

Лабораторная работа №4

дисциплина: Архитектура компьютера

Савенкова Алиса Евгеньевна

Содержание

| | | |
|----------|---|-----------|
| 1 | Цель работы | 5 |
| 2 | Задание | 6 |
| 3 | Теоретическое введение | 7 |
| 4 | Выполнение лабораторной работы | 9 |
| 4.1 | Программа Hello world! | 9 |
| 4.2 | Транслятор NASM | 12 |
| 4.3 | Расширенный синтаксис командной строки NASM | 13 |
| 4.4 | Компоновщик LD | 14 |
| 4.5 | Задание для самостоятельной работы | 17 |
| 5 | Выводы | 22 |
| | Список литературы | 23 |

Список иллюстраций

| | | |
|------|---|----|
| 4.1 | Создание каталога | 9 |
| 4.2 | Каталог для работы с программами на языке ассемблера NASM . . | 10 |
| 4.3 | Создание файла hello.asm | 10 |
| 4.4 | Открытие файла hello.asm с помощью gedit | 11 |
| 4.5 | Файл hello.asm | 11 |
| 4.6 | Ввод текста | 12 |
| 4.7 | Компиляция текста | 12 |
| 4.8 | Наличие файла hello.o в каталоге | 13 |
| 4.9 | Компиляция в obj.o | 13 |
| 4.10 | Наличие файлов в каталоге | 14 |
| 4.11 | Передача файлов компоновщику | 14 |
| 4.12 | Наличие исполняемого файла в каталоге | 15 |
| 4.13 | Передача файлов на обработку компоновщику | 15 |
| 4.14 | Передача файлов на обработку компоновщику со значением main | 16 |
| 4.15 | Командная строка LD | 16 |
| 4.16 | Запуск исполняемого файла | 17 |
| 4.17 | Создание копии файла | 17 |
| 4.18 | Наличие файла lab4.asm в каталоге | 18 |
| 4.19 | Открытие файла lab4.asm с помощью gedit | 18 |
| 4.20 | Внесение изменений в lab4.asm | 19 |
| 4.21 | Внесение изменений в lab4.asm | 19 |
| 4.22 | Компиляция lab4.asm | 20 |
| 4.23 | Передача lab4.asm на обработку компоновщику | 20 |
| 4.24 | Запуск исполняемого файла | 21 |
| 4.25 | Загрузка файлов на Github | 21 |

Список таблиц

1 Цель работы

Целью данной лабораторной работы является освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

1. Программа Hello world!
2. Транслятор NASM
3. Расширенный синтаксис командной строки NASM
4. компоновщик LD
5. Запуск исполняемого файла
6. Задания для самостоятельной работы

3 Теоретическое введение

Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства. 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); • устройство управления (УУ); • регистры. Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. 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). 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. В состав ЭВМ также входят периферийные устройства, которые можно разделить на: • устройства внешней памяти; • устройства ввода-вывода. Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в

языках высокого уровня, таких как C/C++, Perl, Python и пр. Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — машинные коды. Наиболее распространёнными ассемблерами для архитектуры x86 являются:

- для DOS/Windows: Borland Turbo Assembler (TASM), Microsoft Macro Assembler (MASM) и Watcom assembler (WASM);
- для GNU/Linux: gas (GNU Assembler), использующий AT&T-синтаксис, в отличие от большинства других популярных ассемблеров, которые используют Intel-синтаксис.

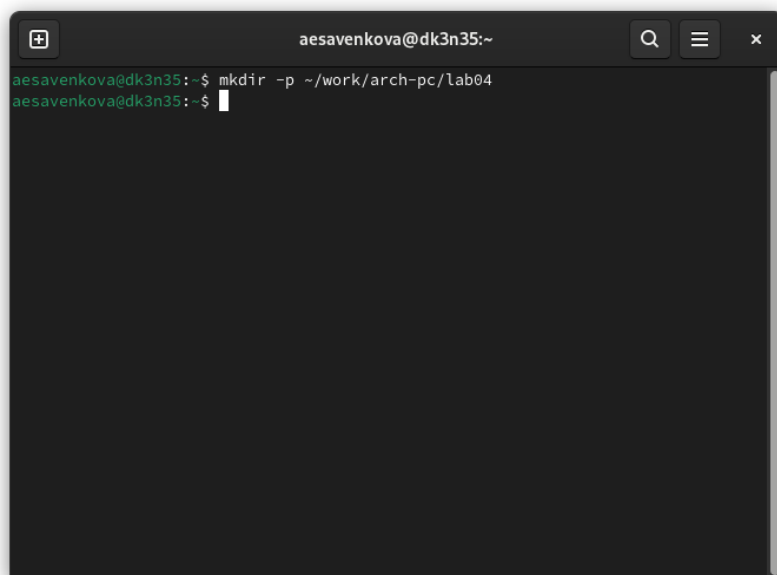
NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64. В процессе создания ассемблерной программы можно выделить четыре шага:

- Набор текста.
- Трансляция.
- Компоновка или линковка.
- Запуск программы.

4 Выполнение лабораторной работы

4.1 Программа Hello world!

Первым действием создаю каталог для работы с программами на языке ассемблера NASM (рис. 4.1).

