Отчёт по лабораторной работе №4

Архитектура компьютера

Сахно Никита НКАбд-05-23

Содержание

[Цель работы 1](#_Toc148698595)

[Теоретическое введение 1](#_Toc148698596)

[Задание 2](#_Toc148698597)

[Выполнение лабораторной работы 3](#_Toc148698598)

[Создание программы Hello world! 3](#_Toc148698599)

[Работа с транслятором NASM 3](#_Toc148698600)

[Работа с расширенным синтаксисом командной строки NASM 3](#_Toc148698601)

[Работа с компоновщиком LD 3](#_Toc148698602)

[Запуск исполняемого файла 4](#_Toc148698603)

[Выполнение заданий для самостоятельной работы. 4](#_Toc148698604)

[Выводы 4](#_Toc148698605)

# Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

# 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.

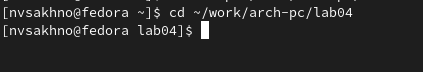
# Задание

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.

# Выполнение лабораторной работы

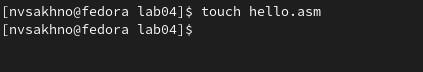
## Создание программы Hello world!

С помощью утилиты cd перемещаюсь в каталог, в котором буду работать (рис. 1)



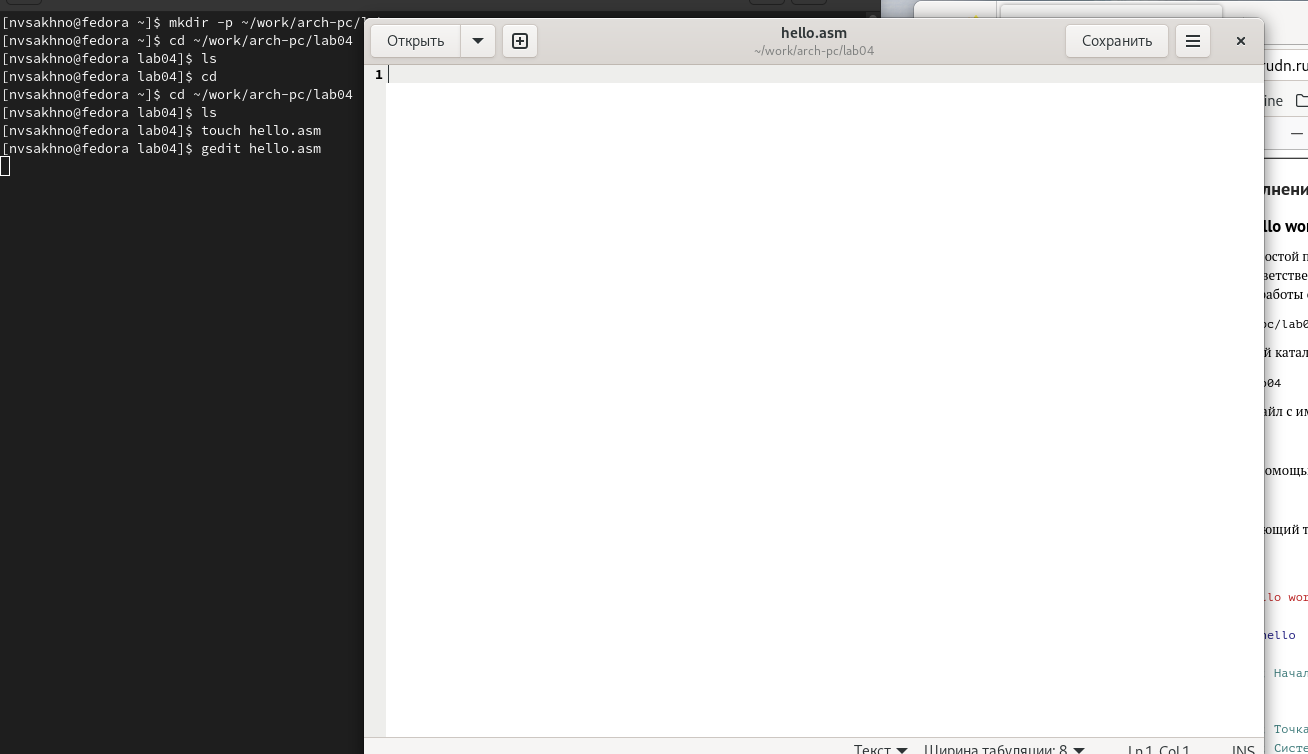
*Перемещение между директориями*

Создаю в текущем каталоге пустой текстовый файл hello.asm с помощью утилиты touch (рис. 2)

**

*Создание пустого файла*

Открываю созданный файл в текстовом редакторе через команду “gedit”. Затем вставляю туда программу для вывода “Hello word!”. (рис. 3)



Открытие файла в текстовом редакторе

## Работа с транслятором NASM

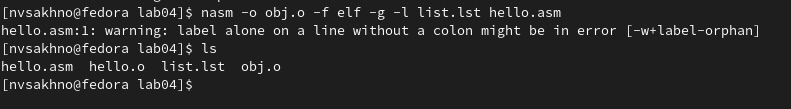
Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду nasm -f elf hello.asm, ключ -f указывает транслятору nasm, что требуется создать бинарный файл в формате ELF (рис. 4). Далее проверяю правильность выполнения команды с помощью утилиты ls: действительно, создан файл “hello.o”.

