

# **Лабораторная работа №4. Создание и процесс обработки программ на языке ассемблера NASM**

**Простейший вариант**

Диана Садова Алексевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
2.1	Порядок выполнения лабораторной работы . . . . .	6
2.1.1	Программа Hello world! . . . . .	6
2.1.2	Транслятор NASM . . . . .	8
2.1.3	Расширенный синтаксис командной строки NASM . . . . .	9
2.2	Компоновщик LD . . . . .	9
2.2.1	Запуск исполняемого файла . . . . .	11
<b>3</b>	<b>Теоретическое введение</b>	<b>12</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>13</b>
4.1	Задание для самостоятельной работы . . . . .	13
4.1.1	В каталоге ~/work/arch-pc/lab04 с помощью команды cp создайте копию файла hello.asm с именем lab4.asm (рис.4.1) .	13
4.1.2	С помощью любого текстового редактора внесите изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! На экран выводилась строка с вашими фамилией и именем (рис.4.2)(рис.4.3) . . . . .	14
4.1.3	Оттранслируйте полученный текст программы lab4.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл (рис.4.4) . .	15
4.1.4	Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/. Загрузите файлы на Github (рис.4.5)(рис.4.6) . . . . .	15
<b>5</b>	<b>Выводы</b>	<b>16</b>
	<b>Список литературы</b>	<b>17</b>

## Список иллюстраций

2.1	Создаем каталог для работы на языке NASM . . . . .	6
2.2	Создаем каталог . . . . .	6
2.3	Создаем текстовый файл . . . . .	7
2.4	Проверяем его наличие . . . . .	7
2.5	Открываем текстовый файл с помощью редактора gedit . . . . .	7
2.6	Вводим текст в hello.asm . . . . .	7
2.7	Преображаем текст в код программы . . . . .	8
2.8	Вводим полный вариант командной строки nasm . . . . .	9
2.9	Проверяем наличие файла . . . . .	9
2.10	Передаем проект на обработку компоновщику . . . . .	10
2.11	Проверяем корректность выполненной работы . . . . .	10
2.12	Создаем исполнитель . . . . .	10
2.13	Запускаем созданный исполняемый файл . . . . .	11
4.1	Создаем копию hello.asm с именем lab4.asm . . . . .	13
4.2	Открываем с помощью текстового редактора и вносим изменения . . . . .	14
4.3	Проверяем, что изменения внесены верно . . . . .	14
4.4	Оттранслируем полученный текст в объектный файл . . . . .	15
4.5	Копируем файлы hello.asm и lab4.asm в локальный репозиторий . . . . .	15
4.6	Загружаем файл на Github . . . . .	15

## Список таблиц

# 1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

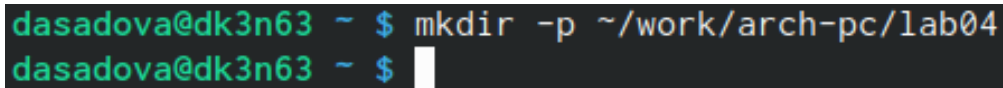
## 2 Задание

### 2.1 Порядок выполнения лабораторной работы

#### 2.1.1 Программа Hello world!

Рассмотрим пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит приветственное сообщение Hello world! на экран.

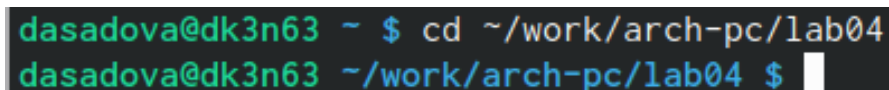
Создайте каталог для работы с программами на языке ассемблера NASM (рис.2.1)



```
dasadova@dk3n63 ~ $ mkdir -p ~/work/arch-pc/lab04
dasadova@dk3n63 ~ $
```

Рис. 2.1: Создаем каталог для работы на языке NASM

Перейдите в созданный каталог (рис.2.2)



```
dasadova@dk3n63 ~ $ cd ~/work/arch-pc/lab04
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```

Рис. 2.2: Создаем каталог

Создайте текстовый файл с именем hello.asm (рис.2.3)(рис.2.4)

```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ touch hello.asm
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```

Рис. 2.3: Создаем текстовый файл

```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ ls
hello.asm
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```

Рис. 2.4: Проверяем его наличие

Проверив его наличие мы удостоверились, что программа выполнена корректно. Откройте этот файл с помощью любого текстового редактора, например, gedit (рис.2.5)

```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 2.5: Открываем текстовый файл с помощью редактора gedit

С помощью gedit начинаем редактировать текст для дальнейшей работы. И введите в него следующий текст (рис.2.6)

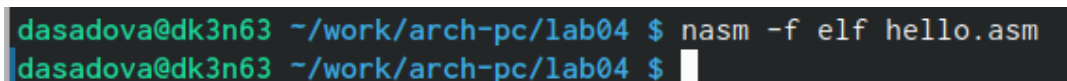
```
1 SECTION .data
2 hello: DB 'Hello world!',10
3
4 helloLen: EQU $-hello
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9 mov eax,4
10 mov ebx,1
11 mov ecx,hello
12 mov edx,helloLen
13 int 80h
14 mov eax,1
15 mov ebx,0
16 int 80h
```

Рис. 2.6: Вводим текст в hello.asm

В отличие от многих современных высокоуровневых языков программирования, в ассемблерной программе каждая команда располагается на отдельной строке. Размещение нескольких команд на одной строке недопустимо. Синтаксис ассемблера NASM является чувствительным к регистру, т.е. есть разница между большими и малыми буквами.

