

# **Отчёт по лабораторной работе №4**

**Дисциплина: Архитектура компьютера**

Кириянова Екатерина Андреевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Выполнение программы Hello world! . . . . .	8
4.2	Использование транслятора NASM . . . . .	9
4.3	Использование расширенного синтаксиса командной строки NASM	9
4.4	Использование компоновщика LD . . . . .	10
4.5	Запуск исполняемого файла . . . . .	10
4.6	Задание для самостоятельной работы . . . . .	10
<b>5</b>	<b>Выводы</b>	<b>14</b>
	<b>Список литературы</b>	<b>15</b>

# Список иллюстраций

4.1	Создание каталога . . . . .	8
4.2	Текстовый файл hello.asm . . . . .	8
4.3	Заполнение файла . . . . .	9
4.4	Компиляция программы . . . . .	9
4.5	Компиляция в obj.o . . . . .	9
4.6	Обработка компоновщиком . . . . .	10
4.7	Обозначение имени . . . . .	10
4.8	Запуск программы . . . . .	10
4.9	Создание копии файла . . . . .	11
4.10	Внесение изменений . . . . .	11
4.11	Компиляция текста программы . . . . .	11
4.12	Обработка файла . . . . .	11
4.13	Запуск программы . . . . .	12
4.14	Заполнение файла . . . . .	12
4.15	Добавление файлов . . . . .	12
4.16	Отправление файлов на сервер . . . . .	12
4.17	Проверка . . . . .	13

# 1 Цель работы

Освоить процедуру компиляции и сборки программ, написанных на ассемблере NASM.

## 2 Задание

1. Выполнить программу “Hello world!”
2. Использовать транслятор NASM
3. Использовать расширенный синтаксис командной строки NASM
4. Использовать компоновщик LD
5. Запустить исполняемый файл
6. Выполнить задание для самостоятельной работы

### 3 Теоретическое введение

Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства (рис. 4.1). 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 (системной) плате. 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) — 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, пре-

образование (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 (половинки 16-битных регистров). Например, AH (high AX) — старшие 8 бит регистра AX, AL (low AX) — младшие 8 бит регистра AX. 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. 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.

## 4 Выполнение лабораторной работы

### 4.1 Выполнение программы Hello world!

Для работы с программами на языке ассемблера создаю новый каталог и перехожу в него (рис.4.1).

```
eakirjyanova@user:~$ mkdir -p ~/work/arch-pc/lab04  
eakirjyanova@user:~$ cd ~/work/arch-pc/lab04  
eakirjyanova@user:~/work/arch-pc/lab04$
```

Рис. 4.1: Создание каталога

Создаю текстовый файл с названием hello.asm и открываю его с помощью текстового редактора gedit (рис.4.2).

