

Отчёт по лабораторной работе №4

Дисциплина: архитектура компьютера

Барбакова Алиса Саяновна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	9
4.1	Создание программы Hello world!	9
4.2	Работа с транслятором NASM	11
4.3	Работа с расширенным синтаксисом командной строки NASM	12
4.4	Работа с компоновщиком LD	13
4.5	Запуск исполняемого файла	15
4.6	Выполнение заданий для самостоятельной работы.	16
5	Выводы	24
	Список литературы	25

Список иллюстраций

4.1	Создание каталога	9
4.2	Создание пустого файла	10
4.3	Заполнение файла	11
4.4	Превращение в объектный код	12
4.5	Компиляция текста программы	13
4.6	Обработка компоновщика	14
4.7	Передача объектного файла на обработку компоновщику .	15
4.8	Запуск исполняемого файла	16
4.9	Создание копии файла	17
4.10	Изменение программы	17
4.11	Компиляция текста программы	18
4.12	Передача объектного файла на обработку компоновщику .	19
4.13	Запуск исполняемого файла	20
4.14	Создании копии файлов в новом каталоге	21
4.15	Добавление файлов на GitHub	21
4.16	Добавление файлов на GitHub	22
4.17	Отправка файлов	23

1 Цель работы

Цель данной лабораторной работы - освоить 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. 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 реги-

страм осуществляется 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.

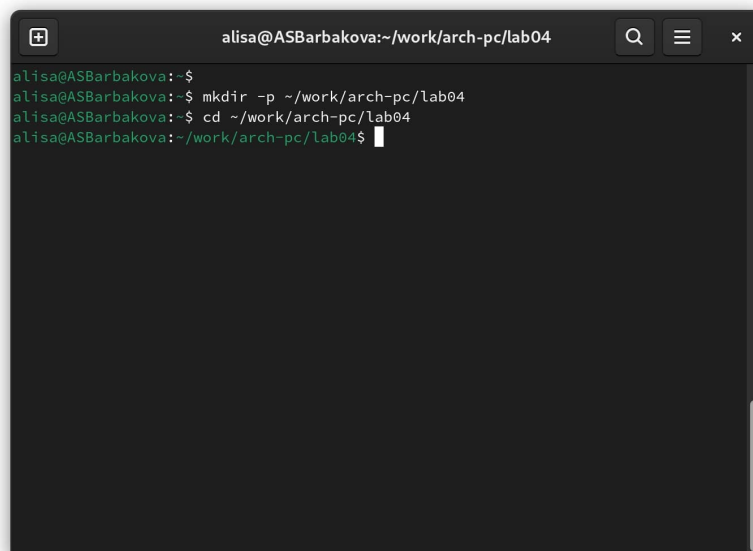
При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. Он заключается в следующем: 1. формирование адреса в памяти очередной команды; 2. считывание кода команды из памяти и её дешифрация; 3. выполнение команды; 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 Создание программы Hello world!

Открываю терминал. Создаю каталог, в котором буду работать, и перехожу туда с помощью `cd` (рис. 4.1).



```
alisa@ASBarbakova:~/work/arch-pc/lab04
alisa@ASBarbakova:~$
alisa@ASBarbakova:~$ mkdir -p ~/work/arch-pc/lab04
alisa@ASBarbakova:~$ cd ~/work/arch-pc/lab04
alisa@ASBarbakova:~/work/arch-pc/lab04$
```

Рис. 4.1: Создание каталога

Создаю в текущем каталоге файл `hello.asm` с помощью утилиты `touch` и открываю его редактором `gedit` (рис. 4.2).