### 2.1.2 Транслятор NASM

NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World» необходимо написать (рис.2.7)



```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```

Рис. 2.7: Преображаем текст в код программы

Если текст программы набран без ошибок, то транслятор преобразует текст программы из файла `hello.asm` в объектный код, который запишется в файл `hello.o`. Таким образом, имена всех файлов получаются из имени входного файла и расширения по умолчанию. При наличии ошибок объектный файл не создаётся, а после запуска транслятора появятся сообщения об ошибках или предупреждения.

С помощью команды `ls` проверьте, что объектный файл был создан. Какое имя имеет объектный файл?

NASM не запускают без параметров. Ключ `-f` указывает транслятору, что требуется создать бинарные файлы в формате ELF. Следует отметить, что формат `elf64` позволяет создавать исполняемый код, работающий под 64-битными версиями Linux. Для 32-битных версий ОС указываем в качестве формата просто `elf`. NASM всегда создаёт выходные файлы в текущем каталоге.



### 2.1.3 Расширенный синтаксис командной строки NASM

Полный вариант командной строки `nasm` выглядит следующим образом: `nasm [-@ косвенный_файл_настроек] [-о объектный_файл] [-f формат_объектного_файла] [-l листинг] [параметры...] [-] исходный_файл`

Выполните следующую команду (рис.2.8)

```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```

Рис. 2.8: Вводим полный вариант командной строки `nasm`

Данная команда скомпилирует исходный файл `hello.asm` в `obj.o` (опция `-o` позволяет задать имя объектного файла, в данном случае `obj.o`), при этом формат выходного файла будет `elf`, и в него будут включены символы для отладки (опция `-g`), кроме того, будет создан файл листинга `list.lst` (опция `-l`).

С помощью команды `ls` проверьте, что файлы были созданы (рис.2.9)

```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```

Рис. 2.9: Проверяем наличие файла

Для более подробной информации см. `man nasm`. Для получения списка форматов объектного файла см. `nasm -hf`. # Теоретическое введение

## 2.2 Компоновщик LD

Как видно из схемы на рис. 4.3, чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику (рис.2.10)

```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```

Рис. 2.10: Передаем проект на обработку компоновщику

Компоновщик — инструментальная программа, которая производит компоновку («линковку»): принимает на вход один или несколько объектных модулей и собирает из них исполняемый или библиотечный файл-модуль.

С помощью команды `ls` проверьте, что исполняемый файл `hello` был создан (рис.2.11)

```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o list.lst obj.o
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```

Рис. 2.11: Проверяем корректность выполненной работы

Компоновщик `ld` не предполагает по умолчанию расширений для файлов, но принято использовать следующие расширения: • `o` – для объектных файлов; • без расширения – для исполняемых файлов; • `tar` – для файлов схемы программы; • `lib` – для библиотек.

Ключ `-o` с последующим значением задаёт в данном случае имя создаваемого исполняемого файла.

Выполните следующую команду (рис.2.12)

```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```


Рис. 2.12: Создаем исполнитель

Какое имя будет иметь исполняемый файл? Какое имя имеет объектный файл из которого собран этот исполняемый файл?

Формат командной строки `LD` можно увидеть, набрав `ld -help`. Для получения более подробной информации см. `man ld`.

### 2.2.1 Запуск исполняемого файла

Запустить на выполнение созданный исполняемый файл, находящийся в текущем каталоге, можно, набрав в командной строке (рис.2.13)

A screenshot of a terminal window with a dark background. The prompt is 'dasadova@dk3n63 ~/work/arch-pc/lab04 \$'. The user enters './hello', and the output is 'Hello world!'. The prompt is then shown again with a cursor.

```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ ./hello
Hello world!
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```

Рис. 2.13: Запускаем созданный исполняемый файл

Проверя работу программы, мы получили надпись “Hello world!” записанную в текстовый и далее преобразованный файл.

### **3 Теоретическое введение**

## 4 Выполнение лабораторной работы

### 4.1 Задание для самостоятельной работы

4.1.1 В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создайте копию файла `hello.asm` с именем `lab4.asm` (рис.4.1)

```
dasadova@dk3n63 ~/work/arch-pc $ cp lab04/hello.asm lab04/lab4.asm
dasadova@dk3n63 ~/work/arch-pc $ cd lab04
dasadova@dk3n63 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```