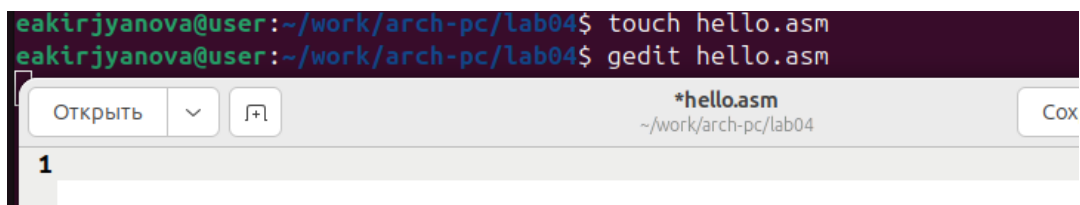
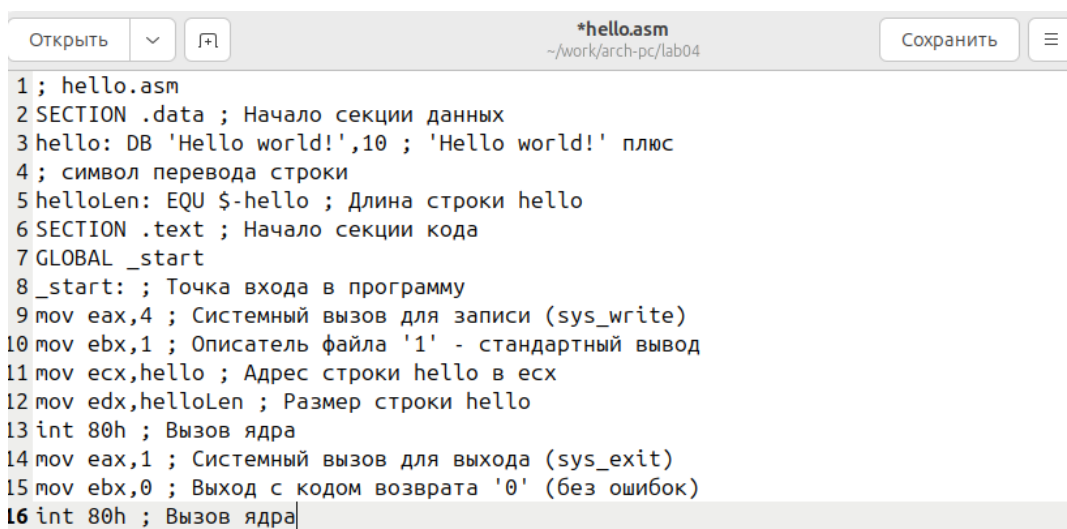


Рис. 4.2: Текстовый файл hello.asm

Ввожу в созданный файл текст программы для вывода Hello world! (рис.4.3).



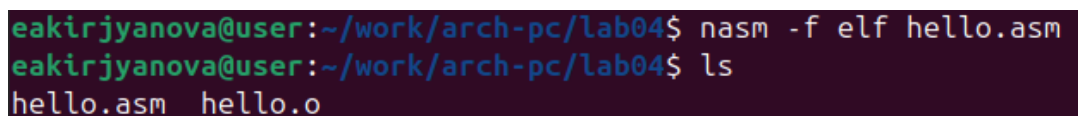


```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для 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 Использование транслятора NASM

С помощью NASM превращаю текст программы в объектный код и проверяю успешность выполнения (рис.4.4).

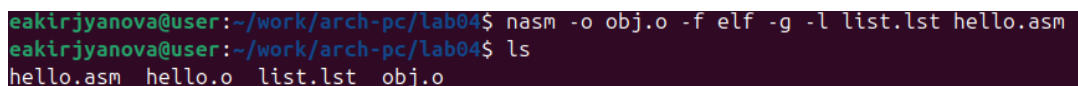


```
eakirjyanova@user:~/work/arch-pc/lab04$ nasm -f elf hello.asm
eakirjyanova@user:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
```

Рис. 4.4: Компиляция программы

## 4.3 Использование расширенного синтаксиса командной строки NASM

Использую команду, которая скомпилирует исходный файл в obj.o, и проверяю успешность выполнения (рис.4.5).



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

Рис. 4.5: Компиляция в obj.o

## 4.4 Использование компоновщика LD

Передаю объектный файл на обработку компоновщику и проверяю успешность выполнения (рис.4.6).

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

Рис. 4.6: Обработка компоновщиком

Задаю исполняемому файлу имя main, так как после ключа -o идет значение main. Объектный файл, из которого собран исполняемый файл, имеет имя obj.o (рис.4.7).

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

Рис. 4.7: Обозначение имени

## 4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл (рис.4.8).

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

Рис. 4.8: Запуск программы

## 4.6 Задание для самостоятельной работы

Создаю копию файла hello.asm с именем lab4.asm и открываю ее с помощью текстового редактора gedit (рис.4.9).