**

*Компиляция текста программы*

## Работа с расширенным синтаксисом командной строки NASM

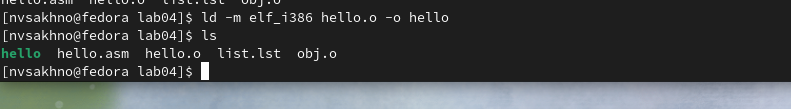
Ввожу команду, которая скомпилирует файл hello.asm в файл obj.o, при этом в файл будут включены символы для отладки (ключ -g), также с помощью ключа -l будет создан файл листинга list.lst (рис. [-@fig:005]). Далее проверяю с помощью утилиты ls правильность выполнения команды.

**

*Компиляция текста программы*

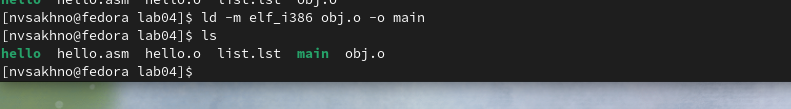
## Работа с компоновщиком LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (рис. 6). Ключ -о задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды.

**

*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

**

*Передача объектного файла на обработку компоновщику*

## Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. 8).

**

*Запуск исполняемого файла*

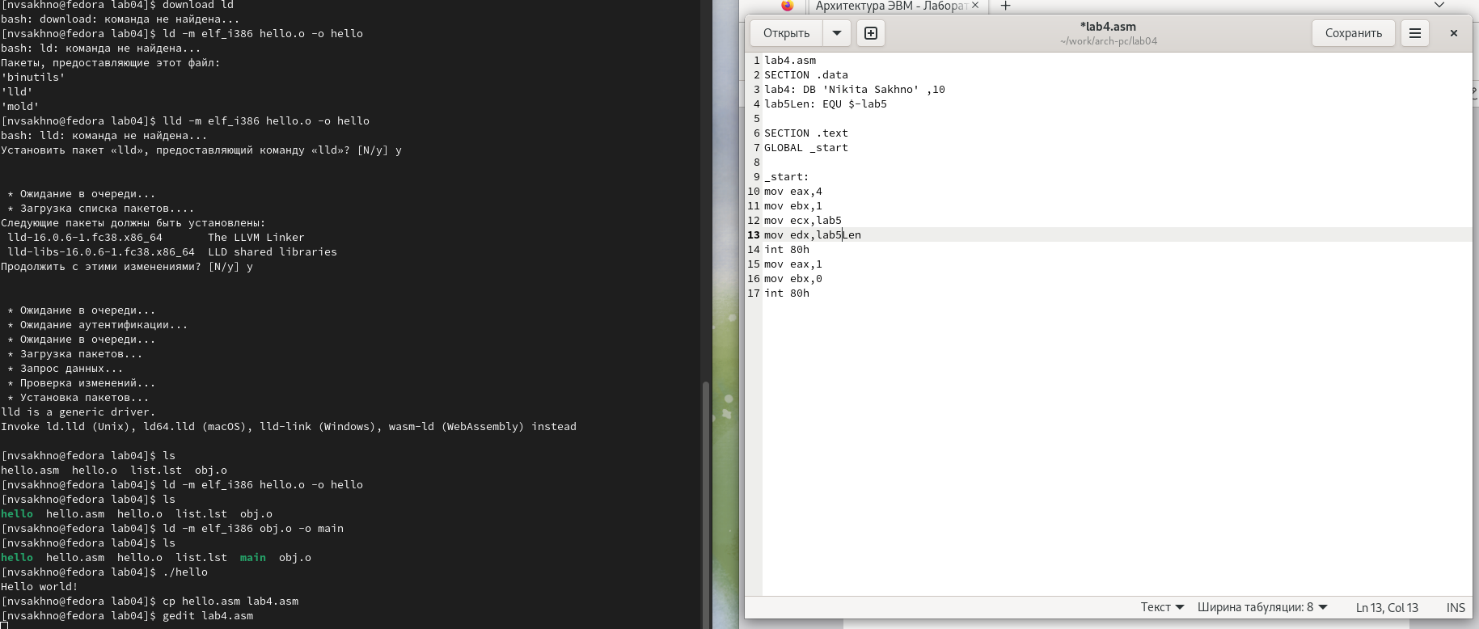
## Выполнение заданий для самостоятельной работы.