A screenshot of a terminal window with a dark background. The window title is 'aesavenkova@dk3n35:~'. The terminal shows two lines of text: 'aesavenkova@dk3n35:~\$ mkdir -p ~/work/arch-pc/lab04' and 'aesavenkova@dk3n35:~\$' followed by a cursor. The window has standard Linux window controls (minimize, maximize, close) and a search icon in the top right corner.

```
aesavenkova@dk3n35:~$ mkdir -p ~/work/arch-pc/lab04
aesavenkova@dk3n35:~$
```

Рис. 4.1: Создание каталога

Далее перехожу в созданный каталог (рис. 4.2).

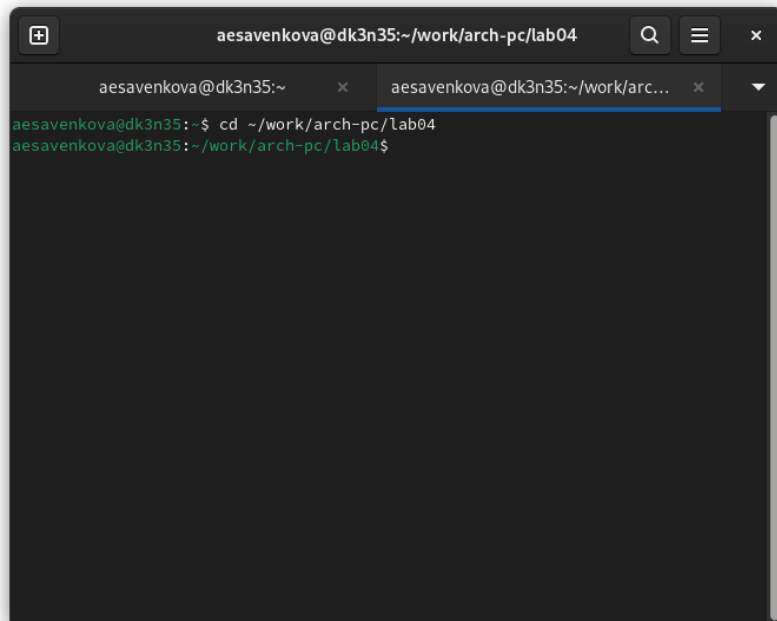


Рис. 4.2: Каталог для работы с программами на языке ассемблера NASM

В этом каталоге создаю текстовый файл с именем hello.asm (рис. 4.3).

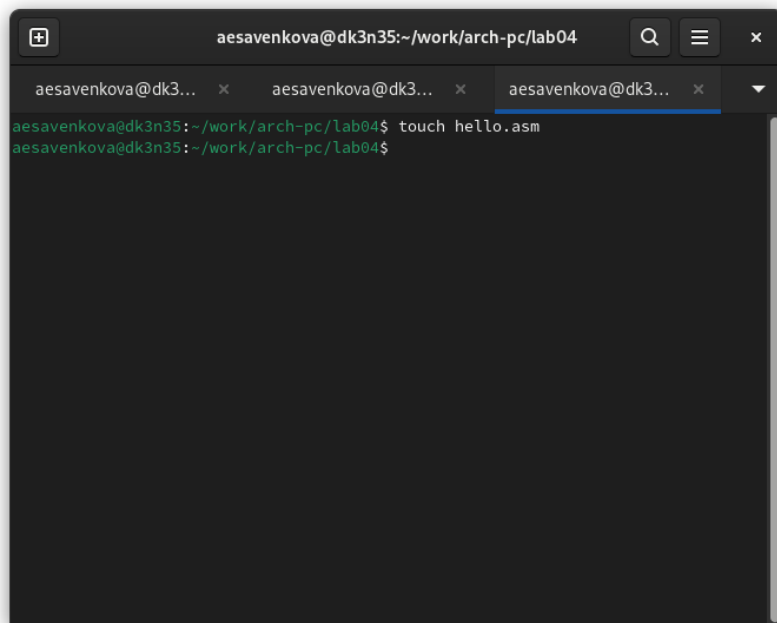


Рис. 4.3: Создание файла hello.asm

Открываю этот файл с помощью текстового редактора gedit (рис. 4.4 - 4.5).

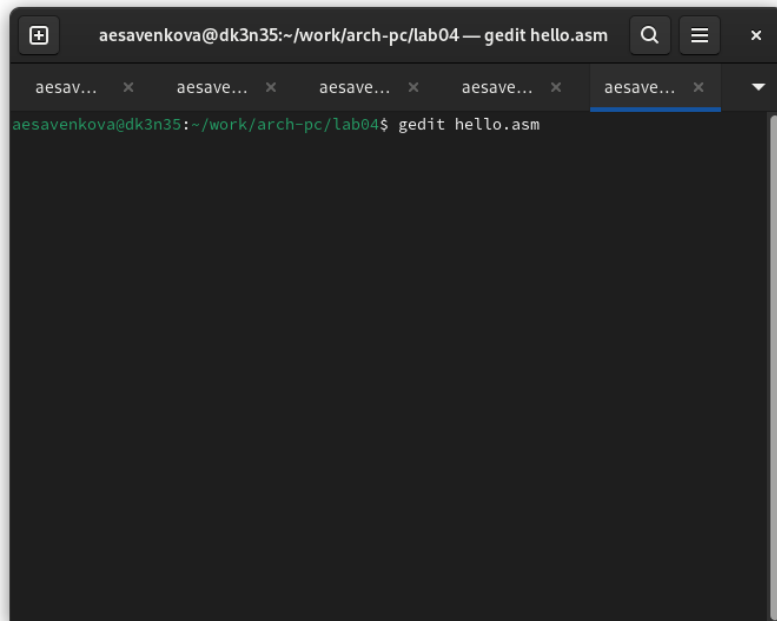


Рис. 4.4: Открытие файла hello.asm с помощью gedit



Рис. 4.5: Файл hello.asm

В открывшемся файле ввожу предложенный текст (рис. 4.6).