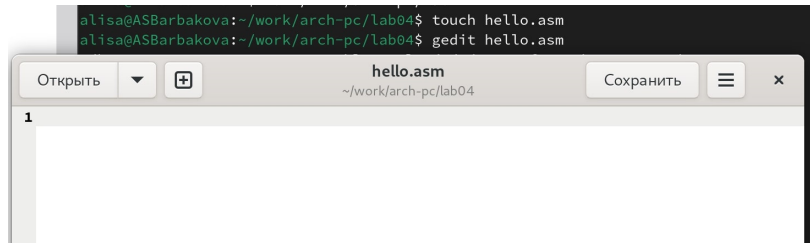
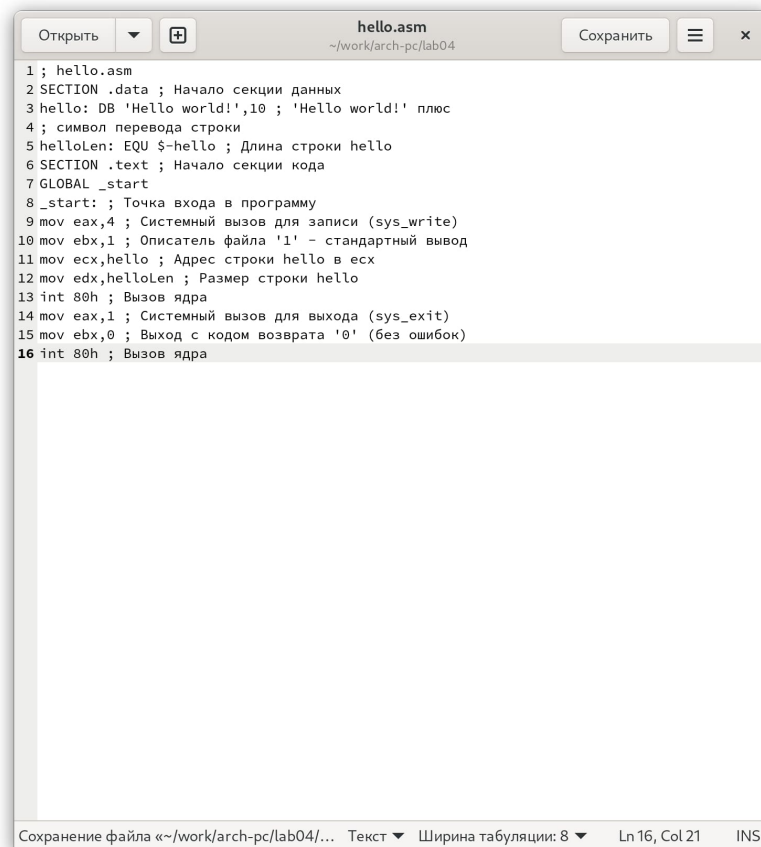


Рис. 4.2: Создание пустого файла

Вставляю в файл программу для вывода “Hello word!” (рис. 4.3).

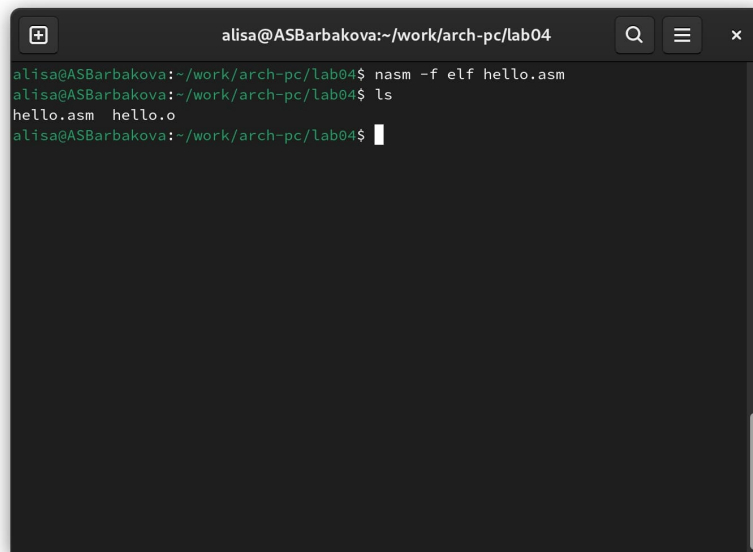


```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для записи (sys_write)
10 mov ebx,1 ; Описатель файла '1' - стандартный вывод
11 mov ecx,hello ; Адрес строки hello в ecx
12 mov edx,helloLen ; Размер строки hello
13 int 80h ; Вызов ядра
14 mov eax,1 ; Системный вызов для выхода (sys_exit)
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16 int 80h ; Вызов ядра
```

Рис. 4.3: Заполнение файла

4.2 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` (рис. 4.4). Далее проверяю правильность выполнения команды с помощью `ls`: файл “hello.o” теперь создан.

