4. Компоновщик LD
5. Запуск исполняемого файла
6. Выполнение заданий для самостоятельной работы.

# 3 Теоретическое введение

Основными функциональными элементами любой ЭВМ являются центральный процессор, память и периферийные устройства. Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих дорожек на материнской плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора входят следующие устройства:

* арифметико-логическое устройство (АЛУ) — выполняет логические и арифметические действия, необходимые для обработки информации, хранящейся в памяти;
* устройство управления (УУ) — обеспечивает управление и контроль всех устройств компьютера;
* регистры — сверхбыстрая оперативная память небольшого объёма, входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные 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 каче- стве 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. В качестве примера приведем названия основных регистров общего назначения (именно эти регистры чаще всего используются при написании программ):
* RAX, RCX, RDX, RBX, RSI, RDI — 64-битные
* EAX, ECX, EDX, EBX, ESI, EDI — 32-битные
* AX, CX, DX, BX, SI, DI — 16-битные
* AH, AL, CH, CL, DH, DL, BH, BL — 8-битные

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. Периферийные устройства в составе ЭВМ:

* устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных.
* устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой.

В основе вычислительного процесса ЭВМ лежит принцип программного управления. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы.

Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции. При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. 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.

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. 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.

# 4 Выполнение лабораторной работы

## 4.1 Программа Hello world!

С помощью утилиты cd перехожу в каталог, в котором буду выполнять работу. Затем с помощью утилиты touch создаю текстовый файл с именем hello.asm. Открываю его с помощью текстового редактора mousepad. (рис. 1)

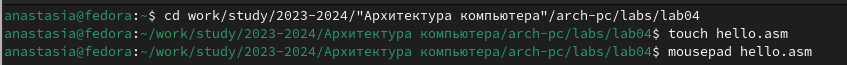


Рис. 1: Создание пустого текстового файла.

Ввожу в текстовый файл программу для вывода “Hello world!”. (рис. 2)

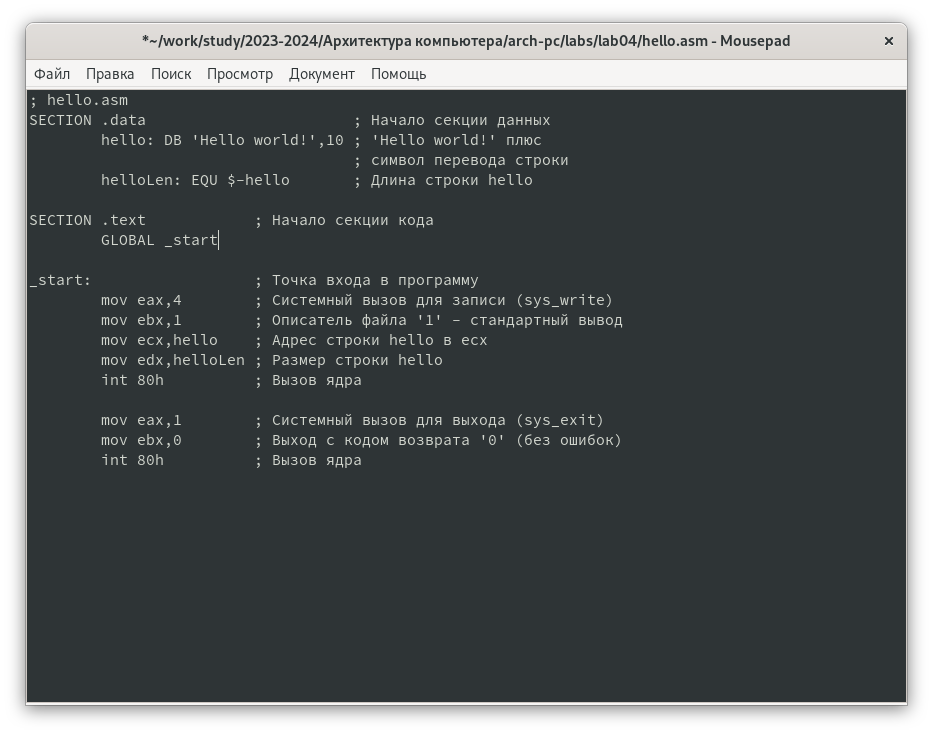


Рис. 2: Заполнение текстового файла.