Рис. 4.6: Ввод текста

4.2 Транслятор NASM

Для компиляции приведённого выше текста программы «Hello World» ввожу команду `nasm -f elf hello.asm` (ключ `-f` указывает транслятору, что требуется создать бинарные файлы в формате ELF) (рис. 4.7).

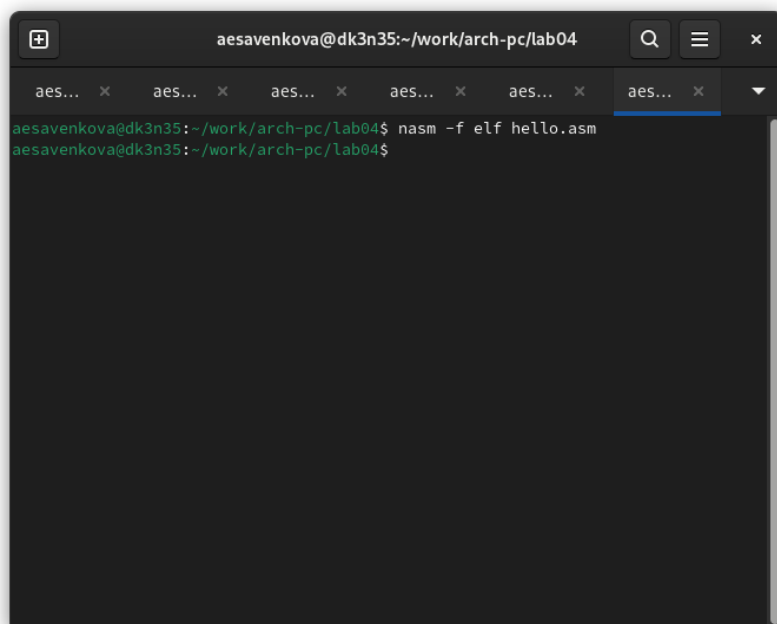
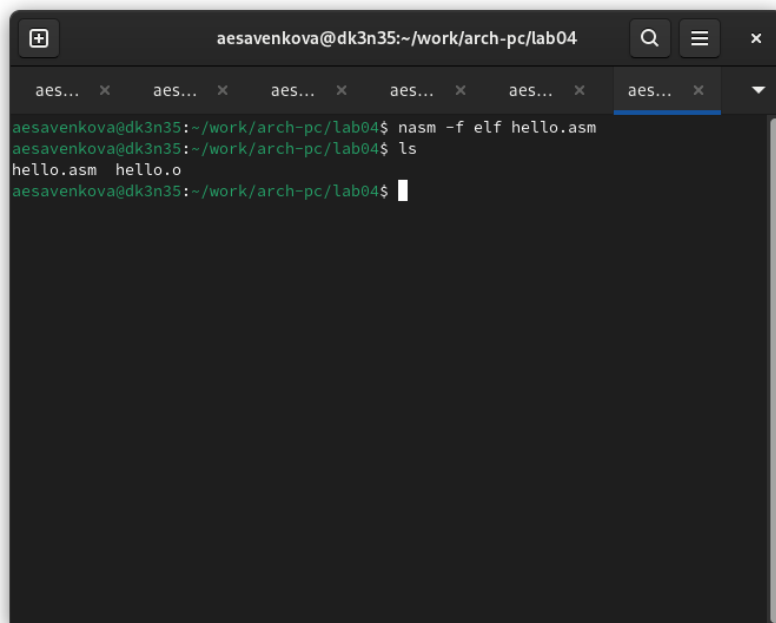


Рис. 4.7: Компиляция текста

С помощью команды `ls` проверяю, что объектный файл под именем `hello.o` был

создан (рис. 4.8).

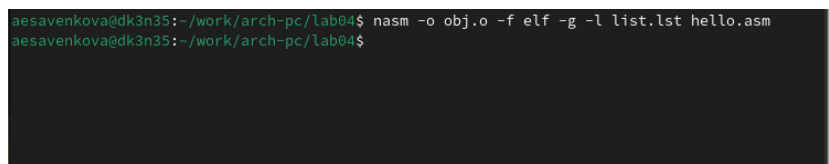


```
aesavenkova@dk3n35:~/work/arch-pc/lab04$ nasm -f elf hello.asm
aesavenkova@dk3n35:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
aesavenkova@dk3n35:~/work/arch-pc/lab04$
```

Рис. 4.8: Наличие файла hello.o в каталоге

4.3 Расширенный синтаксис командной строки NASM

Далее для того, чтобы скомпилировать исходный файл hello.asm в obj.o (опция -o позволяет задать имя объектного файла, в данном случае obj.o), ввожу команду `nasm -o obj.o -f elf -g -l list.lst hello.asm`, при этом формат выходного файла будет elf, и в него будут включены символы для отладки (опция -g), кроме того, будет создан файл листинга list.lst (опция -l) (рис. 4.9).



```
aesavenkova@dk3n35:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
aesavenkova@dk3n35:~/work/arch-pc/lab04$
```

Рис. 4.9: Компиляция в obj.o

Затем с помощью команды `ls` проверяю, что файлы были созданы (рис. 4.10).

