

# **Отчёт по лабораторной работе №4**

**Дисциплина: архитектура компьютера**

**Панявкина Ирина Васильевна**

**НКАбд-04-24**

# Содержание

1 Цель работы.....	4
2 Задание.....	4
3 Теоретическое введение.....	4
4 Выполнение лабораторной работы.....	6
4.1 Создание программы Hello world!.....	6
4.2 Работа с транслятором NASM.....	8
4.3 Работа с расширенным синтаксисом командной строки NASM.....	8
4.4 Работа с компоновщиком LD.....	9
4.5 Запуск исполняемого файла.....	9
4.6 Выполнение заданий для самостоятельной работы.....	10
5 Выводы.....	12
6 Список литературы.....	13

# Список иллюстраций

- 4.1 Перемещение между директориями
- 4.2 Создание пустого файла
- 4.3 Открытие файла в текстовом редакторе
- 4.4 Заполнение файла
- 4.5 Компиляция текста программы
- 4.6 Компиляция текста программы
- 4.7 Передача объектного файла на обработку компоновщику
- 4.8 Передача объектного файла на обработку компоновщику
- 4.9 Запуск исполняемого файла
- 4.10 Создание копии файла
- 4.11 Изменение программы
- 4.12 Компиляция текста программы
- 4.13 Передача объектного файла на обработку компоновщику
- 4.14 Запуск исполняемого файла
- 4.15 Создании копии файлов в новом каталоге
- 4.16 Удаление лишних файлов в текущем каталоге
- 4.17 Добавление файлов и отправка файлов на GitHub

# 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. 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 6 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 7 следующей команде.

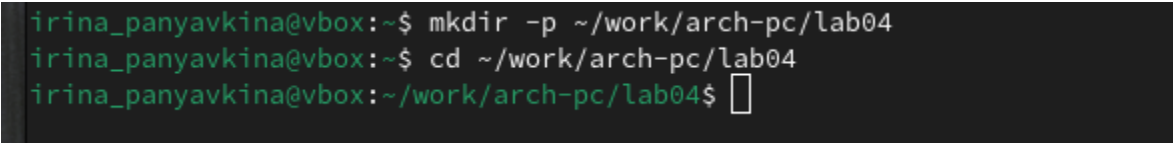
Язык ассемблера (assembly language, сокращённо asm) — машинноориентированный язык низкого уровня. NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64.

## 4 Выполнение лабораторной работы

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 1).

### 4.1 Создание программы Hello world!

С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать (рис. 4.1).



```
irina_panyavkina@vbox:~$ mkdir -p ~/work/arch-pc/lab04
irina_panyavkina@vbox:~$ cd ~/work/arch-pc/lab04
irina_panyavkina@vbox:~/work/arch-pc/lab04$
```

Рис. 4.1: Перемещение между директориями

Создаю в текущем каталоге пустой текстовый файл hello.asm с помощью утилиты touch (рис. 4.2).

```
irina_panyavkina@vbox:~/work/arch-pc/lab04$ touch hello.asm
```

Рис. 4.2: Создание пустого файла

Открываю созданный файл в текстовом редакторе mousepad (рис. 4.3)

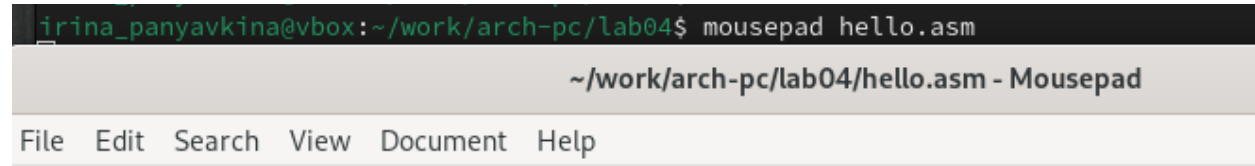


Рис. 4.3: Открытие файла в текстовом редакторе

Заполняю файл, вставляя в него программу для вывода “Hello word!” (рис. 4.4).