## 4.2 Транслятор NASM

С помощью транслятора NASM превращаю текст программы в объектный код, для этого устанавливаю пакет nasm и далее прописываю команду nasm -f elf hello.asm. Проверяю, преобразовался ли файл. Должен появиться файл hello.o, проверя. это с помощью команды ls.(рис. 3)

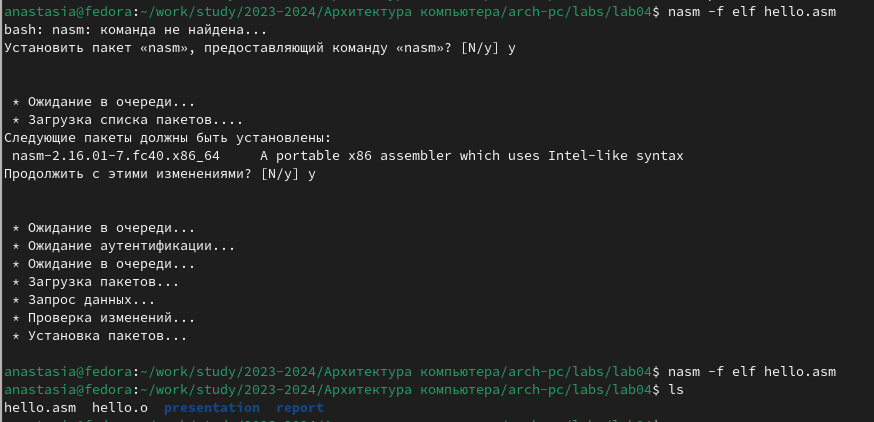


Рис. 3: Преобразование текстового файла в объектный код.

## 4.3 Расширенный синтаксис командной строки NASM

Ввожу команду, которая скомпилирует исходный файл hello.asm в объектный файл obj.o, ключи -g и -l позволят включить в него символы для отладки и создать файл листинга list.lst. С помощью команды ls проверяю, что файлы действительно были созданы. (рис. 4)

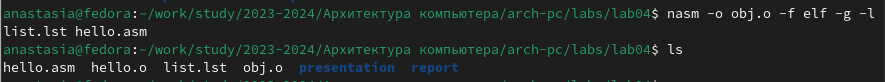


Рис. 4: Компиляция файла.

## 4.4 Компоновщик LD

Передаю файл hello.o на обработку компановщику LD, чтобы получить исполняемый файл hello. Ключ -о позволяет присвоить файлу имя hello. С помощью команды ls убеждаюсь в правильности выполнения команды. (рис. 5)

Рис. 5: Передача файла компановщику LD на обработку.

Рис. 5: Передача файла компановщику LD на обработку.

Выполняю следующую команду (рис. 6). Исполняемый файл будет иметь имя main, указанное после ключа -о. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o.

Рис. 6: Передача файла компановщику LD на обработку.

Рис. 6: Передача файла компановщику LD на обработку.

## 4.5 Запуск исполняемого файла

Запускаю на выполнение исполняемый файл hello, набрав в командной строке ./home. (рис. 7)

Рис. 7: Запуск исполняемого файла.

Рис. 7: Запуск исполняемого файла.

## 4.6 Выполнение заданий для самостоятельной работы.

С помощью команды cp создаю копию файла hello.asm с именем lab4.asm. Открываю файл с помощью текстового редактора mousepad. (рис. 8)

Рис. 8: Создание копии файла.

Рис. 8: Создание копии файла.

