

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

дисциплина: *Архитектура компьютера*

Студент: Ксения Танчила

Группа: НКАбд-05-25

МОСКВА

2025 г.

Оглавление

Список иллюстраций.....	3
Цель работы.....	4
Задание.....	5
Теоритическое введение.....	6
Выполнение лабораторной работы.....	8
Программа Hello world!.....	8
Транслятор NASM.....	8
Запуск исполняемого файла.....	10
Задания для самостоятельной работы.....	10
Выводы.....	12
Список литературы.....	13

Список иллюстраций

Рис. 4.1.1 — Создание текстового файла <code>hello.asm</code>	8
Рис. 4.1.2 — Текст программы.....	8
Рис. 4.2.1 — Компиляция.....	8
Рис. 4.2.2 — Объектный файл.....	9
Рис. 4.2.3 — <code>nasm -o obj.o -f elf -g -l list.lst hello.asm</code>	9
Рис. 4.2.4 — Обработка компоновщика.....	9
Рис. 4.2.5 — <code>ld -m elf_i386 obj.o -o main</code>	10
Рис. 4.2.6 — <code>ld-help</code>	10
Рис. 4.3 — Запуск файла.....	10
Рис. 4.4.1 — Копирование файла <code>hello.asm</code> с именем <code>lab4.asm</code>	11
Рис. 4.4.2 — Запуск <code>lab4.asm</code>	11

1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM

2 Задание

Создание программы Hello world!, работа с транслятором NASM, работа с расширенным синтаксисом командной строки NASM, работа с компоновщиком LD, запуск исполняемого файла, 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 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*) — машинноориентированный 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!

Создала каталог для работы с программами на языке ассемблера NASM.

Перешла в созданный каталог.

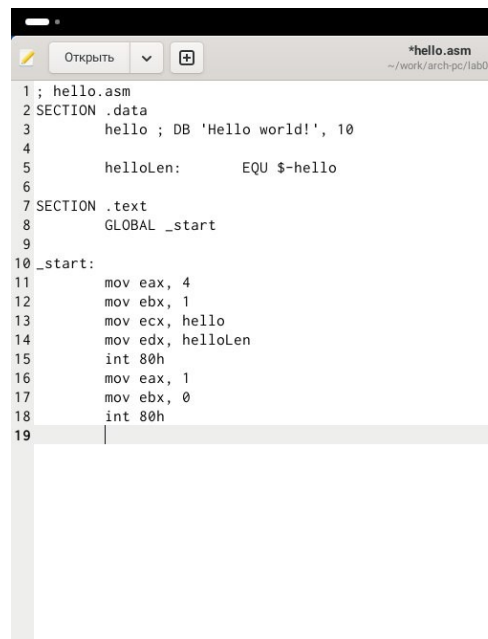
Создала текстовый файл с именем hello.asm.



```
kstanchila@dk6n17 ~ $ mkdir -p ~/work/arch-pc/lab04
kstanchila@dk6n17 ~ $ cd ~/work/arch-pc/lab04
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ touch hello.asm
```

Рис 4.1.1 — Создание текстового файла hello.asm

Открыла этот файл с помощью текстового редактора gedit и ввела в него текст программы.



```
*hello.asm
~/work/arch-pc/lab04

1 ; hello.asm
2 SECTION .data
3     hello ; DB 'Hello world!', 10
4
5     helloLen:    EQU $-hello
6
7 SECTION .text
8     GLOBAL _start
9
10 _start:
11     mov eax, 4
12     mov ebx, 1
13     mov ecx, hello
14     mov edx, helloLen
15     int 80h
16     mov eax, 1
17     mov ebx, 0
18     int 80h
19
```

Рис 4.1.2 — Текст программы

4.2 Транслятор NASM

Для компиляции текста программы «Hello World» написала команду `nasm -f elf hello.asm`.

