

# **Лабораторная работа №4**

**Дисциплина: Архитектура компьютера**

Малюга Валерия Васильевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
3.1	Основные принципы работы компьютера . . . . .	7
3.2	Ассемблер и язык ассемблера . . . . .	8
3.3	Процесс создания и обработки программы на языке ассемблера .	8
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>10</b>
4.1	Создание программы Hello world! . . . . .	10
4.2	Работа с транслятором NASM . . . . .	11
4.3	Работа с расширенным синтаксисом командной строки NASM . .	11
4.4	Работа с компоновщиком LD . . . . .	11
4.5	Запуск исполняемого файла . . . . .	12
4.6	Выполнение заданий для самостоятельной работы . . . . .	12
<b>5</b>	<b>Выводы</b>	<b>15</b>

## Список таблиц

## Список иллюстраций

4.1	Создание программы Hello world . . . . .	10
4.2	Компиляция программы hello . . . . .	11
4.3	Компиляция текста программы . . . . .	11
4.4	Передача объектного файла на обработку компоновщику . . . . .	12
4.5	Запуск программы Hello world . . . . .	12
4.6	Изменение программы lab4.asm . . . . .	13
4.7	Трансляция в объектный файл и его компоновка. Запуск исполняе- мого файла . . . . .	13
4.8	Копирование файлов в локальный репозиторий и отправка на Github	14

# 1 Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ,

## 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

### 3.1 Основные принципы работы компьютера

Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ)

Основной задачей процессора является обработка информации, а также организация ко

- **\*\*арифметико-логическое устройство (АЛУ)\*\*** – выполняет логические и арифметичес
- **\*\*устройство управления (УУ)\*\*** – обеспечивает управление и контроль всех устрой
- **\*\*регистры\*\*** – сверхбыстрая оперативная память небольшого объёма, входящая в со

Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры

Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по имен

Другим важным узлом ЭВМ является **\*\*оперативное запоминающее устройство (ОЗУ)\*\***. О

В состав ЭВМ также входят **\*\*периферийные устройства\*\***, которые можно разделить на

- \*\*\*устройства внешней памяти\*\*\***, которые предназначены для длительного хране
- \*\*\*устройства ввода-вывода\*\*\***, которые обеспечивают взаимодействие ЦП с внешней

В основе вычислительного процесса ЭВМ лежит **\*\*принцип программного управления\*\***.

Набор машинных команд определяется устройством конкретного процессора. В коде маш

При выполнении каждой команды процессор выполняет определённую последовательность

1. формирование адреса в памяти очередной команды;
2. считывание кода команды из памяти и её дешифрация;
3. выполнение команды;
4. переход к следующей команде.

## 3.2 Ассемблер и язык ассемблера

**\*\*Язык ассемблера\*\*** (assembly language, сокращённо asm) – машинно-

ориентированный язык низкого уровня. Можно считать, что он больше любых других языков.

Процессор понимает не команды ассемблера, а последовательности из нулей и единиц.

NASM – это открытый проект ассемблера, версии которого доступны под различные операционные системы. Он поддерживает инструкции x86-64.

Типичный формат записи команд NASM имеет вид:

[метка:] мнемокод [операнд {, операнд}] [; комментарий]

Здесь **\*\*мнемокод\*\*** – непосредственно мнемоника инструкции процессору, которая является обязательной частью команды. **\*\*Операндами\*\*** могут быть числа, данные, адреса регистров.

Программа на языке ассемблера также может содержать **\*\*директивы\*\*** – инструкции, начинающиеся с директивной директивы.

## 3.3 Процесс создания и обработки программы на языке ассемблера

В процессе создания ассемблерной программы можно выделить четыре шага:

- **Набор текста** программы в текстовом редакторе и сохранение её в отдельном файле. Каждый файл имеет свой тип (или расширение), который определяет назначение файла. Файлы с исходным текстом программ на языке ассемблера имеют тип asm.
- **Трансляция** – преобразование с помощью транслятора, например nasm, текста программы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную дополнительную информацию, созданную транслятором. Тип объектного файла – o, файла листинга – lst.
- **Компоновка или линковка** – этап обработки объектного кода компоновщиком (ld), который принимает на вход объектные файлы и собирает по ним исполняемый файл. Исполняемый файл обычно не имеет расширения. Кроме того, можно получить файл карты загрузки программы в ОЗУ, имеющий расширение map.
- **Запуск**