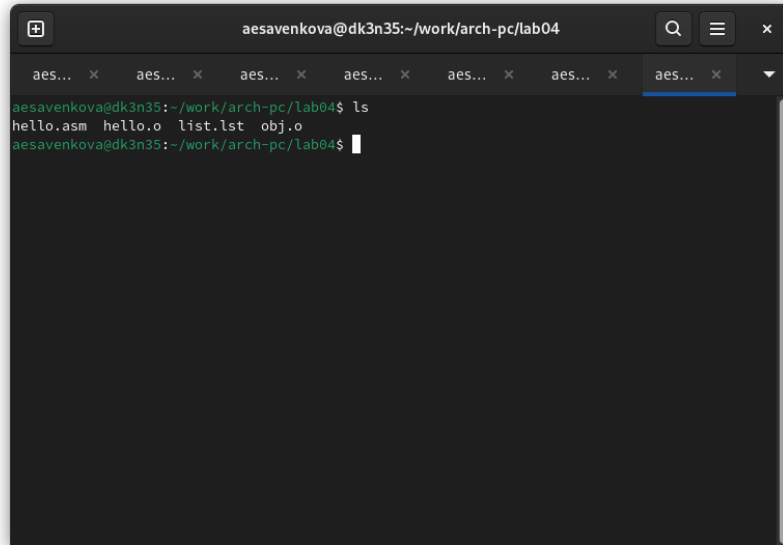


Рис. 4.10: Наличие файлов в каталоге

4.4 Компоновщик LD

Чтобы передать объектный файл на обработку компоновщику ввожу команду `ld -m elf_i386 hello.o -o hello` (рис. 4.11).

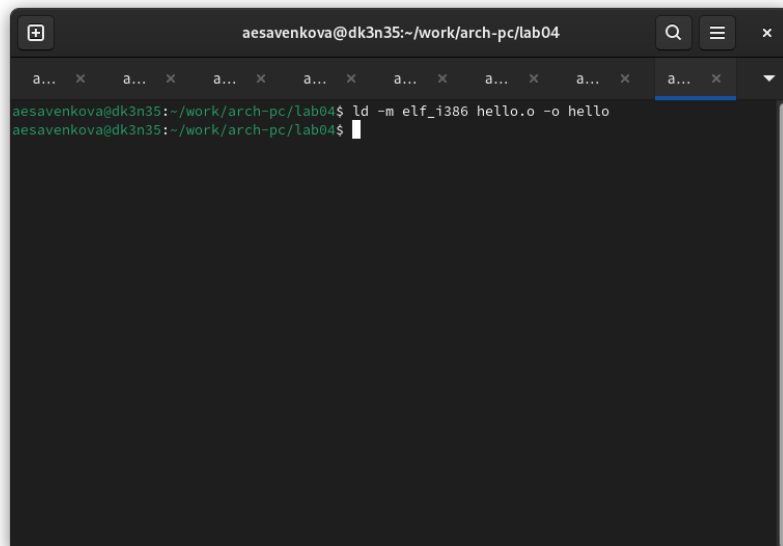
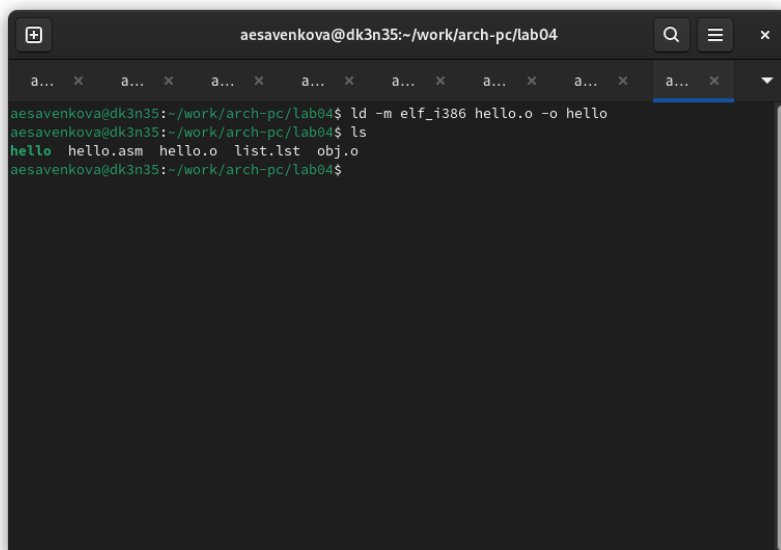