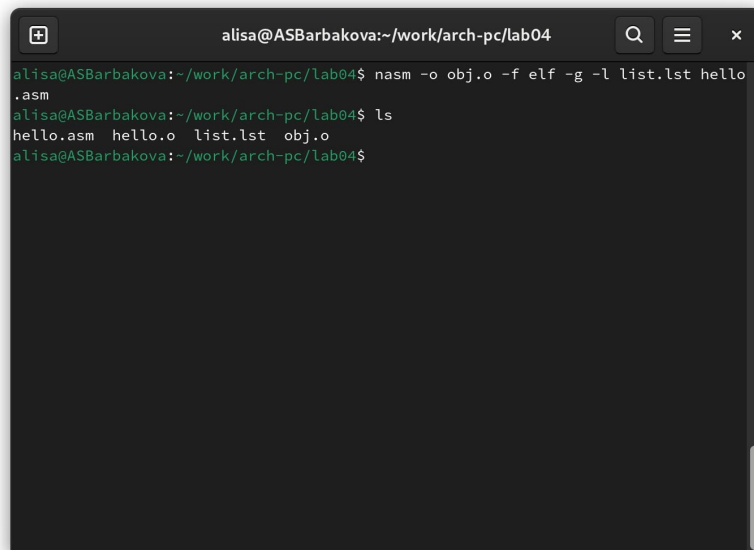
A terminal window with a dark background. The title bar shows 'alisa@ASBarbakova:~/work/arch-pc/lab04'. The terminal content shows the following commands and output:

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

Рис. 4.4: Превращение в объектный код

4.3 Работа с расширенным синтаксисом командной строки NASM

Ввожу команду, которая скомпилирует файл `hello.asm` в файл `obj.o`, при этом формат выходного файла будет `elf`, и в него будут включены символы для отладки (опция `-g`). Кроме того, с помощью ключа `-l` будет создан файл листинга `list.lst` (рис. 4.5). Проверяю всё утилитой `ls`.

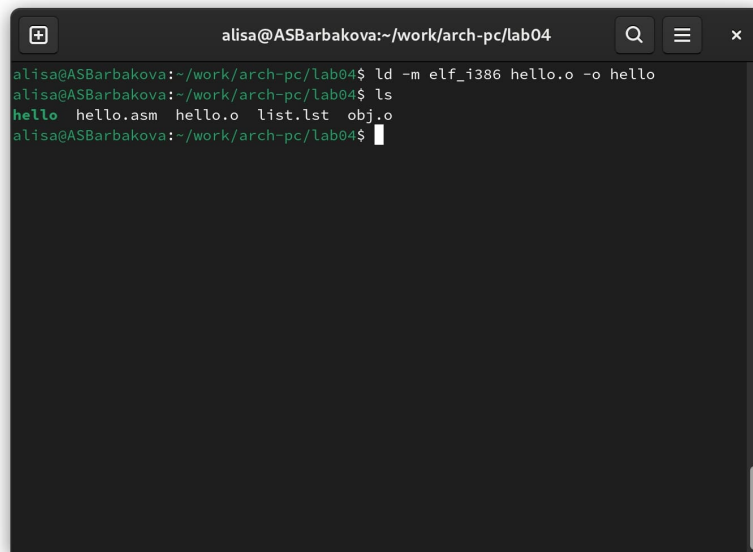
A terminal window with a dark background and light green text. The window title is 'alisa@ASBarbakova:~/work/arch-pc/lab04'. The terminal shows the following commands and output:

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

Рис. 4.5: Компиляция текста программы

4.4 Работа с компоновщиком LD

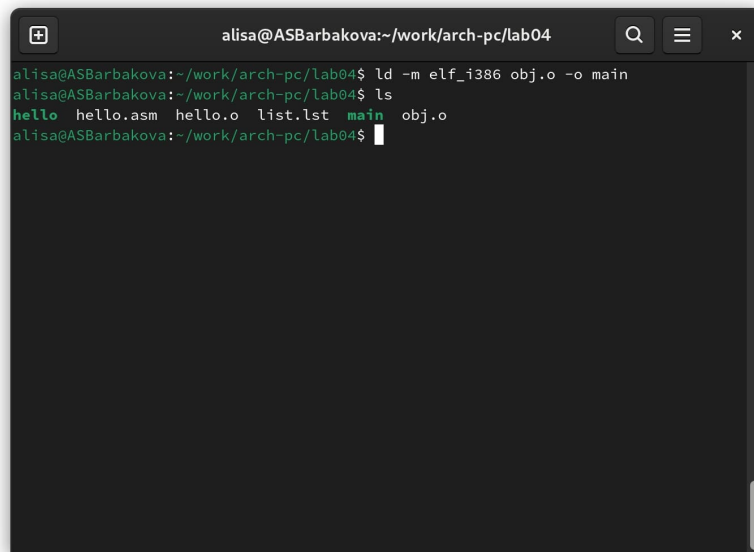
Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемую программу (рис. 4.6). Ключ -o задает имя создаваемого исполняемого файла. Проверяю правильность выполнения команды с помощью утилиты ls.

