РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра компьютерных и информационных наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

дисциплина: Архитектура компьютера

Студент: Симакова Алина

Группа: НКАбд-05-25

МОСКВА

2025 г.

Оглавление

1 Цель работы

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

2 Задание

На основе методических указаний создать программу «Hello world!», научиться работать с транслятором NASM, провести работу с расширенным синтаксисом командной строки NASM, научиться работать с компоновщиком LD, произвести запуск исполняемого файла и выполнить задания 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 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.

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 Программа Hello world!

В домашней директории создаю каталог, в котором буду хранить файлы для текущей лабораторной работы. (рис. 4.1)

```
simakovaalina@fedora:~$ mkdir -p ~/work/arch-pc/lab04
simakovaalina@fedora:~$ cd ~/work/
simakovaalina@fedora:~/work$ cd ~/work/arch-pc/lab04/
simakovaalina@fedora:~/work/arch-pc/lab04$
```

Рис. 4.1: Создание рабочей директории

Создаю в нем файл hello.asm, в котором буду писать программу на языке ассемблера. (рис. 4.2)

```
simakovaalina@fedora:~$ mkdir -p ~/work/arch-pc/lab04
simakovaalina@fedora:~$ cd ~/work/
simakovaalina@fedora:~/work$ cd ~/work/arch-pc/lab04/
simakovaalina@fedora:~/work/arch-pc/lab04$ touch hello.asm
simakovaalina@fedora:~/work/arch-pc/lab04$ mousepad hello.asm
```

Рис. 4.2: Создание .asm файла

С помощью редактора пишу программу в созданном файле. (рис. 4.3)

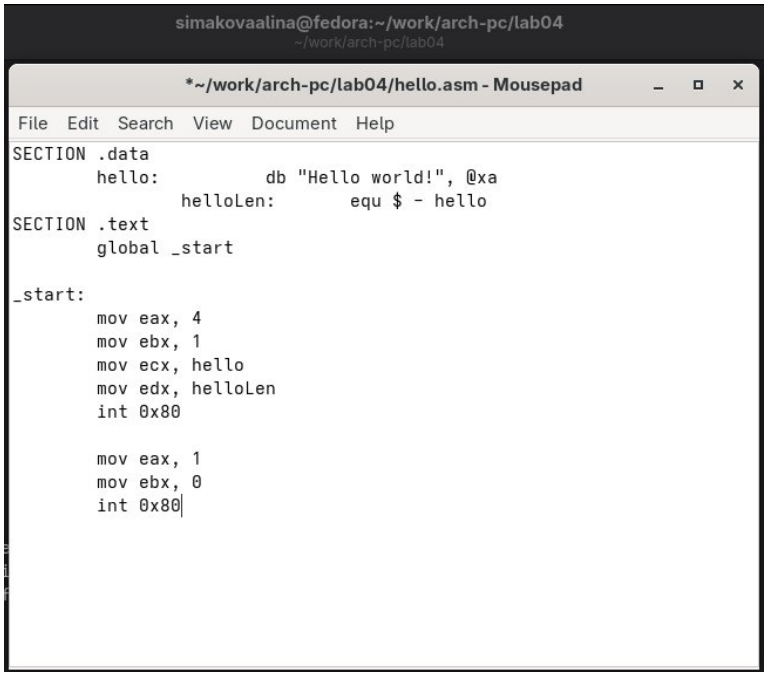

Рис. 4.3: Редактирование файла

4.2 Транслятор NASM

Далее компилирую с помощью NASM свою программу. (рис. 4.4)

```
simakovaalina@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm
simakovaalina@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
simakovaalina@fedora:~/work/arch-pc/lab04$
```

Рис. 4.4: Компиляция программы

4.3 Расширенный синтаксис командной строки NASM

Выполняю команду, указанную на (рис. 4.5), она скомпилировала исходный файл hello.asm в obj.o, расширение .o говорит о том, что файл - объектный, помимо него флаги -g -l подготовят файл отладки и листинга соответственно.

```
simakovaalina@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm
simakovaalina@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
simakovaalina@fedora:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
simakovaalina@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
simakovaalina@fedora:~/work/arch-pc/lab04$
```

