Отчет по лабораторной работе №4

Дисциплина: Архитектура компьютера

Краснова Камилла Геннадьевна

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 электронно-вычислительной машины (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 (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. 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. 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.

# 4 Выполнение лабораторной работы

4.1 Создание программы Hello world! Создаю каталог для работы с программами, перехожу в него с помощью утилиты cd (рис. 1).

Рис. 1: Создание каталога

Рис. 1: Создание каталога

Создаю в текущем каталоге текстовый файл hello.asm, используя touch, и открываю его с помощью текстового редактора gedit (рис. 2).

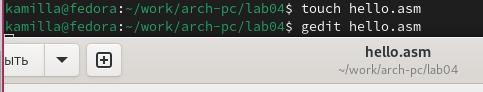


Рис. 2: Создание пустого каталога

Заполняю файл, вставляя в него программу для вывода “Hello world!” (рис. 3).

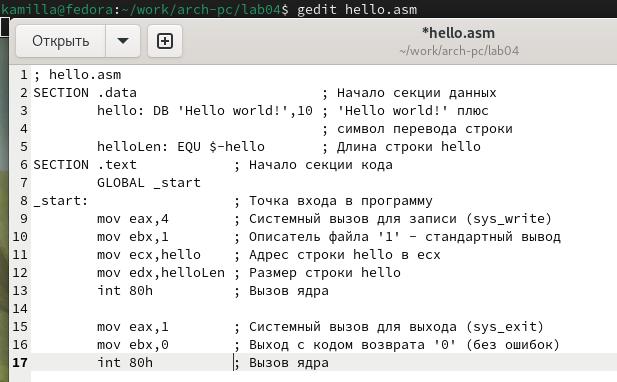


Рис. 3: Заполнение файла

4.2 Работа с транслятором NASM Компилирую текст программы “Hello world!”, используя команду nasm -f elf hello.asm. Проверяю, что создан объектный файл hello.o (рис. 4).

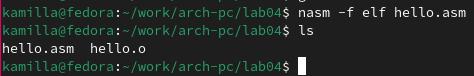


Рис. 4: Компиляция текста программы

4.3 Работа с расширенным синтаксисом командной строки NASM Компилирую файл hello.asm в obj.o. Далее проверяю, что файлы были созданы (рис. 5).

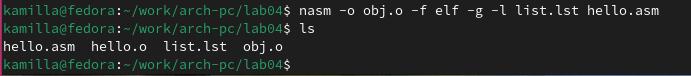


Рис. 5: Компиляция текста программы

4.4 Работа с компоновщиком LD Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello. Далее проверяю правильность выполнения команды (рис. 6).

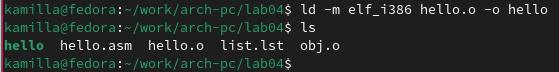


Рис. 6: Передача объектного файла на обработку компоновщику

Выполняю следующую команду (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.

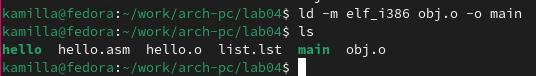


Рис. 7: Передача объектного файла на обработку компоновщику

4.5 Запуск исполняемого файла Запускаю на выполнение созданный исполняемый файл hello (рис. 8).

Рис. 8: Запуск исполняемого файла

Рис. 8: Запуск исполняемого файла

4.6 Выполнение заданий для самостоятельной работы С помощью команды cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm (рис. 9).

Рис. 9: Создание копии файла

Рис. 9: Создание копии файла

С помощью текстового редактора gedit открываю файл lab4.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию (рис. 10).

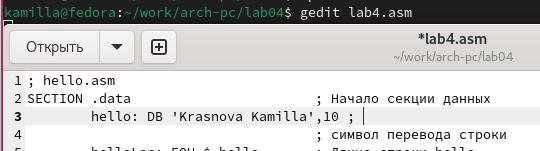


Рис. 10: Изменение программы

Компилирую текст программы в объектный файл. Проверяю, что файл lab4.o создан (рис. 11).

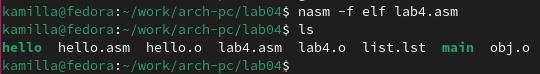


Рис. 11: Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. 12).

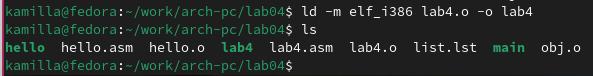


Рис. 12: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, вижу, что на экран действительно выводятся мои имя и фамилия (рис. 13).

Рис. 13: Запуск исполняемого файла

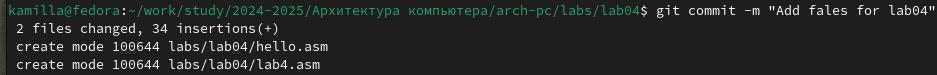
Рис. 13: Запуск исполняемого файла

Копирую файлы hello.asm и lab4.asm в свой локальный репозиторий, проверяю с помощью ls, что файлы скопировались (рис. 14).

Рис. 14: Копирование файлов

Рис. 14: Копирование файлов

С помощью команд git add . (рис. **¿fig:0015?**) и git commit (рис. **¿fig:0016?** добавляю файлы на GitHub.

Добавление изменений 

Отправляю файлы на сервер с помощью команды git push (рис. 15).

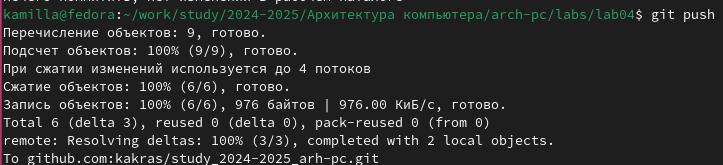


Рис. 15: Отправка файлов

# 5 Выводы

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

# Список литературы

1. [Архитектура ЭВМ](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)