A terminal window with a dark background. The title bar shows the user 'alisa@ASBarbakova' and the directory '~/work/arch-pc/lab04'. The terminal contains the following text:

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

Рис. 4.6: Обработка компоновщика

Выполняю следующую команду (рис. 4.7). Исполняемый файл будет иметь имя `main`. Объектный файл, из которого собран этот исполняемый файл, имеет имя `obj.o`

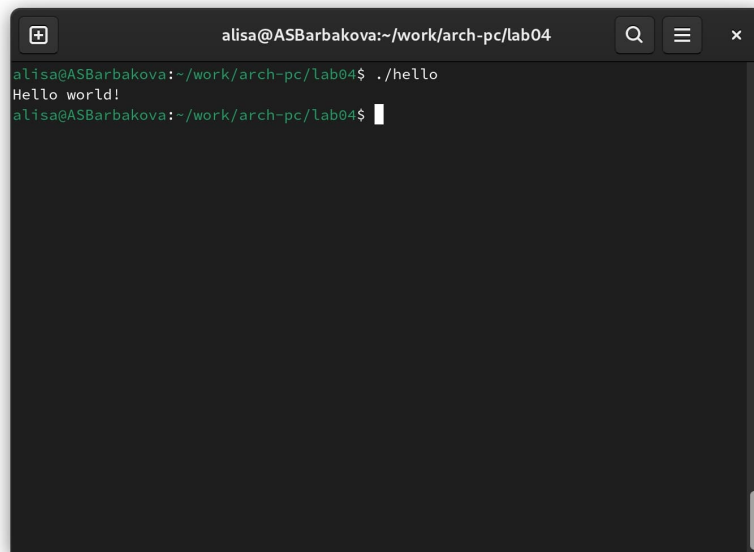
A terminal window with a dark background. The title bar shows 'alisa@ASBarbakova:~/work/arch-pc/lab04'. The terminal contains the following text:

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

Рис. 4.7: Передача объектного файла на обработку компоновщику

4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello командой `./hello`(рис. 4.8).

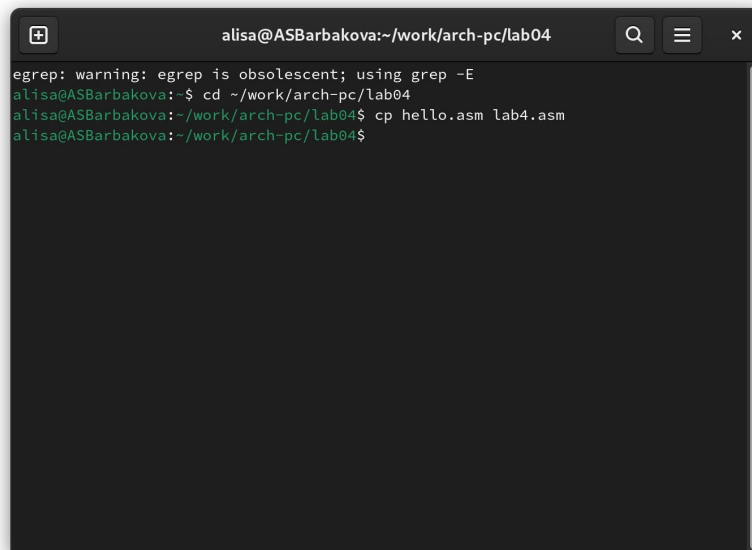
A terminal window with a dark background. The title bar shows 'alisa@ASBarbakova:~/work/arch-pc/lab04'. The prompt is 'alisa@ASBarbakova:~/work/arch-pc/lab04\$'. The command './hello' has been entered, and the output 'Hello world!' is displayed on the next line. The prompt is now 'alisa@ASBarbakova:~/work/arch-pc/lab04\$' with a cursor.