Рис. 4.11: Передача файлов компоновщику

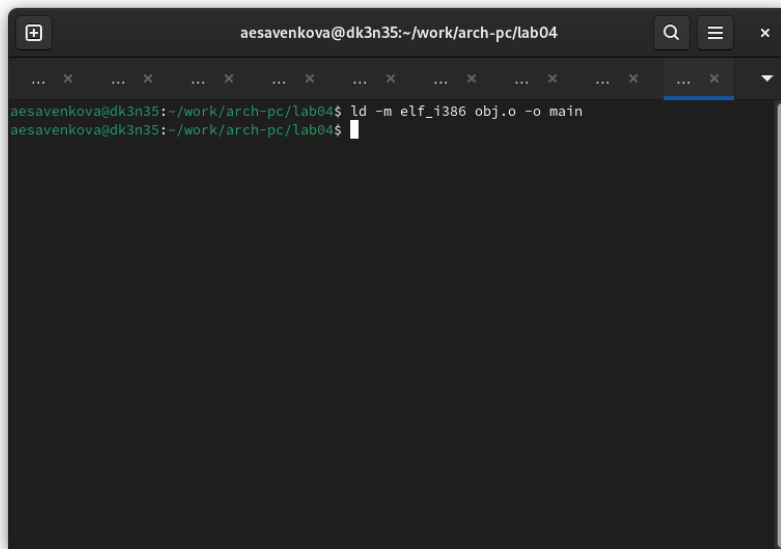
С помощью команды `ls` проверяю, что исполняемый файл `hello` был создан (рис. 4.12).



```
aesavenkova@dk3n35:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
aesavenkova@dk3n35:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o
aesavenkova@dk3n35:~/work/arch-pc/lab04$
```

Рис. 4.12: Наличие исполняемого файла в каталоге

Ввожу следующую команду `ld -m elf_i386 obj.o -o main` (рис. 4.13).

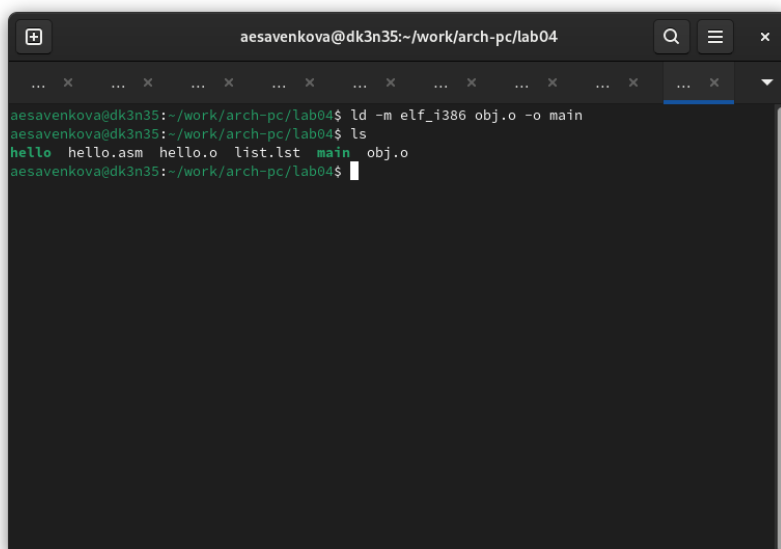


```
aesavenkova@dk3n35:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
aesavenkova@dk3n35:~/work/arch-pc/lab04$
```

Рис. 4.13: Передача файлов на обработку компоновщику

После ключа `-o` было задано значение `main`, поэтому у файла будет имя `main`

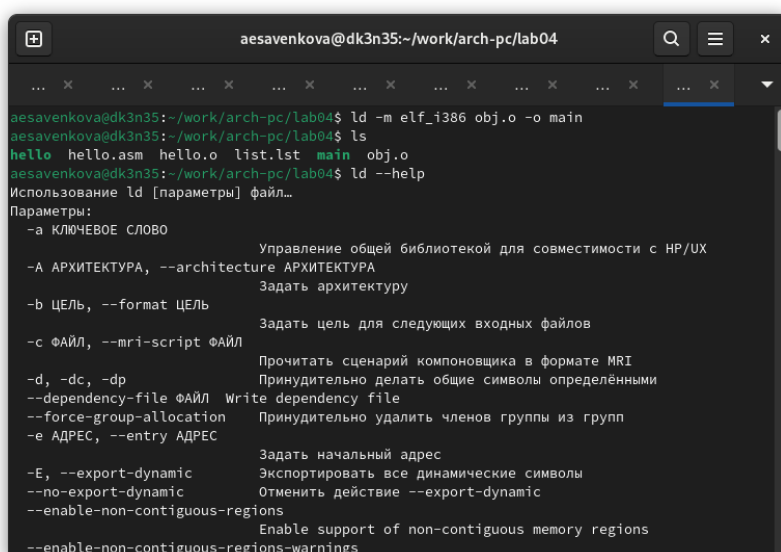
(рис. 4.14).



```
aesavenkova@dk3n35:~/work/arch-pc/lab04
aesavenkova@dk3n35:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
aesavenkova@dk3n35:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst main obj.o
aesavenkova@dk3n35:~/work/arch-pc/lab04$
```