С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm (рис. 9).

**

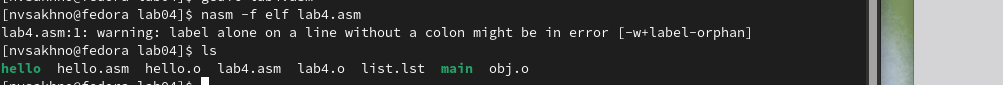
*Создание копии файла*

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

**

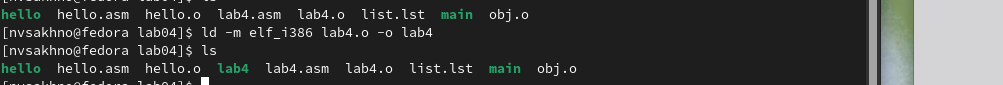
*Изменение программы*

*Компилирую текст программы в объектный* файл (рис.11). Проверяю с помощью утилиты ls, что файл lab4.o создан.

**

*Компиляция текста программы*

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab5 (рис. 12).

**

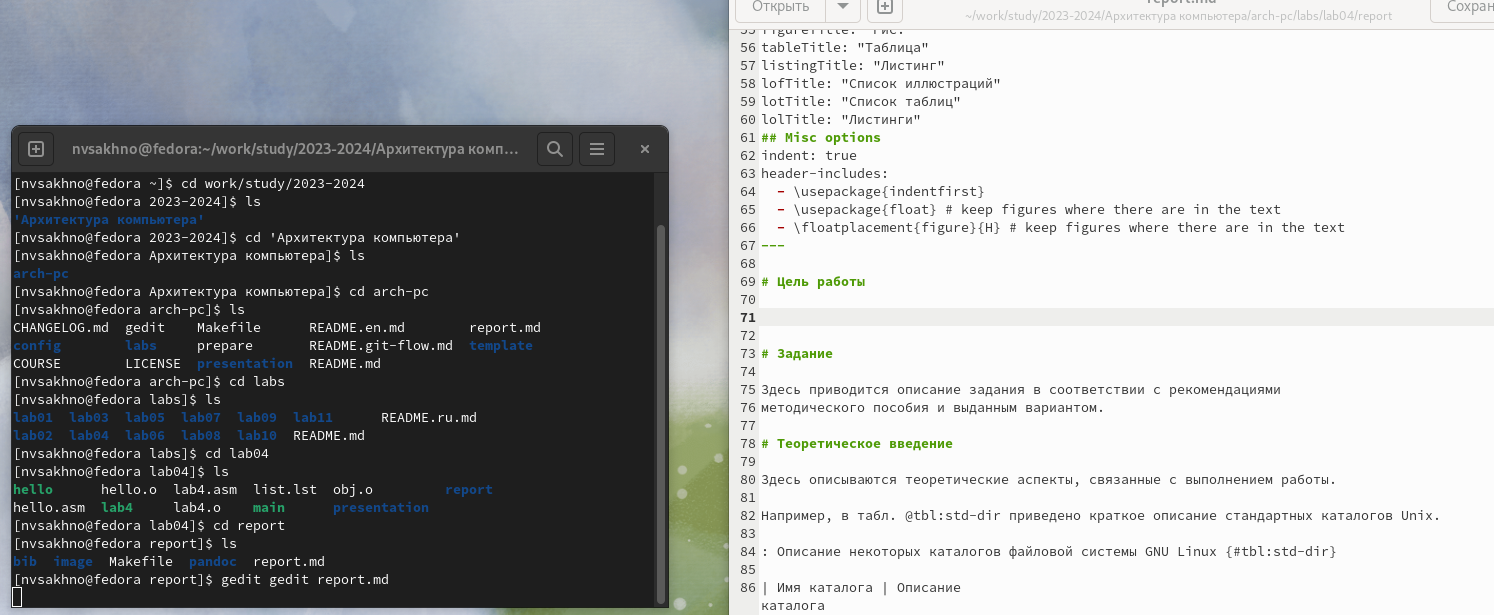
*Передача объектного файла на обработку компоновщику*

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

**

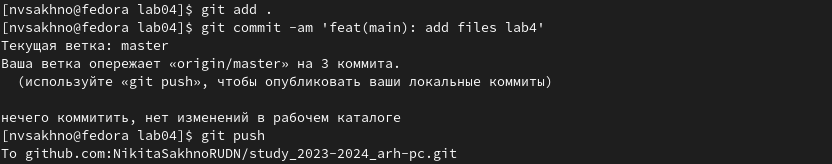
*Запуск исполняемого файла*

Я закончил выполнение лабораторной работы и начал писать отчет, с помощью утилиты cd (Рис 14).

**

*Создание отчета*

И затем выгружаю все на гитхаб (Отчет и файлы hello и lab4) (Рис 15).

**

*Выгрузка файлов на гитхаб*

# Выводы

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