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

Рис. 4.8: Запуск исполняемого файла

4.6 Выполнение заданий для самостоятельной работы.

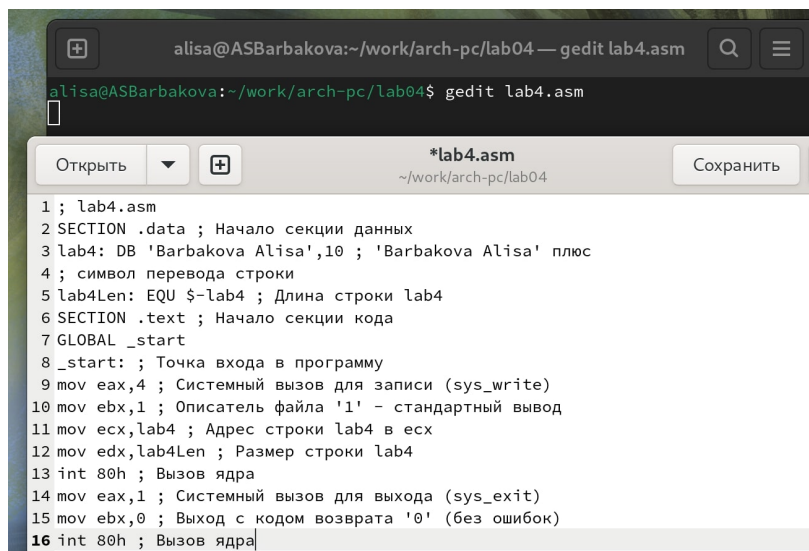
В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создаю копию файла `hello.asm` с именем `lab4.asm` (рис. 4.9).

A terminal window with a dark background. The title bar shows 'alisa@ASBarbakova:~/work/arch-pc/lab04'. The terminal contains the following text: 'egrep: warning: egrep is obsolescent; using grep -E', 'alisa@ASBarbakova:~\$ cd ~/work/arch-pc/lab04', 'alisa@ASBarbakova:~/work/arch-pc/lab04\$ cp hello.asm lab4.asm', and 'alisa@ASBarbakova:~/work/arch-pc/lab04\$'.

```
egrep: warning: egrep is obsolescent; using grep -E
alisa@ASBarbakova:~$ cd ~/work/arch-pc/lab04
alisa@ASBarbakova:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
alisa@ASBarbakova:~/work/arch-pc/lab04$
```

Рис. 4.9: Создание копии файла

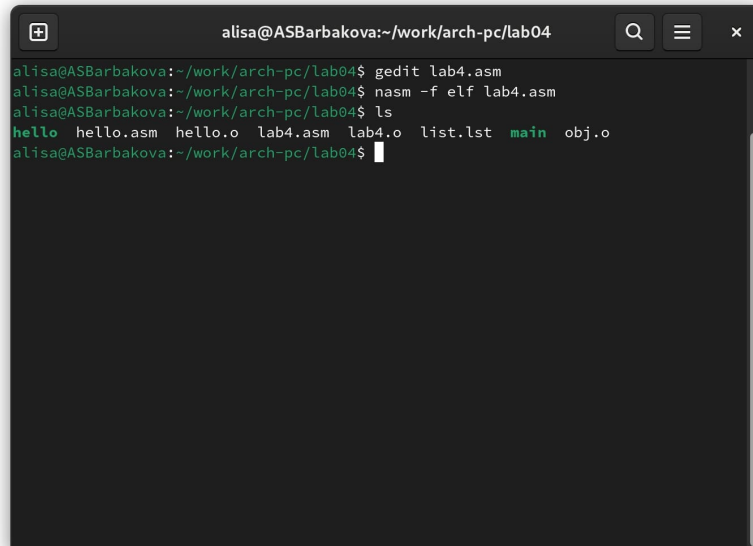
С помощью gedit открываю файл lab4.asm и вношу изменения в программу так, чтобы она выводила мои фамилию и имя. (рис. 4.10).