Рис. 4.14: Передача файлов на обработку компоновщику со значением main

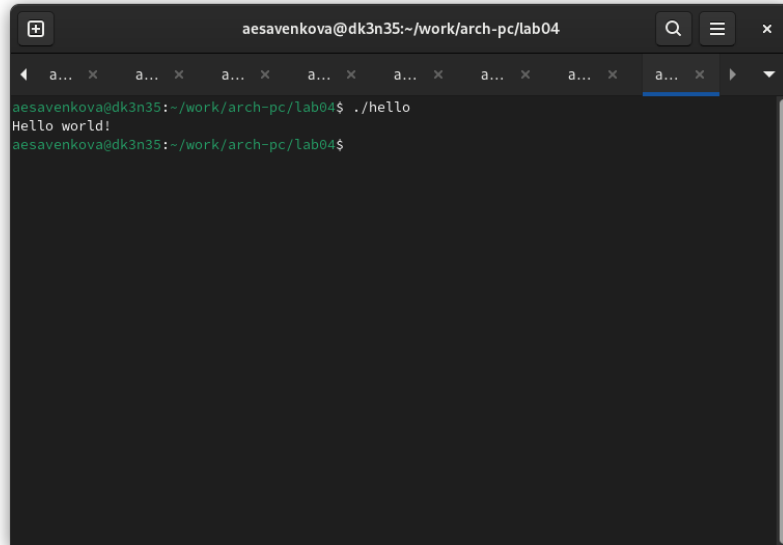
Чтобы увидеть формат командной строки LD, набираю `ld --help` (рис. 4.15).



```
aesavenkova@dk3n35:~/work/arch-pc/lab04
aesavenkova@dk3n35:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
aesavenkova@dk3n35:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst main obj.o
aesavenkova@dk3n35:~/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
```

Рис. 4.15: Командная строка LD

Запускаю на выполнение созданный исполняемый файл, введя `./hello` (рис. 4.16).

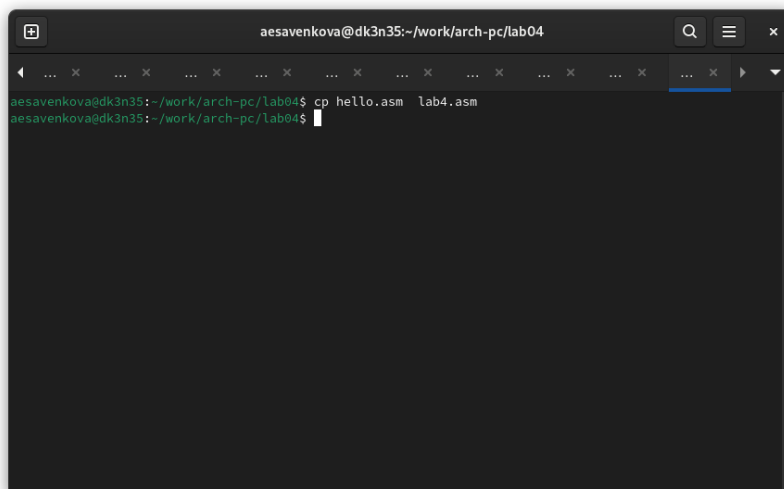
A terminal window with a dark background. The title bar shows the user 'aesavenkova@dk3n35' and the directory '~/work/arch-pc/lab04'. The terminal shows the command './hello' being executed, which outputs 'Hello world!'. The prompt is now 'aesavenkova@dk3n35: ~/work/arch-pc/lab04\$'.

```
aesavenkova@dk3n35:~/work/arch-pc/lab04$ ./hello
Hello world!
aesavenkova@dk3n35:~/work/arch-pc/lab04$
```

Рис. 4.16: Запуск исполняемого файла

4.5 Задание для самостоятельной работы

С помощью команды `cp` создаю копию файла `hello.asm` с именем `lab4.asm` в каталоге `~/work/arch-pc/lab04` (рис. 4.17).

A terminal window with a dark background. The title bar shows the user 'aesavenkova@dk3n35' and the directory '~/work/arch-pc/lab04'. The terminal shows the command 'cp hello.asm lab4.asm' being executed. The prompt is now 'aesavenkova@dk3n35: ~/work/arch-pc/lab04\$'.

```
aesavenkova@dk3n35:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
aesavenkova@dk3n35:~/work/arch-pc/lab04$
```

Рис. 4.17: Создание копии файла

Проверяю наличие файла в каталоге (рис. 4.18).

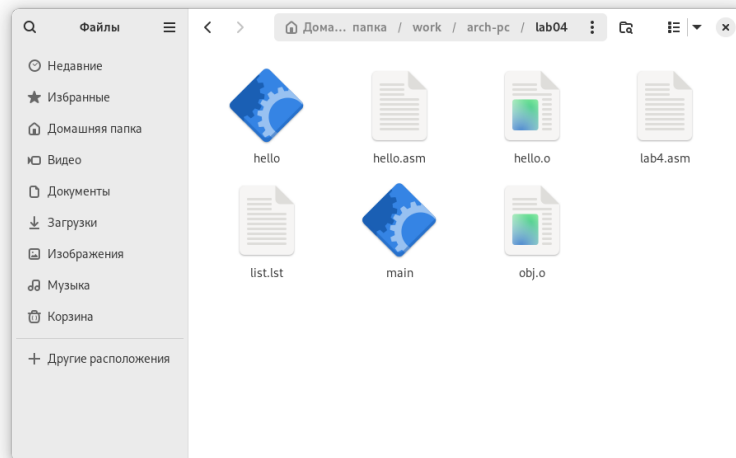


Рис. 4.18: Наличие файла lab4.asm в каталоге

Открываю файл lab4.asm с помощью текстового редактора gedit (рис. 4.19).

