Отчёт по лабораторной работе №4

Дисциплина: Архитектура компьютера

Абрамова Ульяна Михайловна

Содержание

# 1 Цель работы

Освоение 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:

- формирование адреса в памяти очередной команды;  
- считывание кода команды из памяти и её дешифрация;  
- выполнение команды;  
- 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

1. Создание программы Hello world! С помощью утилиты cd перемещаюсь в каталог, в котором буду работать (рис. 1)

Рис. 1: Перемещение в домашний каталог

Рис. 1: Перемещение в домашний каталог

Создаю в текущем каталоге пустой текстовый файл hello.asm с помощью утилиты touch, открываю созданный файл в текстовом редакторе gedit и заполняю в него программу для вывода “Hello world!” (рис. 2)

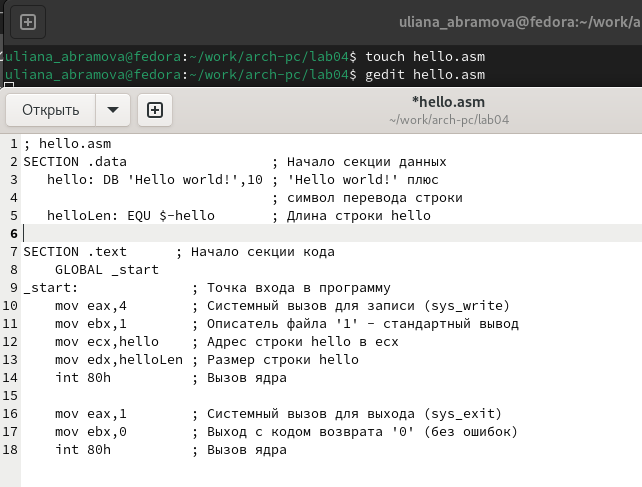


Рис. 2: Создание и заполнения файла для вывода “Hello world!”

1. Работа с транслятором NASM Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду nasm -f elf hello.asm, ключ -f указывает транслятору nasm, что требуется создать бинарный файл в формате ELF (рис. **¿fig:004?**). Далее проверяю правильность выполнения команды с помощью утилиты ls: действительно, создан файл “hello.o”.

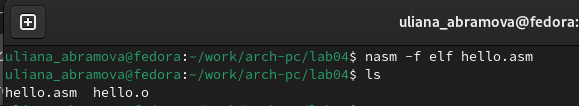


Рис. 3: Превращение текста программы для вывода “Hello world! в объектный код

1. Работа с расширенным синтаксисом командной строки NASM Ввожу команду, которая скомпилирует файл hello.asm в файл obj.o, при этом в файл будут включены символы для отладки (ключ -g), также с помощью ключа -l будет создан файл листинга list.lst (рис. **¿fig:004?**). Далее проверяю с помощью утилиты ls правильность выполнения команды.

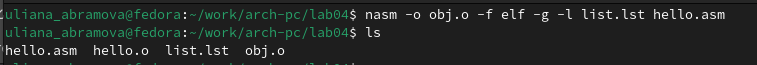


Рис. 4: Ввод команды для скомпилирования файла hello.asm

1. Работа с компоновщиком LD Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (рис. **¿fig:005?**). Ключ -о задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды.

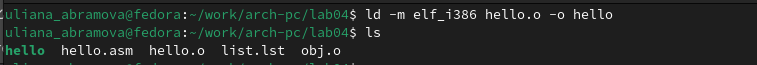


Рис. 5: Передача файла hello.o на обработку компононовщику LD

Выполняю следующую команду (рис. **¿fig:006?**). 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

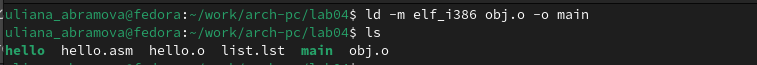


Рис. 6: Аналогичная операция

1. Запуск исполняемого файла Запускаю на выполнение созданный исполняемый файл hello (рис. **¿fig:007?**).

Рис. 7: Запуск

Рис. 7: Запуск

1. Выполнение заданий для самостоятельной работы С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm (рис. **¿fig:008?**).

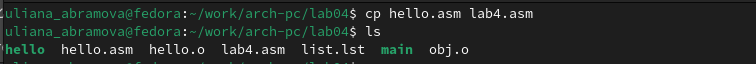


Рис. 8: Копирование файла

С помощью текстового редактора gedit открываю файл lab4.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис. **¿fig:009?**).

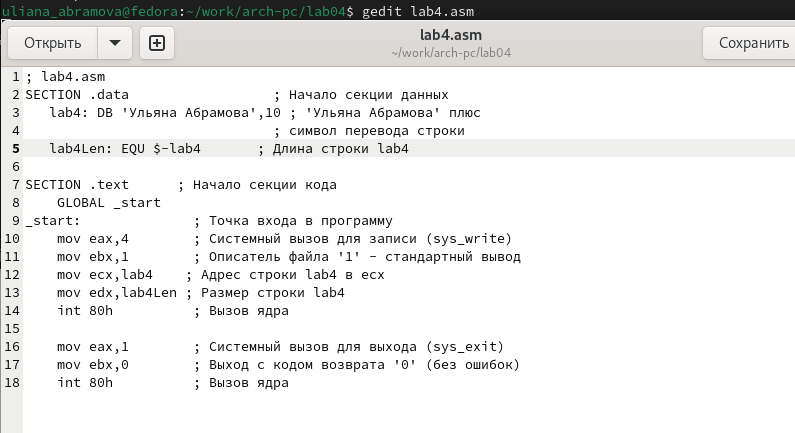


Рис. 9: Редактирование файла

Компилирую текст программы в объектный файл. Проверяю с помощью утилиты ls, что файл lab4.o создан. Передаю объектный файл lab5.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. **¿fig:010?**).

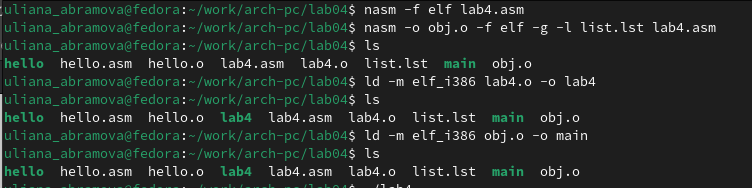


Рис. 10: Компилирование и передача объектного файла на обработку компановщику

Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия (рис. **¿fig:011?**).

Рис. 11: Запуск

Рис. 11: Запуск

Далее добавляю файлы на GitHub и отправляю на сервер с помощью команды git push (рис. **¿fig:012?**).

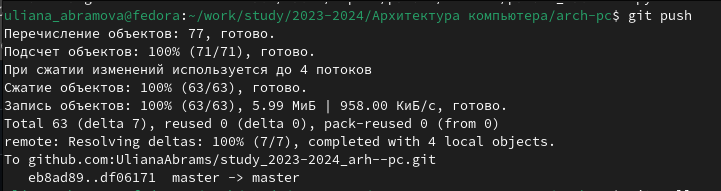


Рис. 12: Отправка файлов

# 5 Выводы

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

# 6 Список литературы

1. [Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089533/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)