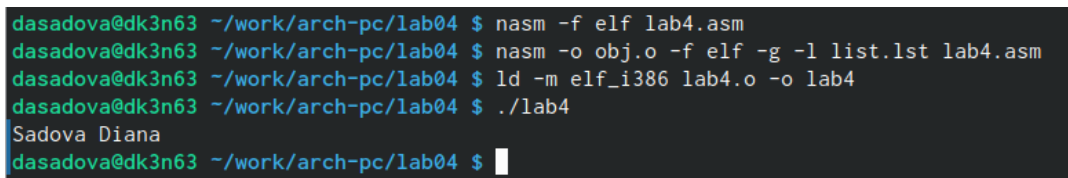
Рис. 4.1: Создаем копию `hello.asm` с именем `lab4.asm`

**4.1.2 С помощью любого текстового редактора внесите изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! На экран выводилась строка с вашими фамилией и именем (рис.4.2)(рис.4.3)**



```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ gedit lab4.asm
lab4.asm
~/work/arch-pc/lab04
Открыть Сохранить
1 SECTION .data
2 hello: DB 'Sadova Diana',10
3
4 helloLen: EQU $-hello
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9 mov eax,4
10 mov ebx,1
11 mov ecx,hello
12 mov edx,helloLen
13 int 80h
14 mov eax,1
15 mov ebx,0
16 int 80h
```

Рис. 4.2: Открываем с помощью текстового редактора и вносим изменения



```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
dasadova@dk3n63 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst lab4.asm
dasadova@dk3n63 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
dasadova@dk3n63 ~/work/arch-pc/lab04 $ ./lab4
Sadova Diana
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```

Рис. 4.3: Проверяем, что изменения внесены верно

### 4.1.3 Оттранслируйте полученный текст программы lab4.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл (рис.4.4)

```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
dasadova@dk3n63 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst lab4.asm
dasadova@dk3n63 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o lab4
dasadova@dk3n63 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
dasadova@dk3n63 ~/work/arch-pc/lab04 $
```

Рис. 4.4: Оттранслируем полученный текст в объектный файл

### 4.1.4 Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/. Загрузите файлы на Github (рис.4.5)(рис.4.6)

```
dasadova@dk3n63 ~/work/arch-pc/lab04 $ cp hello.asm ~/work/study/2023-2024/“Архитектура компьютера”/Diana-study_2023-2024_arh--pc/labs/lab04
dasadova@dk3n63 ~/work/arch-pc/lab04 $ cp lab4.asm ~/work/study/2023-2024/“Архитектура компьютера”/Diana-study_2023-2024_arh--pc/labs/lab04
dasadova@dk3n63 ~/work/study/2023-2024/“Архитектура компьютера”/Diana-study_2023-2024_arh--pc/labs/lab04 $ cd ~/work/study/2023-2024/“Архитектура компьютера”/Diana-study_2023-2024_arh--pc/labs/lab04
dasadova@dk3n63 ~/work/study/2023-2024/Архитектура компьютера/Diana-study_2023-2024_arh--pc/labs/lab04 $ ls
hello.asm lab4.asm presentation report
dasadova@dk3n63 ~/work/study/2023-2024/Архитектура компьютера/Diana-study_2023-2024_arh--pc/labs/lab04 $
```

Рис. 4.5: Копируем файлы hello.asm и lab4.asm в локальный репозиторий

```
dasadova@dk8n75 ~/work/study/2023-2024/Архитектура компьютера/Diana-study_2023-2024_arh--pc/labs/lab04/report $ git add .
dasadova@dk8n75 ~/work/study/2023-2024/Архитектура компьютера/Diana-study_2023-2024_arh--pc/labs/lab04/report $ git commit -am 'feat
t(main): add files lab-4'
[master 44295a2] feat(main): add files lab-4
5 files changed, 1 insertion(+), 2 deletions(-)
delete mode 100644 labs/lab04/report/.~lock.report.docx#
create mode 100644 labs/lab04/report/image.zip
dasadova@dk8n75 ~/work/study/2023-2024/Архитектура компьютера/Diana-study_2023-2024_arh--pc/labs/lab04/report $ git push
Перечисление объектов: 66, готово.
Подсчет объектов: 100% (65/65), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (58/58), готово.
Запись объектов: 100% (58/58), 2.01 МБ | 2.70 МБ/с, готово.
Всего 58 (изменений 17), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (17/17), completed with 3 local objects.
To github.com:DianaSadova/Diana-study_2023-2024_arh--pc.git
57b3fia..44295a2 master -> master
dasadova@dk8n75 ~/work/study/2023-2024/Архитектура компьютера/Diana-study_2023-2024_arh--pc/labs/lab04/report $
```

Рис. 4.6: Загружаем файл на Github

## **5 Выводы**

Освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.



## **Список литературы**