Рис. 4.5: Возможности синтаксиса NASM

4.4 Компоновщик LD

Затем мне необходимо передать объектный файл компоновщику. Соответственно, делаю это с помощью команды ld. (рис. 4.6)

```
simakovaalina@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm
simakovaalina@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
simakovaalina@fedora:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
simakovaalina@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
simakovaalina@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
ld: warning: cannot find entry symbol _start; not setting start address
simakovaalina@fedora:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o
simakovaalina@fedora:~/work/arch-pc/lab04$
```

Рис. 4.6: Отправка файла компоновщику

Выполняю следующую команду …, результатом исполнения команды будет созданный файл main, скомпонованный из объектного файла obj.o. (рис. 4.7)

```
simakovaalina@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm
simakovaalina@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
simakovaalina@fedora:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
simakovaalina@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
simakovaalina@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
ld: warning: cannot find entry symbol _start; not setting start address
simakovaalina@fedora:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o
simakovaalina@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
ld: warning: cannot find entry symbol _start; not setting start address
simakovaalina@fedora:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
simakovaalina@fedora:~/work/arch-pc/lab04$
```

Рис. 4.7: Создание исполняемого файла

4.5 Запуск исполняемого файла

Запускаю исполняемый файл из текущего каталога. (рис. 4.8)

```
simakovaalina@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm -o hello.o
simakovaalina@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
simakovaalina@fedora:~/work/arch-pc/lab04$ ./hello
Hello world!
simakovaalina@fedora:~/work/arch-pc/lab04$
```

Рис. 4.8: Запуск программы

5 Задания для самостоятельной работы

Для начала создаю копию файла для последующей работы с ней. (рис. 4.9)

```
simakovaalina@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm -o hello.o
simakovaalina@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
simakovaalina@fedora:~/work/arch-pc/lab04$ ./hello
Hello world!
simakovaalina@fedora:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
simakovaalina@fedora:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
simakovaalina@fedora:~/work/arch-pc/lab04$
```

Рис. 4.9 – Создание копии

Затем редактирую копию файла, заменив текст на свое имя и фамилию. (рис. 4.10)

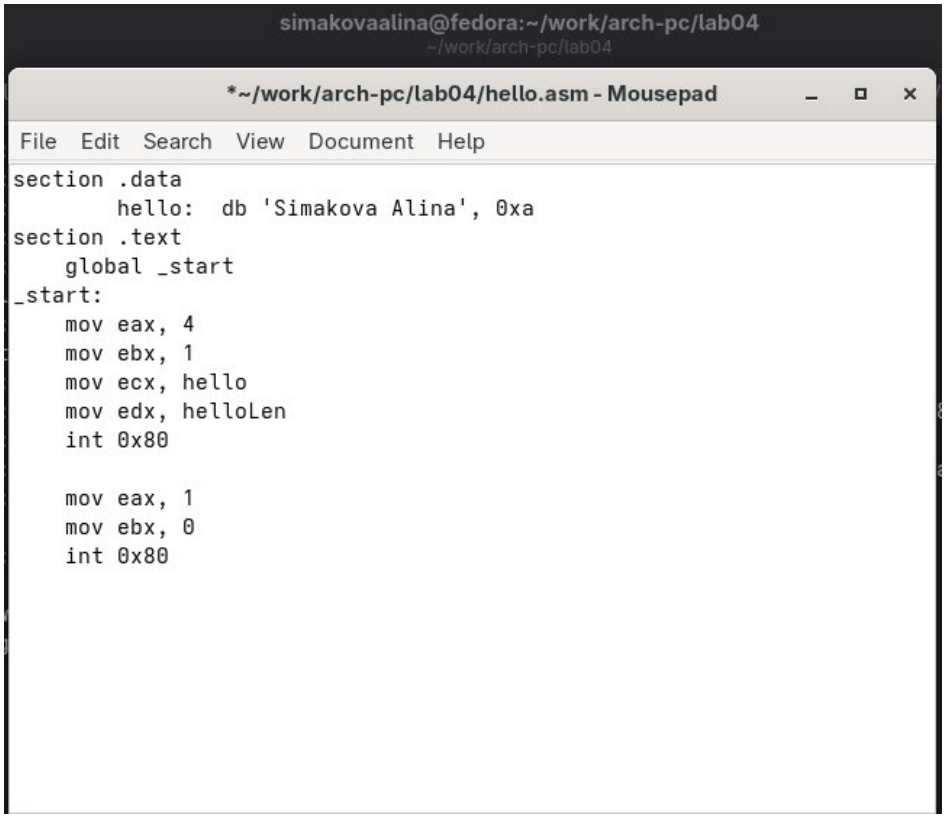

Рис. 4.10: – Редактирование копии

После этого транслирую копию файла в объектный файл, компоную и запускаю. (рис. 4.11)

