Отчёт по лабораторной работе №4

Дисциплина: архитектура компьютера

Луцкая Алиса Витальевна

Содержание

# 1 Цель работы

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

# 2 Задание

1. Создание программы Hello world!
2. Работа с транслятором NASM
3. Работа с расширенным синтаксисом командной строки NASM
4. Работа с компоновщиком 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 Создание программы Hello world!

Создаю с помощью mkdir нужный каталог ~/work/arch-pc/lab04, с помощью cd перхожу в нее (рис. 1).

Рис. 1: Перемещение между директориями

Рис. 1: Перемещение между директориями

Создаю с помощью утилиты touch файл hello.asm (рис. 2).

Рис. 2: Создание файла

Рис. 2: Создание файла

Открываю созданный файл в gedit и заполняю его, вставляя программу для вывода “Hello word!” (рис. 3).

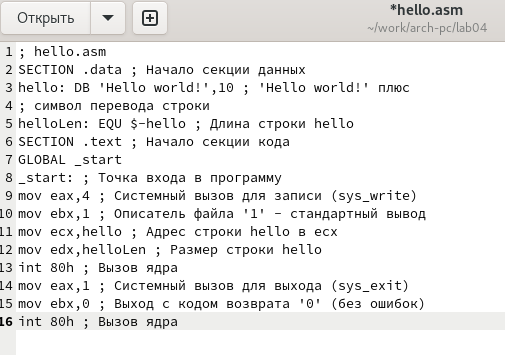


Рис. 3: Заполнение файла

## 4.2 Работа с транслятором NASM

Превращаю программу в объектный код с помощью транслятора NASM. Далее проверяю правильность выполнения команды с помощью у ls (рис. 4)

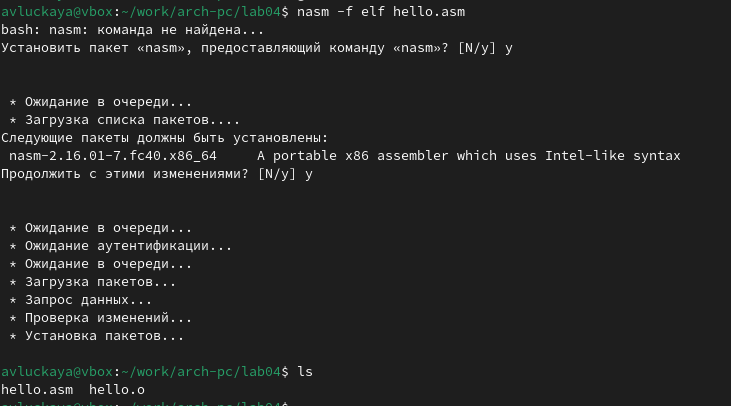


Рис. 4: Компиляция текста программы

## 4.3 Работа с расширенным синтаксисом командной строки NASM

Компилирую файл hello.asm в файл obj.o, далее проверяю правильность выполнения команды. (рис. 5).

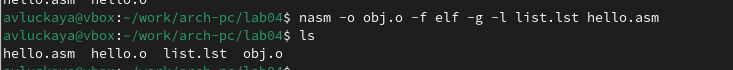


Рис. 5: Компиляция текста программы

## 4.4 Работа с компоновщиком LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello, проверяю с помощью утилиты ls правильность выполнения команды. (рис. 6).

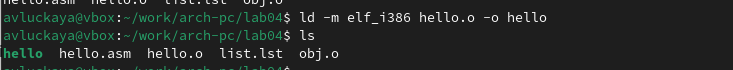


Рис. 6: Передача объектного файла на обработку компоновщику

Выполняю следующую команду, проверяю корректность выполнения (рис. 7). Исполняемый файл будет иметь имя main, т.к. после ключа -о было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o

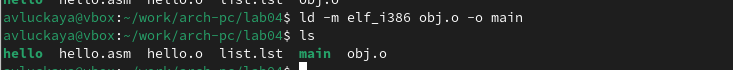


Рис. 7: Передача объектного файла на обработку компоновщику

## 4.5 Запуск исполняемого файла

Запускаю для выполнение созданный файл hello (рис. 8).

Рис. 8: Запуск исполняемого файла

Рис. 8: Запуск исполняемого файла

## 4.6 Выполнение заданий для самостоятельной работы.

С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm, открываю созданный файл в gedit и изменяю его, чтобы она выводила мое имя(рис. 9).

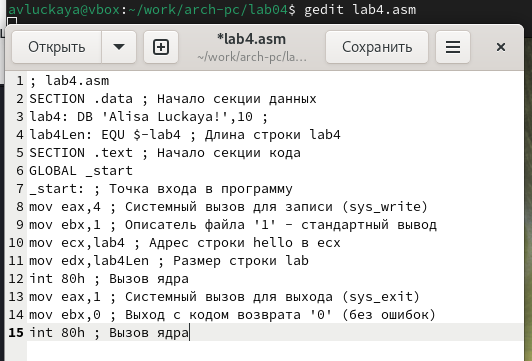


Рис. 9: Изменение прогграммы

Компилирую текст программы в объектный файл, проверяю, что файл lab4.o создан. (рис. 10).

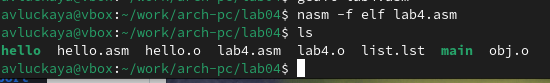


Рис. 10: Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab5, проверяю коректность выполнения (рис. 11).

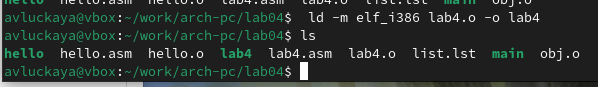


Рис. 11: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, действительно, выводятся мои имя и фамилия (рис. 12).

Рис. 12: Запуск исполняемого файла

Рис. 12: Запуск исполняемого файла

Копирую файл hello.asm в локальный репозиторий в каталог ~/work/study/2024-2024/“Архитектура компьютера”/arch-pc/labs/lab04/ с помощью ср (рис. 13).

Рис. 13: Копирование файла

Рис. 13: Копирование файла

Копирую файл lab4.asm в локальный репозиторий в каталог ~/work/study/2024-2024/“Архитектура компьютера”/arch-pc/labs/lab04/ с помощью ср, перехожу в этот каталог и проверяю коректность выполнения команд(рис. 14).

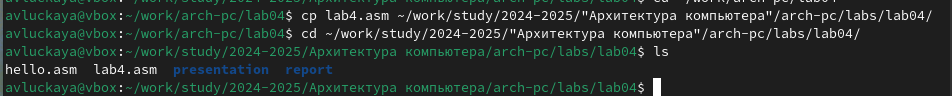


Рис. 14: Копирование файла

С помощью команд git add . и git commit добавляю файлы на GitHub, отправляю файлы на сервер с помощью команды git push (рис. 15).

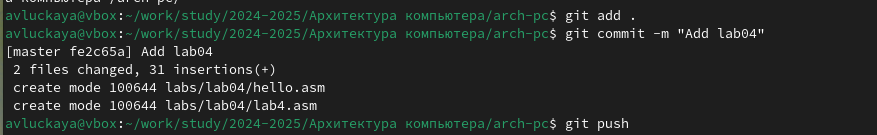


Рис. 15: Загрузка изменений на GitHub

# 5 Выводы

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

# 6 Список литературы

1. https://esystem.rudn.ru/pluginfile.php/1584628/mod\_resource/content/1/%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%965.pdf