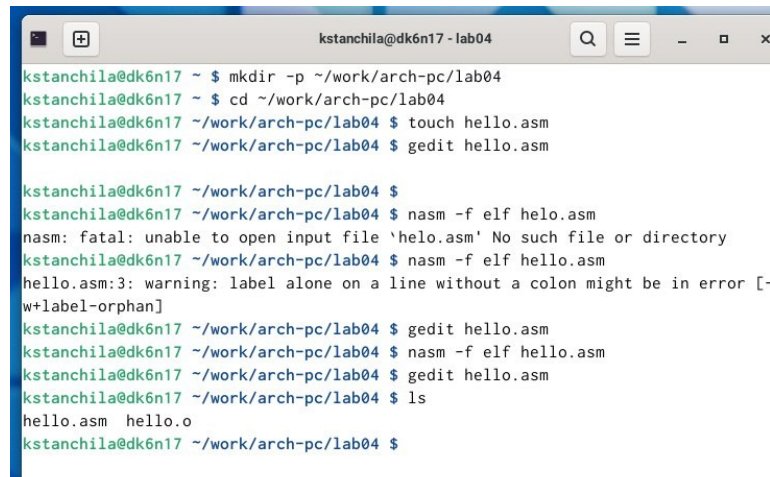


```
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 4.2.1 — Компиляция

С помощью команды ls проверила, что объектный файл был создан.

Объектный файл имеет имя hello.o

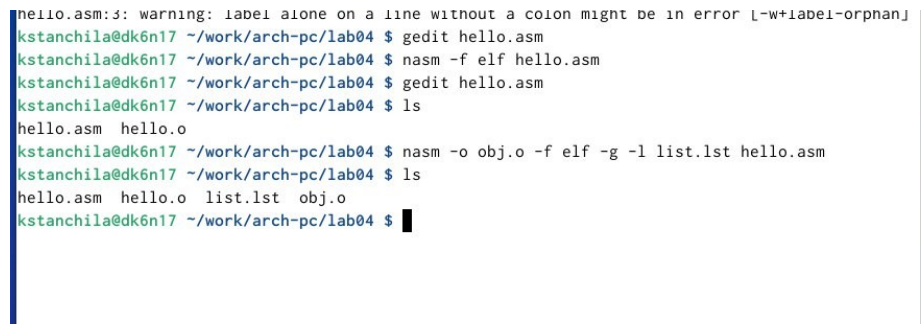


```
kstanchila@dk6n17 - lab04
kstanchila@dk6n17 ~ $ mkdir -p ~/work/arch-pc/lab04
kstanchila@dk6n17 ~ $ cd ~/work/arch-pc/lab04
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ touch hello.asm
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ gedit hello.asm

kstanchila@dk6n17 ~/work/arch-pc/lab04 $
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ nasm -f elf helo.asm
nasm: fatal: unable to open input file 'helo.asm' No such file or directory
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
hello.asm:3: warning: label alone on a line without a colon might be in error [-w+label-orphan]
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ gedit hello.asm
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ gedit hello.asm
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
kstanchila@dk6n17 ~/work/arch-pc/lab04 $
```

Рис. 4.2.2 — Объектный файл

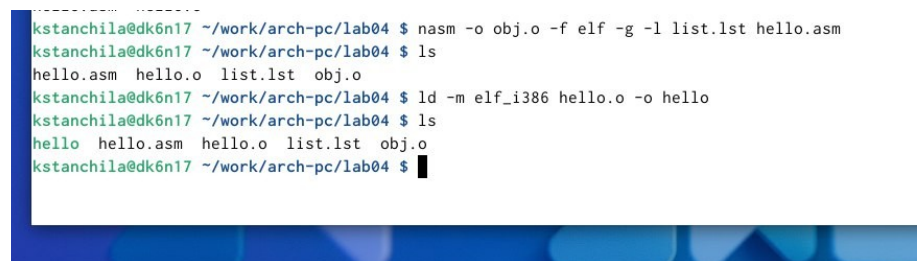
Выполнила команду `nasm -o obj.o -f elf -g -l list.lst hello.asm`.