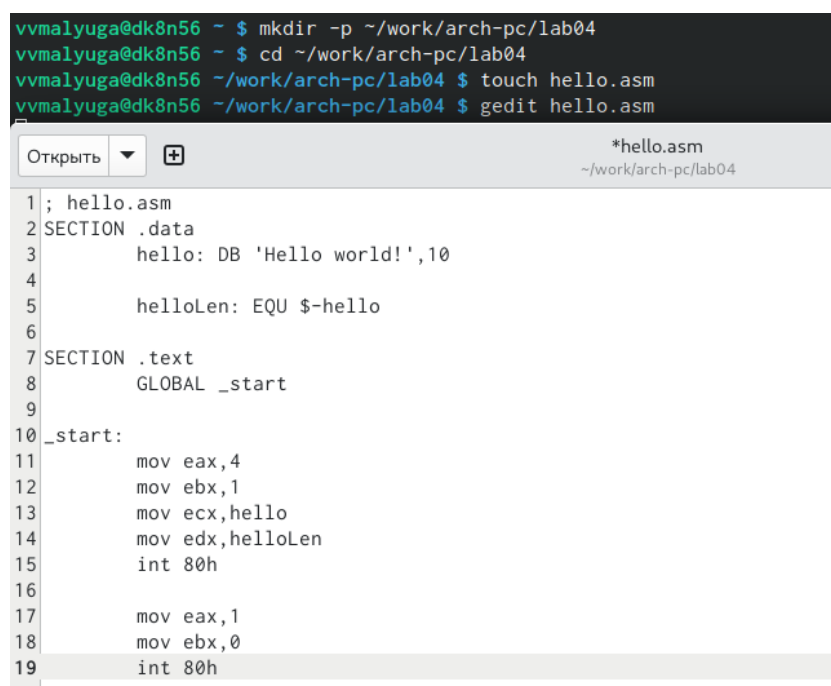


**программы.** Конечной целью является работоспособный исполняемый файл. Ошибки на предыдущих этапах могут привести к некорректной работе программы, поэтому может присутствовать этап отладки программы при помощи специальной программы — отладчика. При нахождении ошибки необходимо провести коррекцию программы, начиная с первого шага.

## 4 Выполнение лабораторной работы

### 4.1 Создание программы Hello world!

Создаю каталог для работы с программами на языке ассемблера NASM. Перешла в созданный каталог, создала текстовый файл с именем `hello.asm`. Открыла этот файл с помощью текстового редактора `gedit` и ввела в него необходимый текст (рис. 4.1).



```
vvimalyuga@dk8n56 ~ $ mkdir -p ~/work/arch-pc/lab04
vvimalyuga@dk8n56 ~ $ cd ~/work/arch-pc/lab04
vvimalyuga@dk8n56 ~/work/arch-pc/lab04 $ touch hello.asm
vvimalyuga@dk8n56 ~/work/arch-pc/lab04 $ gedit hello.asm
```

The screenshot shows a terminal window with the commands to create the directory, change to it, create the file, and open it with gedit. Below the terminal is a window of the gedit text editor. The title bar shows the file name as `*hello.asm` and the path as `~/work/arch-pc/lab04`. The editor contains the following assembly code:

```
1 ; hello.asm
2 SECTION .data
3     hello: DB 'Hello world!',10
4
5     helloLen: EQU $-hello
6
7 SECTION .text
8     GLOBAL _start
9
10 _start:
11     mov eax,4
12     mov ebx,1
13     mov ecx,hello
14     mov edx,helloLen
15     int 80h
16
17     mov eax,1
18     mov ebx,0
19     int 80h
```

Рис. 4.1: Создание программы Hello world

## 4.2 Работа с транслятором NASM

Для компиляции программы “Hello world” написала в терминале команду `nasm -f elf hello.asm`. Впоследствии проверила выполнение этой команды с помощью `ls`. Действительно, транслятор преобразовал `hello.asm` в `hello.o` (рис. 4.2).

```
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $
```

Рис. 4.2: Компиляция программы hello

## 4.3 Работа с расширенным синтаксисом командной строки NASM

Ввела команду, которая скомпилировала файл `hello.asm` в файл `obj.o`, при этом в файл были включены символы для отладки (ключ `-g`), также с помощью ключа `-l` был создан файл листинга `list.lst`. Далее проверила с помощью утилиты `ls` правильность выполнения команды (рис. [4.3]).

```
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $
```

Рис. 4.3: Компиляция текста программы

## 4.4 Работа с компоновщиком LD

Передала объектный файл на обработку компоновщику, чтобы получить исполняемую программу. Выполнила 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` (рис.

