

# **Отчёт по лабораторной работе № 4**

**дисциплина: Архитектура компьютера**

Терещенкова Маргарита Владимировна

# 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>9</b>
4.1	Создание программы Hello world! . . . . .	9
4.2	Работа с транслятором NASM . . . . .	10
4.3	Работа с расширенным синтаксисом командной строки NASM . . .	11
4.4	Компоновщик LD . . . . .	11
4.5	Запуск исполняемого файла . . . . .	12
<b>5</b>	<b>Выполнение заданий для самостоятельной работы.</b>	<b>13</b>
<b>6</b>	<b>Исправление ошибки</b>	<b>15</b>
<b>7</b>	<b>Выводы</b>	<b>17</b>
	<b>Список литературы</b>	<b>18</b>

# Список иллюстраций

4.1	Перемещение между директориями. Создание пустого файла. . . . .	9
4.2	Заполнение файла . . . . .	10
4.3	Установка <i>rust</i> . . . . .	10
4.4	Компиляция текста программы . . . . .	11
4.5	Компиляция текста программы . . . . .	11
4.6	Передача объектного файла на обработку компоновщику . . . . .	12
4.7	Передача объектного файла на обработку компоновщику . . . . .	12
4.8	Запуск исполняемого файла . . . . .	12
5.1	Создание копии файла . . . . .	13
5.2	Изменение программы . . . . .	13
5.3	Компиляция текста программы . . . . .	14
5.4	Передача объектного файла на обработку компоновщику . . . . .	14
5.5	Запуск исполняемого файла . . . . .	14
6.1	Создании директории . . . . .	15
6.2	Создании копии файлов в новом каталоге . . . . .	15
6.3	Проверка удаления . . . . .	15
6.4	Добавление файлов на <i>GitHub</i> . . . . .	16
6.5	Отправка файлов . . . . .	16

# 1 Цель работы

Освоение 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 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.

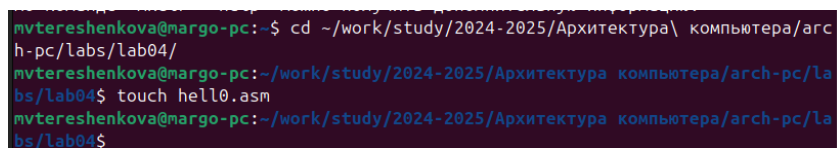
Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются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



## 4 Выполнение лабораторной работы

### 4.1 Создание программы Hello world!

С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать. Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch`.



```
mvtereshenkova@margo-pc:~$ cd ~/work/study/2024-2025/Архитектура\ компьютера/arch-
h-pc/labs/lab04/
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/l
bs/lab04$ touch hello.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/l
bs/lab04$
```

Рис. 4.1: Перемещение между директориями. Создание пустого файла.

Открываю созданный файл в текстовом редакторе `mousepad`. Заполняю файл, вставляя в него программу для вывода “Hello word!”

```

; hello.asm
SECTION .data
; Начало секции данных
hello:DB 'Hello world!',10 ; 'Hello world!' плюс
helloLen:EQU $-hello
; символ перевода строки
SECTION .text
; Длина строки hello
; 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,hello; Адрес строки hello в ecx
mov edx,helloLen ; Размер строки hello
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. 4.2: Заполнение файла

## 4.2 Работа с транслятором NASM

Устанавливаем nasm с помощью `sudo apt install nasm`

```

mvtreshenkova@margo-pc:~$ sudo apt install nasm
[sudo] пароль для mvtreshenkova:
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Следующие НОВЫЕ пакеты будут установлены:
  nasm
Обновлено 0 пакетов, установлено 1 новых пакетов, для удаления отмечено 0 пакетов,
и 9 пакетов не обновлено.
Необходимо скачать 459 кВ архивов.
После данной операции объём занятого дискового пространства возрастёт на 3 407 кВ.
Пол:1 http://ru.archive.ubuntu.com/ubuntu noble/universe amd64 nasm amd64 2.16.0
1-1build1 [459 kB]
Получено 459 кВ за 0с (1 014 кВ/с)
Выбор ранее не выбранного пакета nasm.
dpkg: предупреждение: список файлов пакета «texlive» отсутствует; предполагаем,
что на данный момент у пакета нет установленных файлов
(Чтение базы данных ... на данный момент установлено 366253 файла и каталога.)
Подготовка к распаковке .../nasm_2.16.01-1build1_amd64.deb ...
Распаковывается nasm (2.16.01-1build1) ...
Настраивается пакет nasm (2.16.01-1build1) ...
Обрабатываются триггеры для man-db (2.12.0-4build2) ...

```

Рис. 4.3: Установка 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. Далее проверяю правильность выполнения команды с помощью утилиты `ls`: файл “`hello.o`” создан.

```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab04$ nasm -f elf hello.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab04$ ls
hello.asm  hello.o  presentation  report
```

Рис. 4.4: 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` будет создан файл листинга `list.lst`. Далее проверяю с помощью утилиты `ls` правильность выполнения команды.