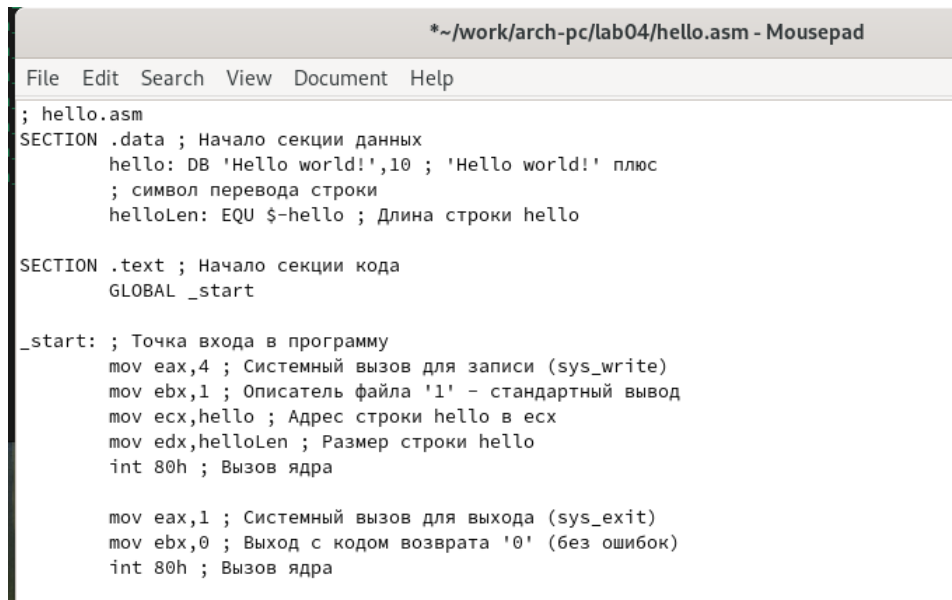


Рис. 4.4: Заполнение файла

## 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`, ключ `-f` указывает транслятору `nasm`, что требуется создать бинарный файл в формате ELF (рис. 4.5). Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “`hello.o`”.

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

Рис. 4.5: Компиляция текста программы

## 4.3 Работа с расширенным синтаксисом командной строки NASM

Ввожу команду, которая скомпилирует файл `hello.asm` в файл `obj.o`, при этом в файл будут включены символы для отладки (ключ `-g`), также с помощью ключа `-l 10` будет создан файл листинга `list.lst` (рис. 4.6). Далее проверяю с помощью утилиты `ls` правильность выполнения команды.

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

Рис. 4.6: Компиляция текста программы



## 4.4 Работа с компоновщиком LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (рис. 4.7). Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды.

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

Рис. 4.7: Передача объектного файла на обработку компоновщику

Выполняю следующую команду (рис. 4.8). Исполняемый файл будет иметь имя main, т.к. после ключа -o было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o.

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

Рис. 4.8: Передача объектного файла на обработку компоновщику

## 4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. 4.9).

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

Рис. 4.9: Запуск исполняемого файла

## 4.6 Выполнение заданий для самостоятельной работы.

С помощью утилиты `cp` создаю в текущем каталоге копию файла `hello.asm` с именем `lab4.asm` (рис. 4.10).

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

Рис. 4.10: Создание копии файла

С помощью текстового редактора `mousepad` открываю файл `lab4.asm` и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис. 4.11).

```
*~/work/arch-pc/lab04/lab4.asm - Mousepad
File Edit Search View Document Help
; lab4.asm
SECTION .data ; Начало секции данных
    lab4: DB 'Irina Panyavkina',10

    lab4Len: EQU $-lab4 ; Длина строки lab4

SECTION .text ; Начало секции кода
    GLOBAL _start

_start: ; Точка входа в программу
    mov eax,4 ; Системный вызов для записи (sys_write)
    mov ebx,1 ; Описатель файла '1' - стандартный вывод
    mov ecx,lab4 ; Адрес строки lab4 в ecx
    mov edx,lab4Len ; Размер строки lab4
    int 80h ; Вызов ядра

    mov eax,1 ; Системный вызов для выхода (sys_exit)
    mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
    int 80h ; Вызов ядра
```

Рис. 4.11: Изменение программы

Компилирую текст программы в объектный файл (рис. 4.12). Проверяю с помощью утилиты ls, что файл lab4.o создан.

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

Рис. 4.12: Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. 4.13).

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

Рис. 4.13: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия (рис. 4.14).

```
irina_panyavkina@vbox:~/work/arch-pc/lab04$ ./lab4
Irina Panyavkina
```

Рис. 4.14: Запуск исполняемого файла

Далее копирую из текущего каталога файлы, созданные в процессе выполнения лабораторной работы, с помощью утилиты cp, указывая вместо имени файла символ \*, чтобы скопировать все файлы. Команда проигнорирует директории в этом каталоге, т. к. не указан ключ -r, это мне и нужно (рис. 4.15). Проверяю с помощью утилиты ls правильность выполнения команды.

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

Рис. 4.15: Создании копии файлов в новом каталоге

Удаляю лишние файлы в текущем каталоге с помощью утилиты `rm`, ведь копии файлов остались в другой директории (рис. 4.16).

```
irina_panyavkina@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ rm hello hello.o lab4 lab4.o list.lst main obj.o
irina_panyavkina@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm lab4.asm presentation report
```

Рис. 4.16: Удаление лишних файлов в текущем каталоге

С помощью команд `git add .` и `git commit` добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №4 и отправляю файлы на сервер с помощью команды `git push` (рис. 4.17).

```
irina_panyavkina@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git add .
irina_panyavkina@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git commit -m "Add files for lab04"
[master 5820919] Add files for lab04
2 files changed, 38 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
irina_panyavkina@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.04 KiB | 1.04 MiB/s, done.
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:irinapanyavkina/study_2024-2025_arhpc.git
318a760..5820919 master -> master
```

Рис. 4.17: Добавление и отправка файлов на GitHub

## 5 Выводы

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

## Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Lupin С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.

13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).