Отчёт по лабораторной работе №4

Дисциплина: архитектура компьютеров

Карпачев Ярослав Олегович

Содержание

# 1 Цель работы

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

# 2 Задание

1. Cоздание 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 электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства (рис. 4.1). 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 (системной) плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора (ЦП) 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 (половинки 16-битных регистров). Например, AH (high AX) — старшие 8 бит регистра AX, AL (low AX) — младшие 8 бит регистра AX. 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 ЭВМ лежит принцип программного управления. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы. Программа состоит из машинных команд, которые указывают, какие операции и над какими данными (или операндами), в какой последовательности необходимо выполнить. Набор машинных команд определяется устройством конкретного процессора. 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. 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. Кроме того, в зависимости от команды при её выполнении могут проходить не все этапы. Более подробно введение о теоретических основах архитектуры ЭВМ см. в [9; 11]

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня,таких как C/C++, Perl, Python и пр.

# 4 Выполнение лабораторной работы

1. Программа Hello world!

Создание каталога с помощью команды mkdir и проверка ls. (рис.1 [??]).

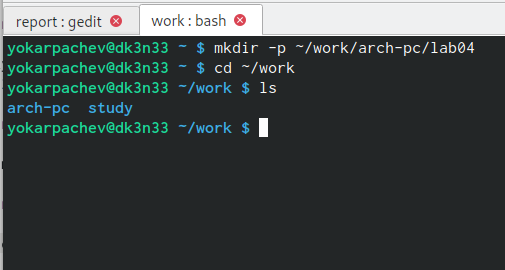


рис.1 Создание каталога

Перехожу в каталог ~/work/arch-pc и создайю текстовый файл с именем hello.asm (рис.2 [??]).

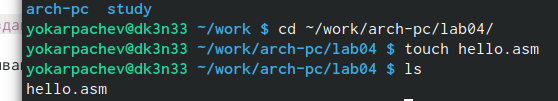


рис.2 Создание файла

Открываю созданный файл с помощью vim и ввожу программу (рис.3 [??]).

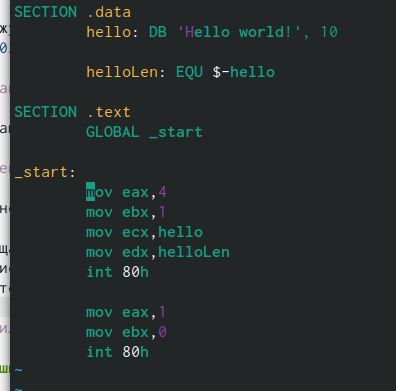


рис.3 Введение программы

1. NASM

Превращаю текст программы в объектный код, используя команду nasm -f elf hello.asm. (рис.4 [??]).

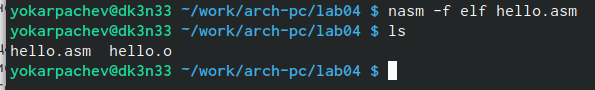


рис.4 Создание объектного файла

1. Расширенный синтаксис командной строки NASM

Ввожу команду nasm -o obj.o -f elf -g -l list.lst hello.asm. Данная команда компилирует файл в obj.o, кроме того, будет создан файл листинга list.lst (опция -l). (рис.5 [??]).

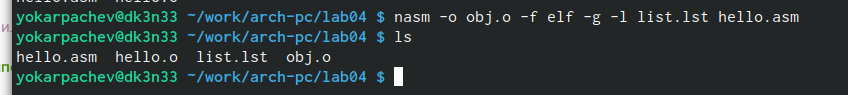


рис.5 Компиляция программы

1. Компоновщик

Передаю объектный файл на обработку компановщику, получаю исполняемый файл (рис.6 [??]).

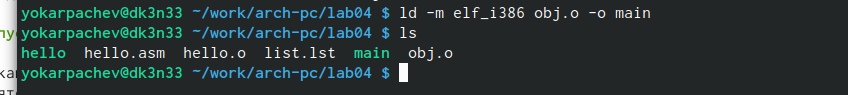


рис.6 Передача файла

Выполняю команду ld -m elf\_i386 obj.o -o main (рис.7 [??]).

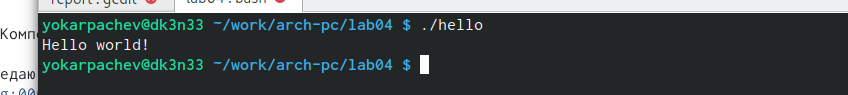
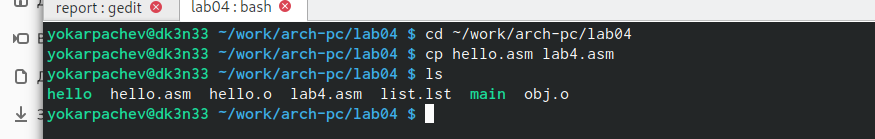


рис.7 Передача файла компановщику

1. Запуск исполняемого файла

Запускаю созданный исполняемый файл. На экран выводbтся Hello word! (рис.8 [??]).

r # Задание для самостоятельной работы

Перехожу в каталог ~/work/arch-pc/lab04, создаю копию файла hello.asm c именем lab4.asm с помощью команды cp. С помощью текстового редактора открываю файл lab4.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис.9 [??]).