A screenshot of the gedit text editor. The title bar shows 'alisa@ASBarbakova:~/work/arch-pc/lab04 — gedit lab4.asm'. The editor window has a menu bar with 'Открыть', a dropdown arrow, a '+' icon, and 'Сохранить'. The file name '*lab4.asm' and path '~/work/arch-pc/lab04' are shown. The terminal at the bottom shows 'alisa@ASBarbakova:~/work/arch-pc/lab04\$ gedit lab4.asm'. The main text area contains assembly code for lab4.asm.

```
1 ; lab4.asm
2 SECTION .data ; Начало секции данных
3 lab4: DB 'Barbakova Alisa',10 ; 'Barbakova Alisa' плюс
4 ; символ перевода строки
5 lab4Len: EQU $-lab4 ; Длина строки lab4
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для записи (sys_write)
10 mov ebx,1 ; Описатель файла '1' - стандартный вывод
11 mov ecx,lab4 ; Адрес строки lab4 в ecx
12 mov edx,lab4Len ; Размер строки lab4
13 int 80h ; Вызов ядра
14 mov eax,1 ; Системный вызов для выхода (sys_exit)
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16 int 80h ; Вызов ядра
```

Рис. 4.10: Изменение программы

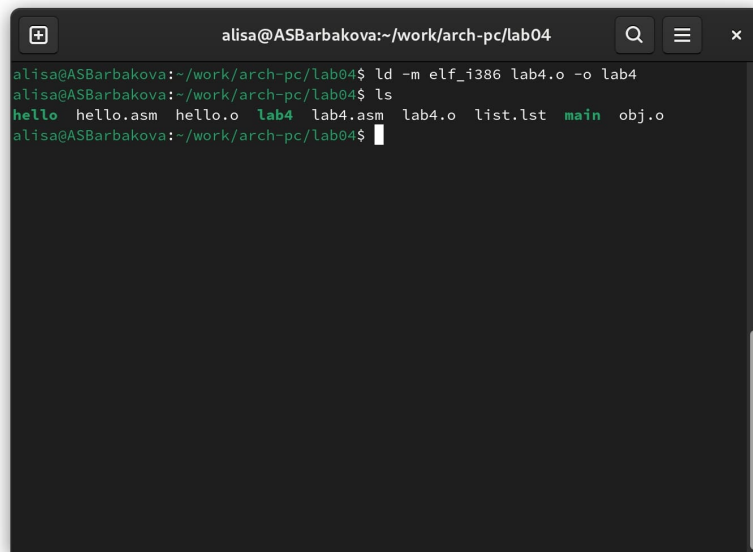
Компилирую текст программы в объектный файл (рис. 4.11). Проверяю с помощью `ls`, что файл `lab4.o` создан.



```
alisa@ASBarbakova:~/work/arch-pc/lab04$ gedit lab4.asm
alisa@ASBarbakova:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
alisa@ASBarbakova:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o
alisa@ASBarbakova:~/work/arch-pc/lab04$
```

Рис. 4.11: Компиляция текста программы

Передаю объектный файл `lab4.o` на обработку компоновщику LD, чтобы получить исполняемый файл `lab4` (рис. 4.12).

A terminal window with a dark background and light green text. The window title is 'alisa@ASBarbakova:~/work/arch-pc/lab04'. The terminal shows the following commands and output:

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

Рис. 4.12: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4 и вижу, что на экран выводятся мои фамилия и имя (рис. 4.13).

