Отчёт по лабораторной работе №4

Архитектура компьютеров и операционные системы.

Брыляков Никита Евгеньевич

Содержание

# 1 Цель работы

Освоение 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. Задание 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 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. 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!

Создаю каталог для работы с программами на языке ассемблера NASM: (рис. [??]).

Создание каталога

Создание каталога

Перехожу в созданный каталог (рис. [??]).

Переход в каталог

Переход в каталог

Создаю текстовый файл с именем hello.asm (рис. [??]).

Создание текстового файла

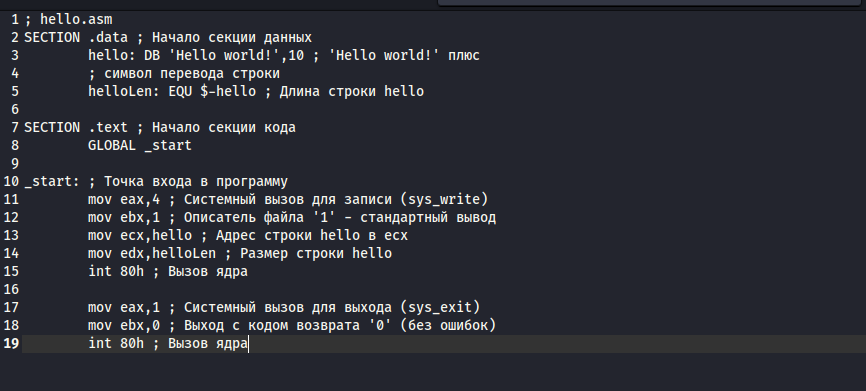
Создание текстового файла

Открываю этот файл с помощью mousepad (рис. [??]).

Открытие файла в текстовом редакторе

Открытие файла в текстовом редакторе

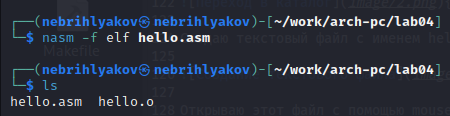
Ввожу в него следующий текст: (рис. [??]).



Заполнение файла

## 4.2 Транслятор NASM

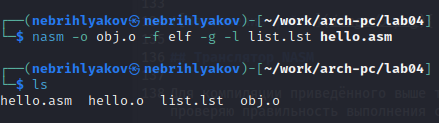
Для компиляции приведённого выше текста программы «Hello World» пишу необходимую команду и сразу проверяю правильность выполнения с помощью команды ls: (рис. [??]).



Компиляция текста программы

## 4.3 Расширенный синтаксис командной строки NASM

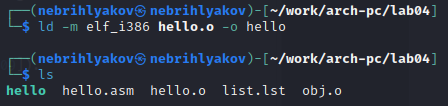
Cкомпилирую исходный файл hello.asm в obj.o и сразу проверяю правильность выполнения с помощью команды ls:(рис. [??]).



Компиляция текста программы

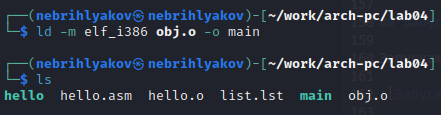
## 4.4 Компоновщик LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello и проверяю с помощью ls правильность выполнения команды. (рис. [??]).



Передача объектного файла на обработку компоновщику

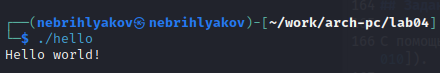
Выполняю следующую команду. Исполняемый файл будет иметь имя main. Проверяю с помощью команды ls (рис. [??]).