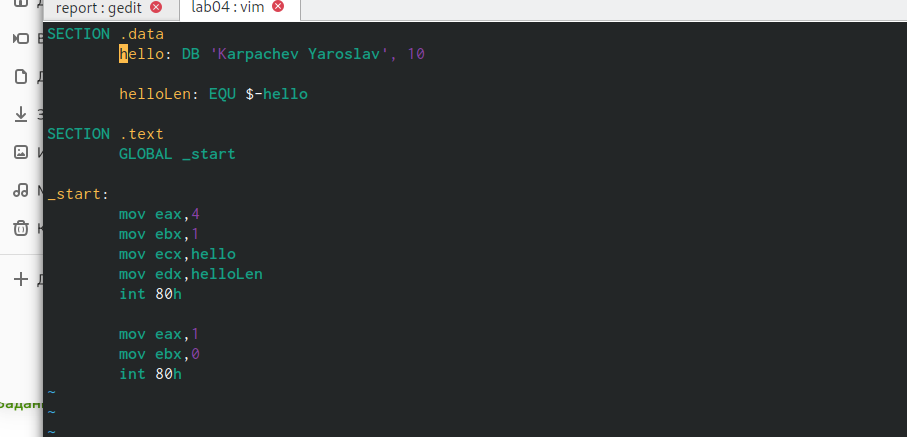


рис.9 Копирование файла и введение программы

Создаю объектный файл, передаю на обработку компановщику. Полученный исполняемый файл запускаю. На экран выводятся моё имя и моя фамилия.(рис.10 [??]).

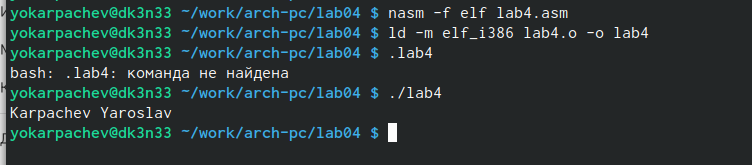


рис.10 Компиляция, запуск и обработка исполняемого файла

Копирование файлов в репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/study\_2023-2024\_arh-pc/labs/lab04 (рис.11 [??]).

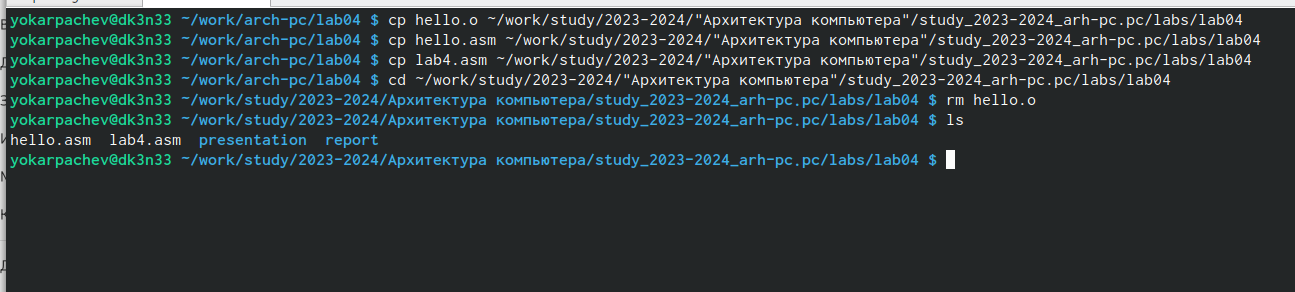


рис.11 Копирование файлов

Добавляю файлы в коммит (рис.12 [??]).

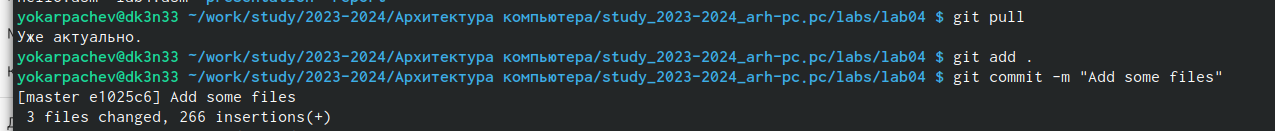


рис.12 Добавление файлов

Отправка файлов на сервер с помощью git push (рис.13 [??]).

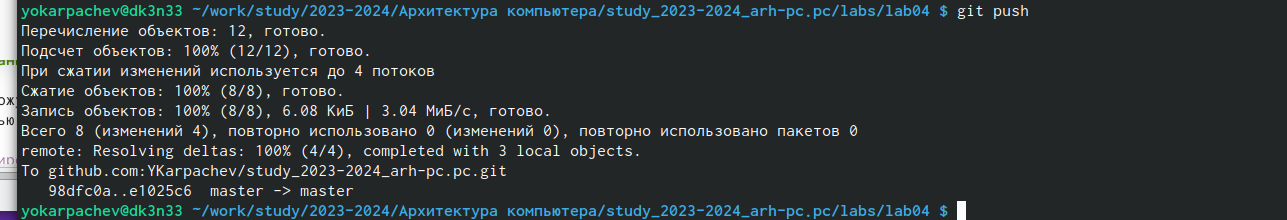


рис.13 Отправлка файлов

# 5 Выводы

Я освоил процедуры компиляции и сборки программ, написанных на ассемблере NASM.