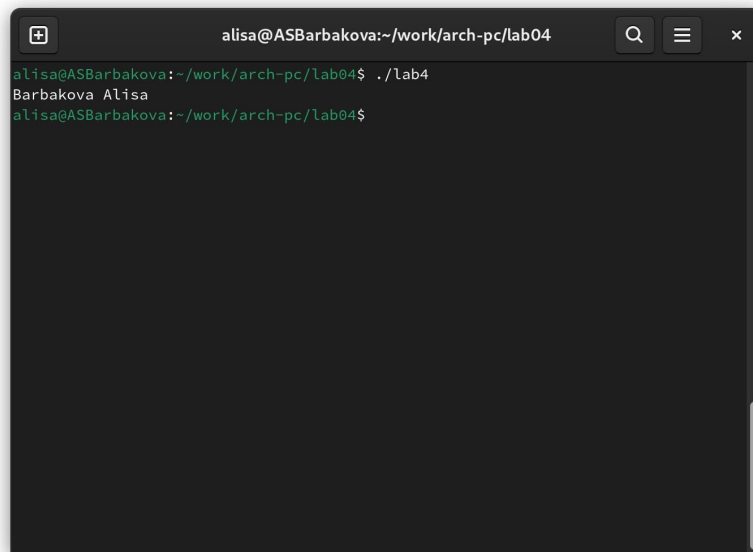
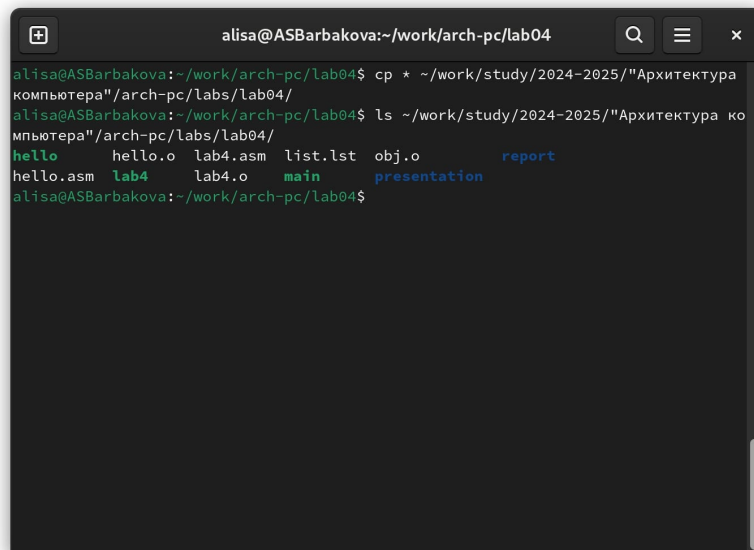
A terminal window with a dark background. The title bar at the top shows 'alisa@ASBarbakova:~/work/arch-pc/lab04' and standard window controls. The terminal content shows a prompt 'alisa@ASBarbakova:~/work/arch-pc/lab04\$' followed by the command './lab4'. The next line shows the output 'Barbakova Alisa', and the prompt returns to 'alisa@ASBarbakova:~/work/arch-pc/lab04\$'.

Рис. 4.13: Запуск исполняемого файла

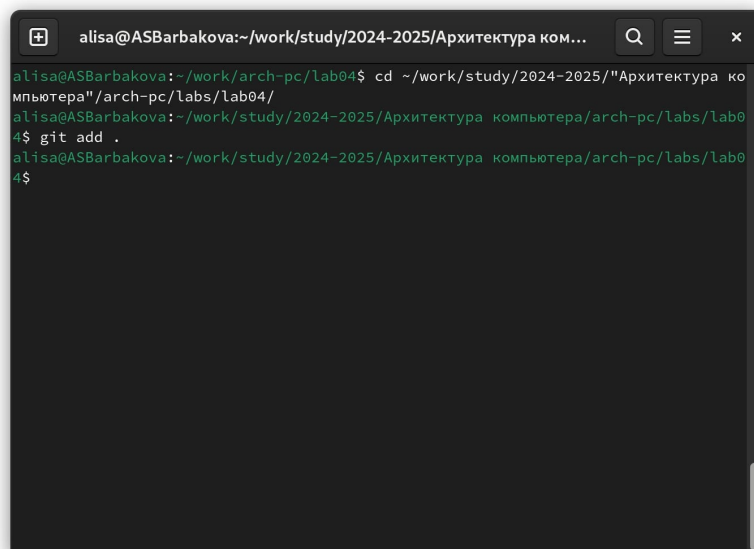
Копирую файлы `hello.asm` и `lab4.asm` в мой локальный репозиторий в каталог `~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/` командой `ср*`. (рис. 4.14). Проверяю командой `ls` правильность выполнения команды.



```
alisa@ASBarbakova:~/work/arch-pc/lab04$ cp * ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04/
alisa@ASBarbakova:~/work/arch-pc/lab04$ ls ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04/
hello.o  lab4.asm  list.lst  obj.o      report
hello.asm lab4      lab4.o    main      presentation
```