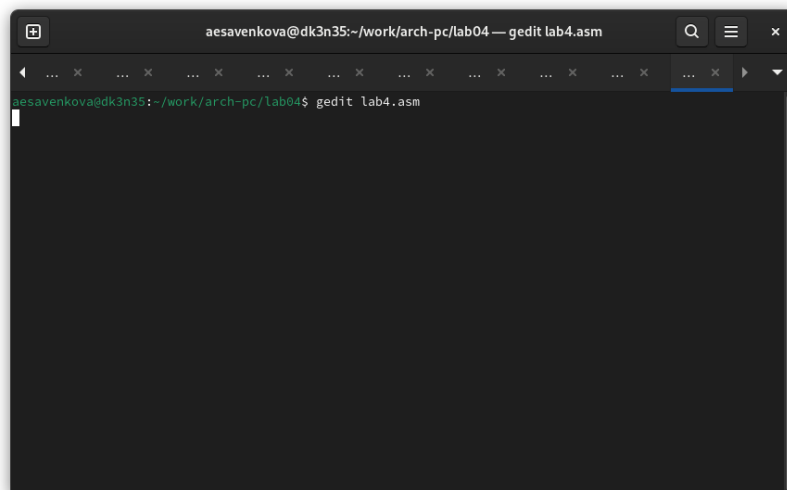


Рис. 4.19: Открытие файла lab4.asm с помощью gedit

Следующим шагом вношу изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с моими фамилией и именем (рис. 4.20).



Рис. 4.20: Внесение изменений в lab4.asm

Затем с помощью команды `nasm -f elf lab4.asm` компилирую текст программы из файла `lab4.asm` в объектный код, который запишется в файл `lab4.o`, и проверяю наличие файла с помощью `ls` (рис. 4.21).

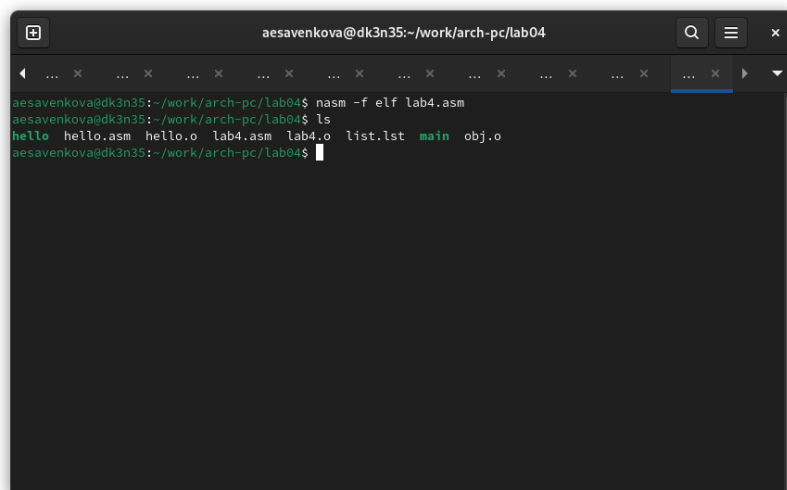
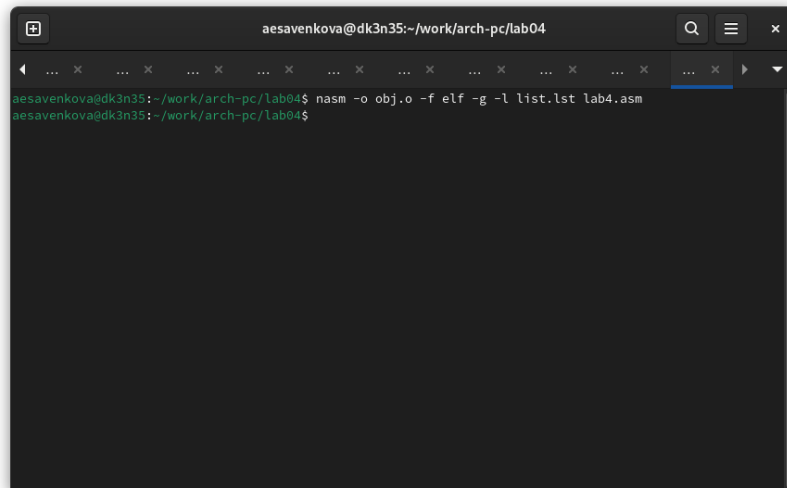


Рис. 4.21: Внесение изменений в lab4.asm

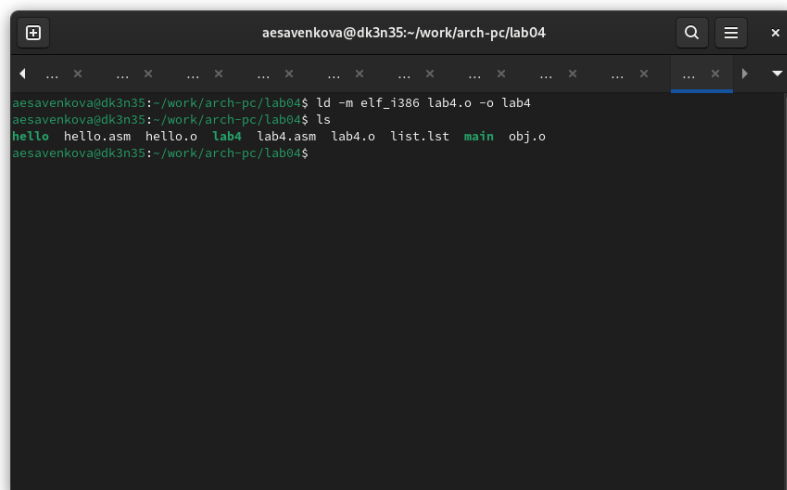
Далее ввожу команду `nasm -o obj.o -f elf -g -l list.lst lab4.asm`, чтобы скомпилировать исходный файл `lab4.asm` в `obj.o` (рис. 4.22).



```
aesavenkova@dk3n35:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst lab4.asm
aesavenkova@dk3n35:~/work/arch-pc/lab04$
```

Рис. 4.22: Компиляция lab4.asm

Заключительным шагом передаю объектный файл на обработку компоновщику (рис. 4.23).



```
aesavenkova@dk3n35:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
aesavenkova@dk3n35:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o
aesavenkova@dk3n35:~/work/arch-pc/lab04$
```

Рис. 4.23: Передача lab4.asm на обработку компоновщику

Запускаю исполняемый файл и удостоверяюсь, что на экран действительно выводятся мои фамилия и имя (рис. 4.24).