```
hello.asm:3: warning: label alone on a line without a colon might be in error [-w+label-orphan]
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ gedit hello.asm
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ gedit hello.asm
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
kstanchila@dk6n17 ~/work/arch-pc/lab04 $
```

Рис. 4.2.3 — `nasm -o obj.o -f elf -g -l list.lst hello.asm`

Объектный файл передала на обработку компоновщику командой `ld -m elf_i386 hello.o -o hello`. С помощью команды ls проверила, что файлы были созданы.



```
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o
kstanchila@dk6n17 ~/work/arch-pc/lab04 $
```

Рис. 4.2.4 — Обработка компоновщика

Выполнила команду `ld -m elf_i386 obj.o -o main`.

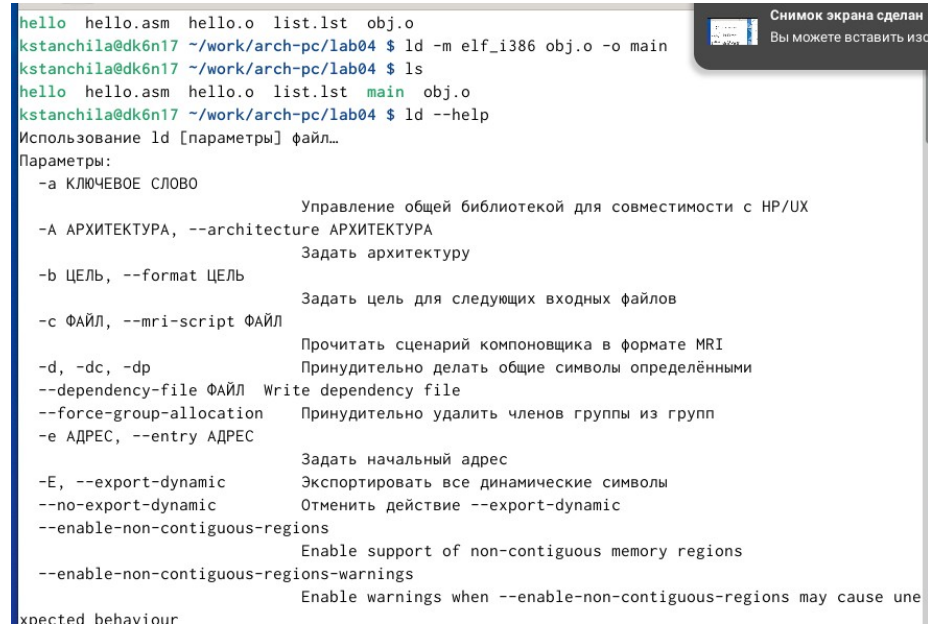
```
hello hello.asm hello.o list.lst obj.o
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o list.lst main obj.o
kstanchila@dk6n17 ~/work/arch-pc/lab04 $
```

Рис. 4.2.5 — `ld -m elf_i386 obj.o -o main`

Исполняемый файл имеет название `main`.

Объектный файл имеет имя `obj.o`.

Формат командной строки LD увидели, набрав `ld --help`.



```
hello hello.asm hello.o list.lst obj.o
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o list.lst main obj.o
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ld --help
Использование ld [параметры] файл...
Параметры:
-a КЛЮЧЕВОЕ СЛОВО                Управление общей библиотекой для совместимости с HP/UX
-A АРХИТЕКТУРА, --architecture АРХИТЕКТУРА    Задать архитектуру
-b ЦЕЛЬ, --format ЦЕЛЬ            Задать цель для следующих входных файлов
-c ФАЙЛ, --mri-script ФАЙЛ        Прочитать сценарий компоновщика в формате MRI
-d, -dc, -dp                      Принудительно делать общие символы определёнными
--dependency-file ФАЙЛ            Write dependency file
--force-group-allocation          Принудительно удалить членов группы из групп
-e АДРЕС, --entry АДРЕС          Задать начальный адрес
-E, --export-dynamic              Экспортировать все динамические символы
--no-export-dynamic              Отменить действие --export-dynamic
--enable-non-contiguous-regions    Enable support of non-contiguous memory regions
--enable-non-contiguous-regions-warnings Enable warnings when --enable-non-contiguous-regions may cause unexpected behaviour
```

Рис. 4.2.6 — `ld --help`

4.3 Запуск исполняемого файла

Запустили на выполнение созданный исполняемый файл, находящийся в текущем каталоге.