Рис. 4.14: Создании копии файлов в новом каталоге

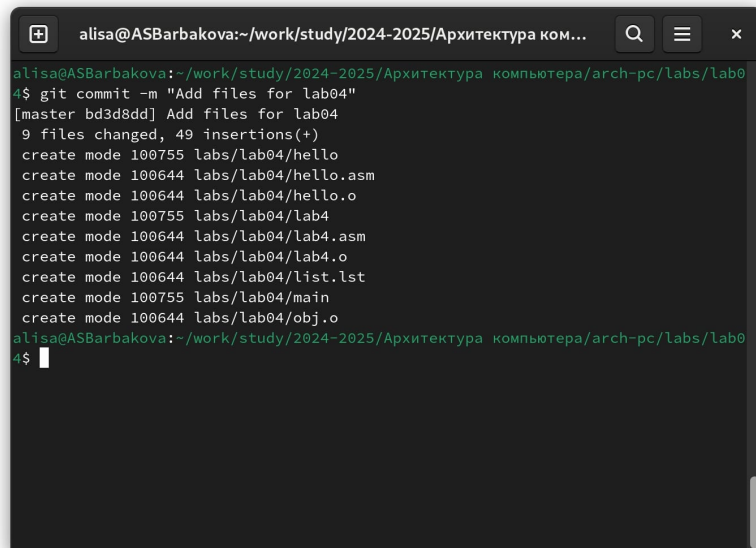
Перехожу в нужный каталог и использую команду `git add .` (рис. 4.15).



```
alisa@ASBarbakova:~/work/study/2024-2025/Архитектура ком...$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04/
alisa@ASBarbakova:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git add .
alisa@ASBarbakova:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$
```

Рис. 4.15: Добавление файлов на GitHub

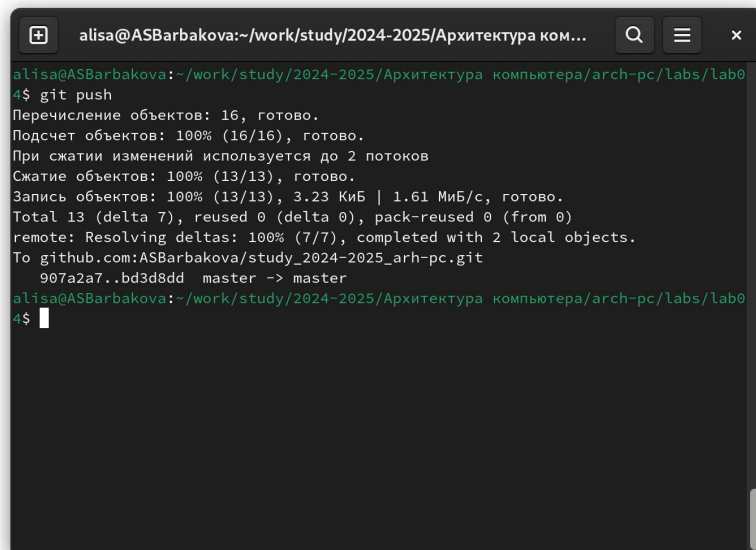
Далее с помощью `git commit` добавляю файлы на GitHub, комментируя действие как “Add files for lab4” (рис. 4.16).



```
alisa@ASBarbakova:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git commit -m "Add files for lab04"
[master bd3d8dd] Add files for lab04
9 files changed, 49 insertions(+)
create mode 100755 labs/lab04/hello
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/hello.o
create mode 100755 labs/lab04/lab4
create mode 100644 labs/lab04/lab4.asm
create mode 100644 labs/lab04/lab4.o
create mode 100644 labs/lab04/list.lst
create mode 100755 labs/lab04/main
create mode 100644 labs/lab04/obj.o
alisa@ASBarbakova:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$
```

Рис. 4.16: Добавление файлов на GitHub

Отправляю файлы на сервер с помощью команды `git push` (рис. 4.17).



```
alisa@ASBarbakova:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ git push
Перечисление объектов: 16, готово.
Подсчет объектов: 100% (16/16), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (13/13), готово.
Запись объектов: 100% (13/13), 3.23 КиБ | 1.61 МиБ/с, готово.
Total 13 (delta 7), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (7/7), completed with 2 local objects.
To github.com:ASBarbakova/study_2024-2025_arh-pc.git
   907a2a7..bd3d8dd  master -> master
alisa@ASBarbakova:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$
```

Рис. 4.17: Отправка файлов

5 Выводы

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

Список литературы

1. Архитектура ЭВМ Л04
2. Архитектура ЭВМ Л03