```
eakirjyanova@user:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
eakirjyanova@user:~/work/arch-pc/lab04$ gedit lab4.asm
```

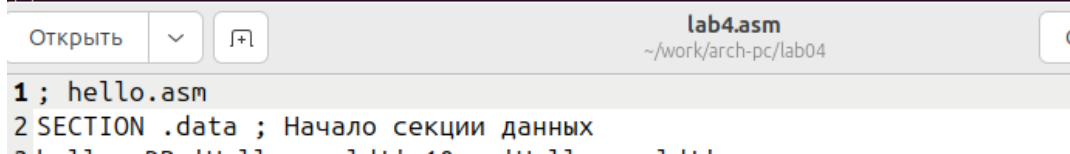


Рис. 4.9: Создание копии файла

Редактирую текст программы для того, чтобы вывести свое имя (рис.4.10).

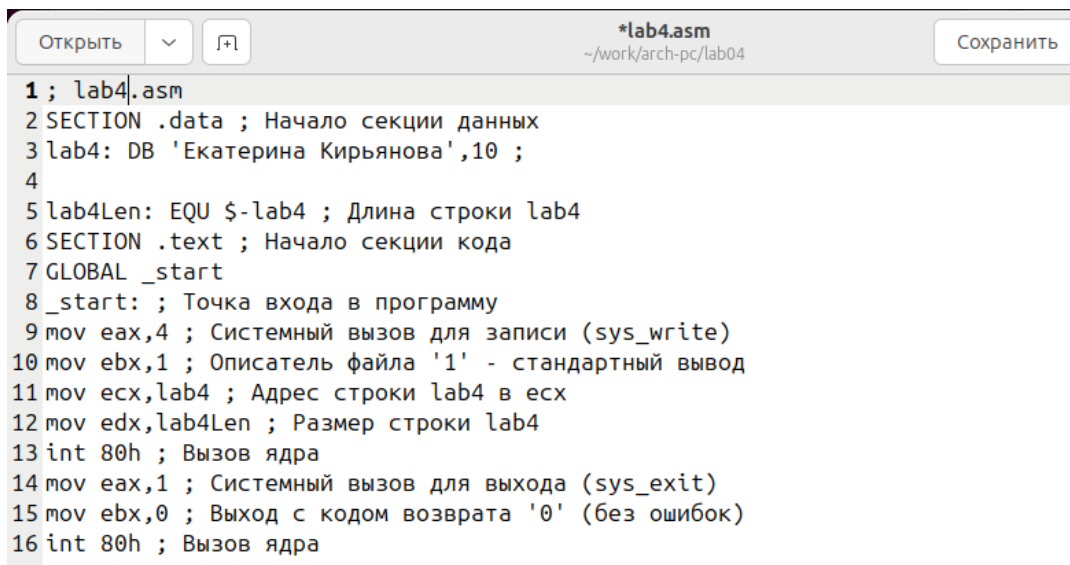


Рис. 4.10: Внесение изменений

Транслирую полученный текст программы в объектный файл и проверяю успешность выполнения (рис.4.11).

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

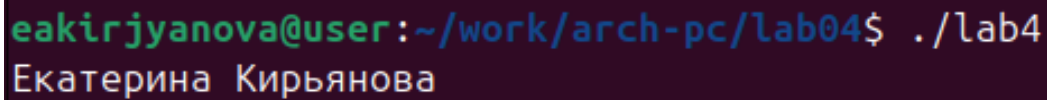
Рис. 4.11: Компиляция текста программы

Передаю объектный файл компоновщику LD (рис.4.12).

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

Рис. 4.12: Обработка файла

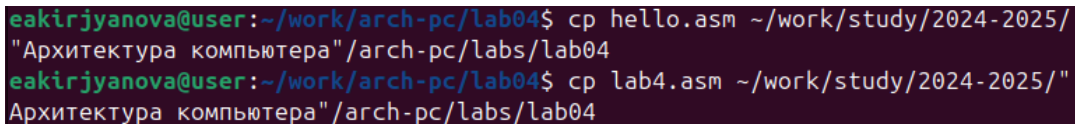
Запускаю исполняемый файл (рис.4.13).