```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
```

Рис. 4.5: Компиляция текста программы

### 4.4 Компоновщик LD

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

```

mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab04$ ls
hello.asm  hello  hello.asm  hello.o  list.lst  obj.o  presentation  report

```

Рис. 4.6: Передача объектного файла на обработку компоновщику

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

```

bs/lab04$ ld -m elf_i386 obj.o -o main
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab04$ ls
hello.asm  hello.asm  list.lst  obj.o      report
hello      hello.o    main      presentation

```

Рис. 4.7: Передача объектного файла на обработку компоновщику

## 4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл `hello`

```

mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab04$ ./hello
Hello world!

```

Рис. 4.8: Запуск исполняемого файла

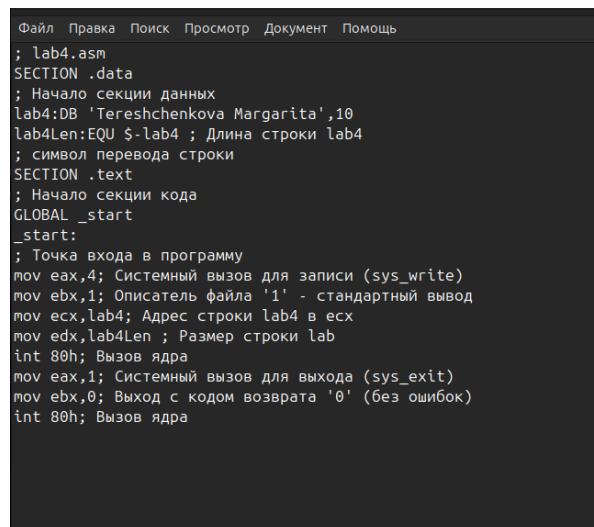
## 5 Выполнение заданий для самостоятельной работы.

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

```
mvtreshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab4$ cp hello.asm lab4.asm
```

Рис. 5.1: Создание копии файла

С помощью текстового редактора `mouesrad` открываю файл `lab4.asm` и вношу изменения в программу так, чтобы она выводила мои имя и фамилию.



```
Файл  Правка  Поиск  Просмотр  Документ  Помощь
; lab4.asm
SECTION .data
; Начало секции данных
lab4:DB 'Tereshchenkova Margarita',10
lab4Len:EQU $-lab4 ; Длина строки lab4
; символ перевода строки
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 lab
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. 5.2: Изменение программы

Компилирую текст программы в объектный файл. Проверяю с помощью утилиты `ls`, что файл `lab4.o` создан.

```

mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab4$ nasm -f elf lab4.asm
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab4$ ls
hello.asm  hello.o  lab4.o  list.lst  obj.o  report
hello      hello.o  lab4.o  main      presentation

```

Рис. 5.3: Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4

```

mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab4$ ld -m elf_i386 lab4.o -o lab4
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab4$ ls
hello.asm  hello.asm  lab4      lab4.o  main  presentation
hello      hello.o    lab4.asm  list.lst  obj.o  report

```

Рис. 5.4: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, на экран выводятся мои имя и фамилия

```

mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab4$ ./lab4
Tereshchenkova Margarita

```

Рис. 5.5: Запуск исполняемого файла

## 6 Исправление ошибки

По привычке я начала работу не в том каталоге, поэтому создаю директорию lab04 с помощью mkdir (как указано в порядке выполнения лабораторной работы) 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. Проверяю с помощью утилиты ls правильность выполнения команды.

```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab04$ mkdir ~/work/study/2024-2025/Архитектура\ компьютера/arch-pc/lab04
```

Рис. 6.1: Создании директории

```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab04$ cp * ~/work/study/2024-2025/Архитектура\ компьютера/arch-pc/lab04
cp: не указан -r; пропускается каталог 'presentation'
cp: не указан -r; пропускается каталог 'report'
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab04$ ls ~/work/study/2024-2025/Архитектура\ компьютера/arch-pc/lab04
hello.asm  hello.asm  lab4       lab4.o     main
hello      hello.o    lab4.asm  list.lst  obj.o
```

Рис. 6.2: Создании копии файлов в новом каталоге

Удаляю лишние файлы в текущем каталоге с помощью утилиты rm, ведь копии файлов остались в другой директории.

```
mvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab04$ ls
hello.asm  lab4.asm  presentation  report
```

Рис. 6.3: Проверка удаления

С помощью команд `git add .` и `git commit` добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы.

```
nvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ g
it add .
nvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ g
it commit -m "Add fales for lab04"
[master 1c3c941] Add fales for lab04
13 files changed, 130 insertions(+), 32 deletions(-)
create mode 100644 lab04/hello.asm
create mode 100755 lab04/hello
create mode 100644 lab04/hello.asm
create mode 100644 lab04/hello.o
create mode 100755 lab04/lab4
create mode 100644 lab04/lab4.asm
create mode 100644 lab04/lab4.o
create mode 100644 lab04/list.lst
create mode 100755 lab04/main
create mode 100644 lab04/obj.o
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
```

Рис. 6.4: Добавление файлов на GitHub

Отправляю файлы на сервер с помощью команды `git push`

```
nvtereshenkova@margo-pc:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ g
it push
Перечисление объектов: 21, готово.
Подсчет объектов: 100% (21/21), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (16/16), 7.50 КиБ | 1.88 МиБ/с, готово.
Всего 16 (изменений 8), повторно использовано 0 (изменений 0), повторно использо
вано пакетов 0
remote: Resolving deltas: 100% (8/8), completed with 3 local objects.
To github.com:ritondriy/study_2024-2025_arh-pc.git
1519154..1c3c941 master -> master
```

Рис. 6.5: Отправка файлов



## **7 Выводы**

Благодаря данной лабораторной работе освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

# **Список литературы**

1. Архитектура ЭВМ