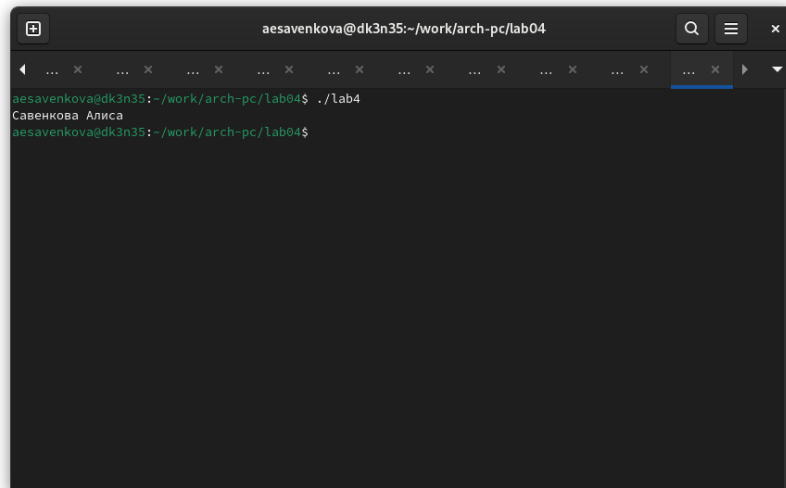
A terminal window with a dark background. The title bar shows the user 'aesavenkova@dk3n35' and the current directory '~/work/arch-pc/lab04'. The terminal shows the command './lab4' being executed, followed by the output 'Савенкова Алиса' and the prompt 'aesavenkova@dk3n35:~/work/arch-pc/lab04\$'.

Рис. 4.24: Запуск исполняемого файла

Копирую файлы hello.asm и lab4.asm в свой локальный репозиторий в каталог ~/work/study/2024-2025/“Архитектура компьютера”/arch-pc/labs/lab04/. Загружаю файлы на Github (рис. 4.25).

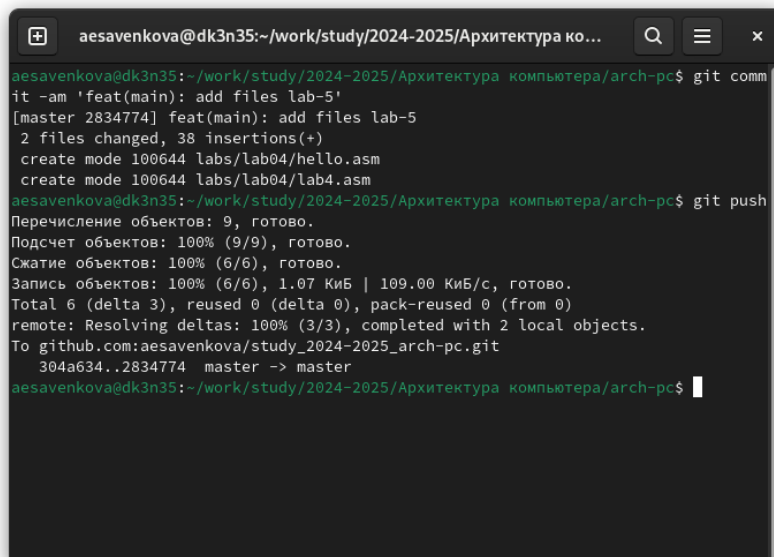
A terminal window with a dark background. The title bar shows the user 'aesavenkova@dk3n35' and the current directory '~/work/study/2024-2025/Архитектура ко...'. The terminal shows the following commands and output:
1. 'git commit -m "feat(main): add files lab-5"'
2. Output: '[master 2834774] feat(main): add files lab-5', '2 files changed, 38 insertions(+)', 'create mode 100644 labs/lab04/hello.asm', 'create mode 100644 labs/lab04/lab4.asm'
3. 'git push'
4. Output: 'Перечисление объектов: 9, готово.', 'Подсчет объектов: 100% (9/9), готово.', 'Сжатие объектов: 100% (6/6), готово.', 'Запись объектов: 100% (6/6), 1.07 КиБ | 109.00 КиБ/с, готово.', 'Total 6 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)', 'remote: Resolving deltas: 100% (3/3), completed with 2 local objects.', 'To github.com:aesavenkova/study_2024-2025_arch-pc.git', '304a634..2834774 master -> master'
The prompt 'aesavenkova@dk3n35:~/work/study/2024-2025/Архитектура компьютера/arch-pc\$' is visible at the bottom.

Рис. 4.25: Загрузка файлов на Github

5 Выводы

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

Список литературы

1. <https://esystem.rudn.ru/mod/resource/view.php?id=1030832>