Передача объектного файла на обработку компоновщику

## 4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. [??]).



Запуск исполняемого файла

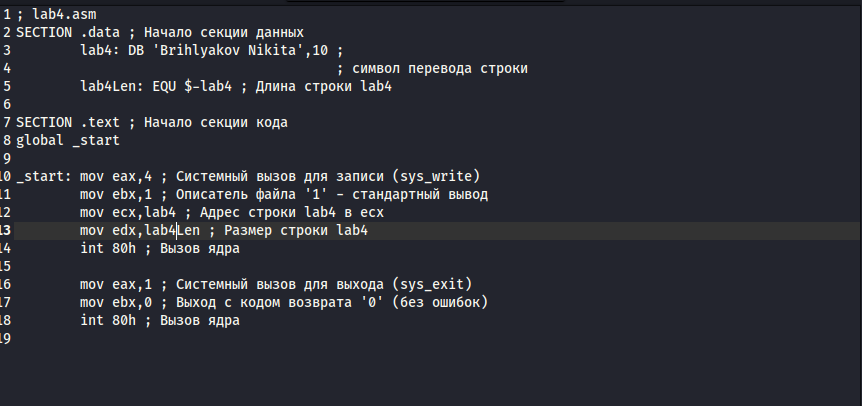
## 4.6 Задание для самостоятельной работы

С помощью команды cp создайте копию файла hello.asm с именем lab4.asm (рис. [??]).

Создание копии файла

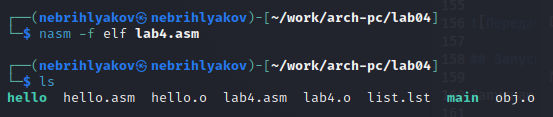
Создание копии файла

С помощью текстового редактора mousepad открываю файл lab5.asm и вношу изменения в текст программы так, чтобы она выводила мои имя и фамилию. (рис. [??]).



Изменение программы

Компилирую текст программы в объектный файл и проверяю с помощью ls, что файл lab4.o создан. (рис. [??]).



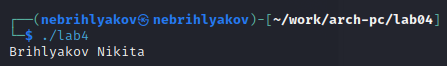
Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. [??]).

Передача объектного файла на обработку компоновщику

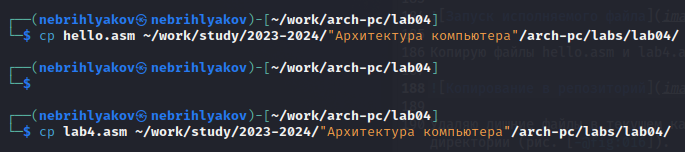
Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4 (рис. [??]).



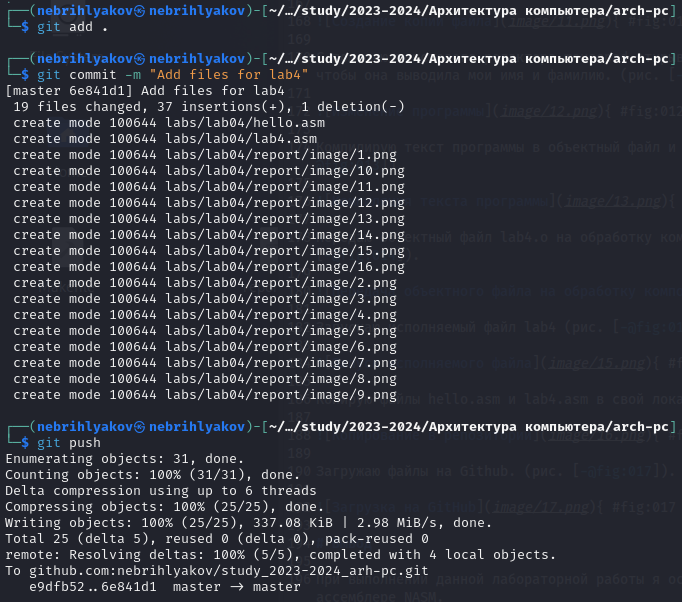
Запуск исполняемого файла

Копирую файлы hello.asm и lab4.asm в свой локальный репозиторий (рис. [??]).



Копирование в репозиторий

Загружаю файлы на Github. (рис. [??]).



Загрузка на GitHub

# 5 Вывод

При выполнении данной лабораторной работы я освоил процедуры компиляции и сборки программ, написанных на ассемблере NASM.

# 6 Список литературы

1. https://esystem.rudn.ru/pluginfile.php/2089084/mod\_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%964.%20%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B8%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0%20NASM.pdf