Вношу изменения в текст программы в файле lab4.asm, чтобы на экран вывелась строка с моими именем и фамилией. (рис. 9)

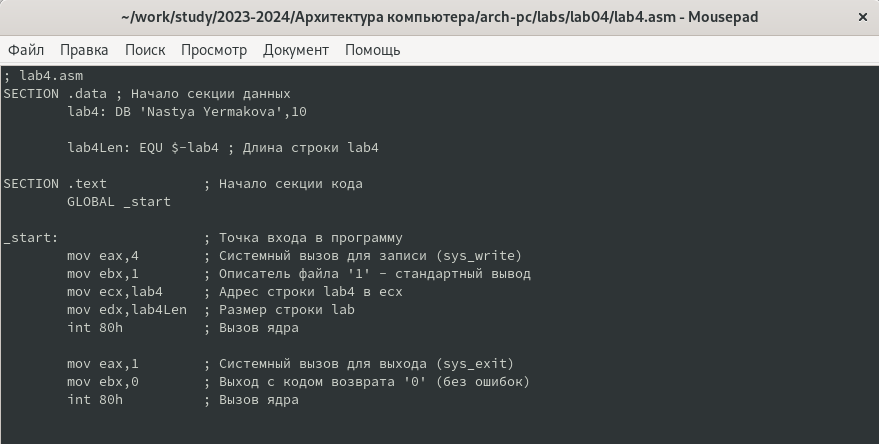


Рис. 9: Корректирование программы.

Компилирую файл программы в объектный файл, с помощью команды ls проверяю, создался ли новый файл lab4.o. (рис. 10)

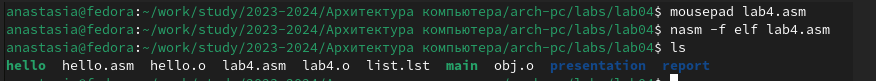


Рис. 10: Компиляция файла.

Передаю объектный файл lab4.o компановщику LD на обработку, получаю файл lab4. Убеждаюсь в этом командой ls. (рис. 11)

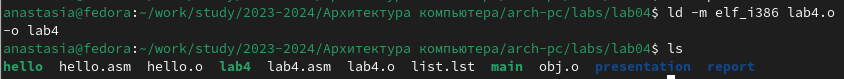


Рис. 11: Передача файла компановщику LD на обработку.

Запускаю получившийся исполняемый файл. (рис. 12)

Рис. 12: Запуск исполняемого файла.

Рис. 12: Запуск исполняемого файла.

Наконец отправляю все файлы на GitHub. (рис. 13)

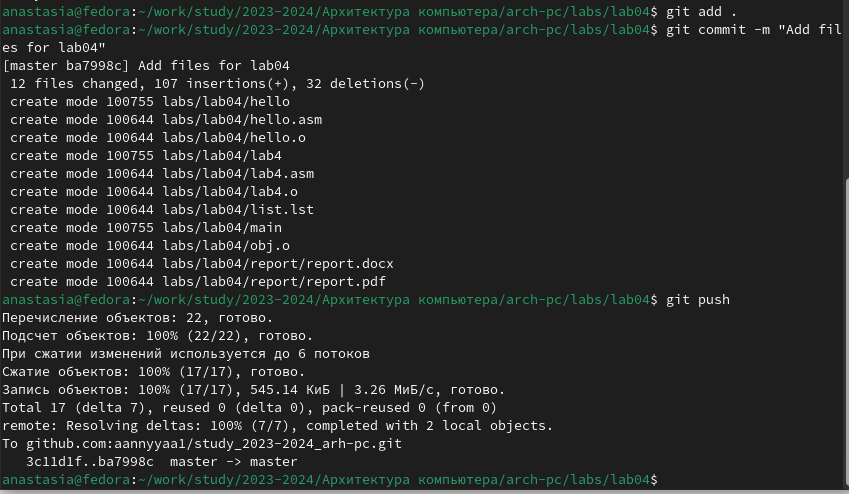


Рис. 13: Отправка файлов на github.

# 5 Выводы

В ходе выполнения данной лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

# 6 Список литературы

1. [Архитектура ЭВМ](https://esystem.rudn.ru/mod/resource/view.php?id=1030552)