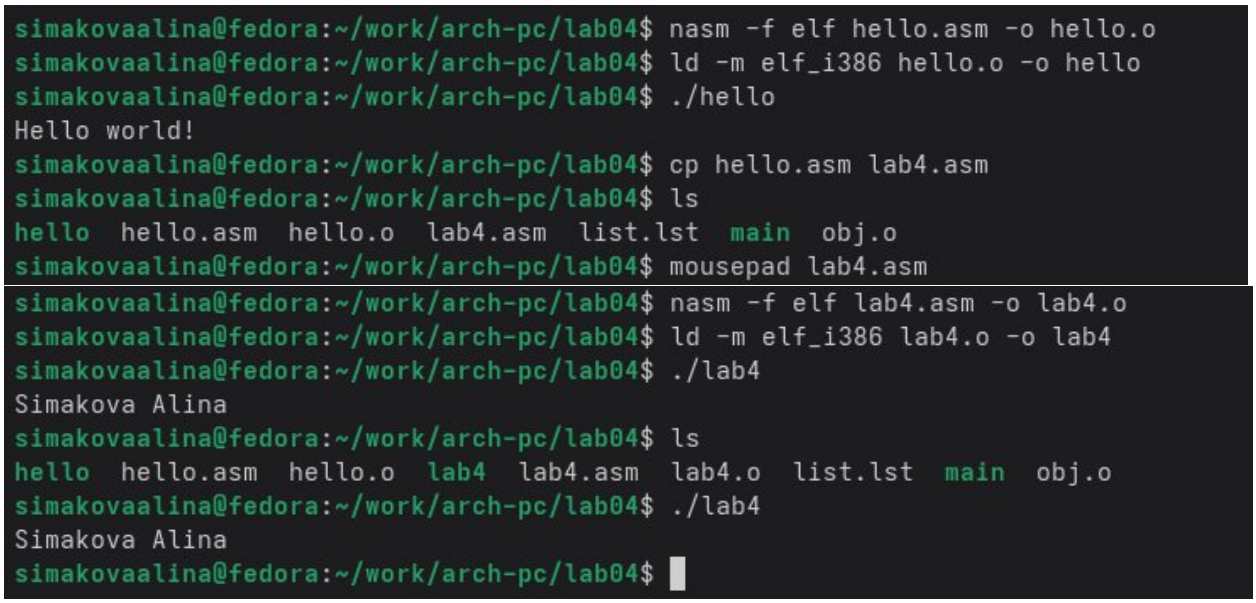

Рис. 4.11: Проверка работоспособности скомпонованной

Убедившись в корректности работы программы, копирую рабочие файлы в свой локальный репозиторий. (рис. 4.12)

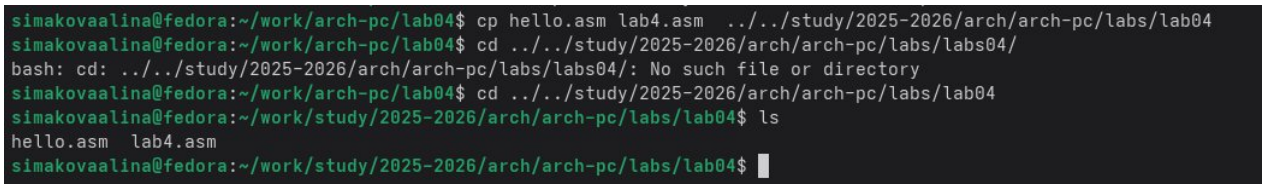

Рис. 4.12 – Отправка файлов в локальный репозиторий

6 Выводы

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

Список литературы

1. https://esystem.rudn.ru/pluginfile.php/2089085/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%965.%20%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B%20%D1%81%20Midnight%20Commander%20%28%29.%20%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B%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.%20%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B5%20%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2%D1%8B%20%D0%B2%20%D0%9E%D0%A1%20GNU%20Linux.pdf
2. https://esystem.rudn.ru/mod/page/view.php?id=1030492
3. https://esystem.rudn.ru/mod/resource/view.php?id=1030495
4. https://esystem.rudn.ru/mod/resource/view.php?id=1030496