[4.4]).

```
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
```

Рис. 4.4: Передача объектного файла на обработку компоновщику

## 4.5 Запуск исполняемого файла

Запустила созданный исполняемый файл, набрав в терминале команду (рис. [4.5]).

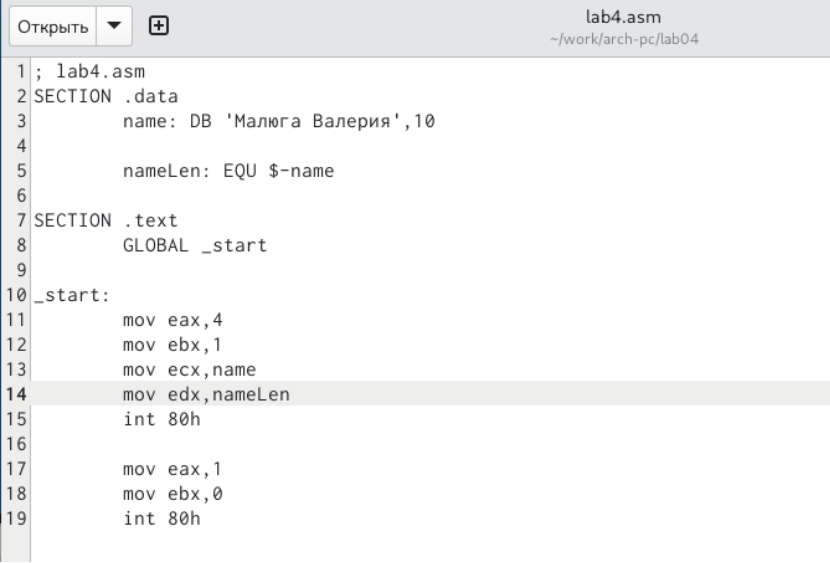
```
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ./hello
Hello world!
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ █
```

Рис. 4.5: Запуск программы Hello world

## 4.6 Выполнение заданий для самостоятельной работы

1. В каталоге ~/work/arch-pc/lab04 с помощью команды `cp` создала копию файла `hello.asm` с именем `lab4.asm` (рис. [4.6]).
2. С помощью текстового редактора `gedit` внесла изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с моими фамилией и именем (рис. [4.6]).

```
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ gedit lab4.asm
```



```
1; lab4.asm
2SECTION .data
3    name: DB 'Малюга Валерия',10
4
5    nameLen: EQU $-name
6
7SECTION .text
8    GLOBAL _start
9
10_start:
11    mov eax,4
12    mov ebx,1
13    mov ecx,name
14    mov edx,nameLen
15    int 80h
16
17    mov eax,1
18    mov ebx,0
19    int 80h
```

Рис. 4.6: Изменение программы lab4.asm

3. Оттранслировала полученный текст программы lab4.asm в объектный файл. Выполнила компоновку объектного файла и запустила получившийся исполняемый файл (рис. [4.7]).

```
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ./lab4
Малюга Валерия
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $
```

Рис. 4.7: Трансляция в объектный файл и его компоновка. Запуск исполняемого файла

4. Скопировала файлы hello.asm и lab4.asm в локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/. Загрузила файлы на Github (рис. [4.8]).

```

vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ cp hello.asm ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/hello.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ cp lab4.asm ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/lab4.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/
vvmalyuga@dk8n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello.asm  lab4.asm  presentation  report
vvmalyuga@dk8n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git add .
vvmalyuga@dk8n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git commit -am 'feat(main): add files lab-4'
[master 6e0d07a] feat(main): add files lab-4
2 files changed, 38 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
vvmalyuga@dk8n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 796 байтов | 796.00 КиБ/с, готово.
Всего 6 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:vvmalyuga/study_2023-2024_arhpc.git
64c53a2..6e0d07a master -> master
vvmalyuga@dk8n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $

```

Рис. 4.8: Копирование файлов в локальный репозиторий и отправка на Github

## **5 Выводы**

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