```
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ./hello
Hello world!
kstanchila@dk6n17 ~/work/arch-pc/lab04 $
```

Рис. 4.3 — Запуск файла

4.4 Задание для самостоятельной работы

1. В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создала копию файла `hello.asm` с именем `lab4.asm`.

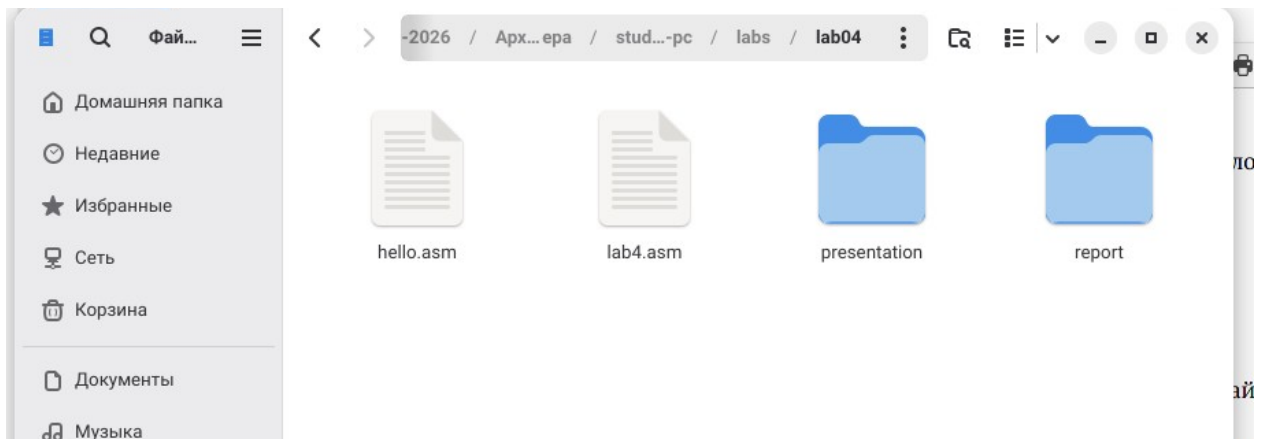


Рис. 4.4.1 — Копирование файла hello.asm с именем lab4.asm

2. С помощью текстового редактора gedit внесла изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с моими фамилией и именем.

3. Оттранслировала полученный текст программы lab4.asm в объектный файл. Выполнила компоновку объектного файла и запустила получившийся исполняемый файл.

```
kstanchila@dk6n17 ~/work/arch-pc/lab04 $ ./lab4
Tanchila Ksenia
kstanchila@dk6n17 ~/work/arch-pc/lab04 $
```

Рис. 4.4.2 — Запуск lab4.asm

4. Скопировала файлы hello.asm и lab4.asm в мой локальный репозиторий в каталог ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/. Загрузила файлы на Github.

6 Выводы

Создала программу Hello world!, научилась работать с транслятором NASM, научилась работать с расширенным синтаксисом командной строки NASM, научилась работать с компоновщиком LD, запустила исполняемый файл, выполнила задания для самостоятельной работы.

Список литературы

1. Пример выполнения лабораторной работы
2. Курс на ТУИС
3. Лабораторная работа №4
4. Программирование на языке ассемблера NASM Столяров А. В