```
eakirjyanova@user:~/work/arch-pc/lab04$ ./lab4
Екатерина Кирьянова
```

Рис. 4.13: Запуск программы

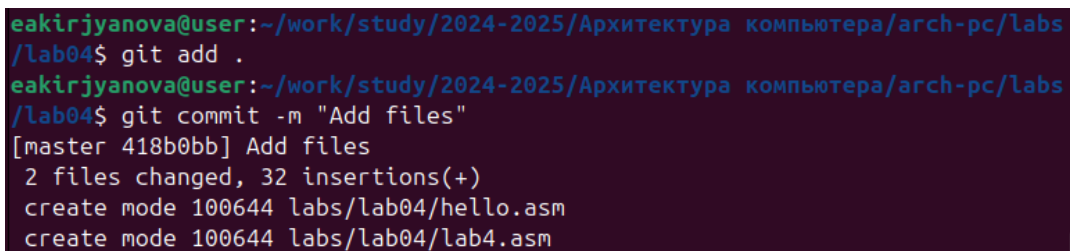
Копирую файлы hello.asm и lab4.asm в свой локальный репозиторий (рис.4.14).



```
eakirjyanova@user:~/work/arch-pc/lab04$ cp hello.asm ~/work/study/2024-2025/
"Архитектура компьютера"/arch-pc/labs/lab04
eakirjyanova@user:~/work/arch-pc/lab04$ cp lab4.asm ~/work/study/2024-2025/
"Архитектура компьютера"/arch-pc/labs/lab04
```

Рис. 4.14: Заполнение файла

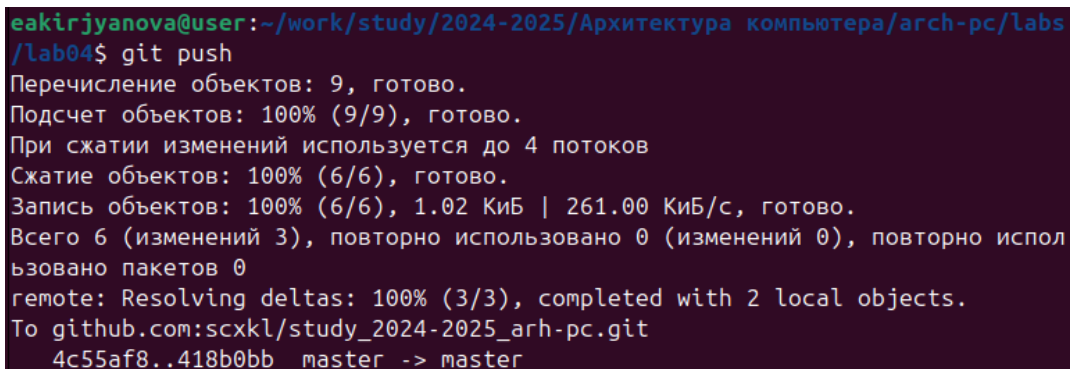
Добавляю файлы на GitHub (рис.4.15).



```
eakirjyanova@user:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs
/lab04$ git add .
eakirjyanova@user:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs
/lab04$ git commit -m "Add files"
[master 418b0bb] Add files
2 files changed, 32 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
```

Рис. 4.15: Добавление файлов

Отправляю файлы на сервер с помощью git push (рис.4.16).



```
eakirjyanova@user:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs
/lab04$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.02 КиБ | 261.00 КиБ/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно испол
зовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:scxkl/study_2024-2025_arh-pc.git
4c55af8..418b0bb master -> master
```

Рис. 4.16: Отправление файлов на сервер

Проверяю успешность выполнения (рис.4.17).

[study\\_2024-2025\\_arh-pc](#) / [labs](#) / [lab04](#)

Add file
...

sckl
Add files

418b0bb · 1 minute ago
History

Name	Last commit message	Last commit date
..		
presentation	feat(main): make course structure	3 weeks ago
report	feat(main): make course structure	3 weeks ago
hello.asm	Add files	1 minute ago
lab4.asm	Add files	1 minute ago

Рис. 4.17: Проверка

## **5 Выводы**

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

# **Список литературы**

1. Архитектура ЭВМ