

# **Лабораторная работа №4**

**Создание и процесс обработки программ на языке ассемблера NASM**

Чекмарев Александр Дмитриевич | группа: НПИбд 02-23

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Самостоятельна работа</b>	<b>11</b>
<b>4</b>	<b>Выводы</b>	<b>15</b>

## Список иллюстраций

2.1	Рис 2.1.1: Создание каталога /work/arch-pc/lab04 . . . . .	6
2.2	Рис 2.1.2: Переход в каталог с помощью команды cd . . . . .	6
2.3	Рис 2.1.3: Создание текстового файла с помощью команды touch .	6
2.4	Рис 2.1.4: Откроем файл с помощью текстового редактора gedit .	7
2.5	Рис 2.1.5: Демонстрация текста в файле . . . . .	7
2.6	Рис 2.2.1: Компиляция текста с помощью команды nasm -f elf hello.asm . . . . .	7
2.7	Рис 2.2.2: Проверка созданного файла . . . . .	8
2.8	Рис 2.3.1: Компиляция исходного файла hello.asm в obj.o . . . . .	8
2.9	Рис 2.3.2: Проверка созданных файлов . . . . .	9
2.10	Рис 2.4.1: Передача объектного файла на обработку компоновщику	9
2.11	Рис 2.4.2: Проверка созданного файла hello . . . . .	9
2.12	Рис 2.4.3: Создание исполняемого файла main и его проверка . .	10
2.13	Рис 2.5.1: Запуск исполняемого файла hello с помощью команды ./hello . . . . .	10
3.1	Рис 3.1.1: Копирование файла . . . . .	11
3.2	Рис 3.2.1: Применение команды gedit . . . . .	11
3.3	Рис 3.2.2: Демонстрация изменённого текста . . . . .	12
3.4	Рис 3.3.1: Компиляция файла . . . . .	12
3.5	Рис 3.3.2: Передача объектного файла на обработку компоновщику	12
3.6	Рис 3.3.3: Запуск исполняемого файла LF с помощью команды ./LF	13
3.7	Рис 3.4.1: Копирование файлов hello.asm и lab4.asm . . . . .	13
3.8	Рис 3.4.2: Проверка . . . . .	13
3.9	Рис 3.4.3: (1)Загрузка файлов на гитхаб . . . . .	13
3.10	Рис 3.4.4: (2)Загрузка файлов на гитхаб . . . . .	14

## Список таблиц

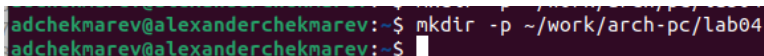
# 1 Цель работы

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

## 2 Выполнение лабораторной работы

### 2.1 Программа Hello world!

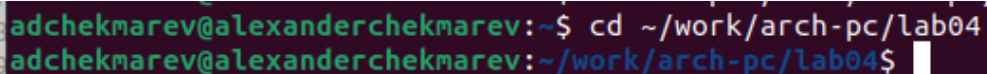
Рассмотрим пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит приветственное сообщение Hello world! на экран. Создадим каталог для работы с программами на языке ассемблера NASM:



```
adchekmarev@alexanderchekmarev:~$ mkdir -p ~/work/arch-pc/lab04
adchekmarev@alexanderchekmarev:~$
```

Рис. 2.1: Рис 2.1.1: Создание каталога /work/arch-pc/lab04

Перейдём в созданный каталог



```
adchekmarev@alexanderchekmarev:~$ cd ~/work/arch-pc/lab04
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$
```

Рис. 2.2: Рис 2.1.2: Переход в каталог с помощью команды cd

Создадим текстовый файл с именем **hello.asm**



```
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$ touch hello.asm
```

Рис. 2.3: Рис 2.1.3: Создание текстового файла с помощью команды touch

Откроем этот файл с помощью любого текстового редактора, например, gedit

```

alexanderchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$ gedit hello.asm
alexanderchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$

```

Рис. 2.4: Рис 2.1.4: Откроем файл с помощью текстового редактора gedit

Введём в нём следующий текст:

report.md	hello.asm
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	
7 SECTION .text	; Начало секции кода
8     GLOBAL _start	
9	
10 _start:	; Точка входа в программу
11     mov eax,4	; Системный вызов для записи (sys_write)
12     mov ebx,1	; Описатель файла '1' - стандартный вывод
13     mov ecx,hello	; Адрес строки hello в есх
14     mov edx,helloLen	; Размер строки hello
15     int 80h	; Вызов ядра
16	
17     mov eax,1	; Системный вызов для выхода (sys_exit)
18     mov ebx,0	; Выход с кодом возврата '0' (без ошибок)
19     int 80h	; Вызов ядра

Рис. 2.5: Рис 2.1.5: Демонстрация текста в файле

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.

## 2.2 Транслятор NASM

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

```

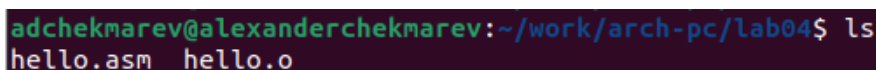
alexanderchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$ nasm -f elf hello.asm

```

Рис. 2.6: Рис 2.2.1: Компиляция текста с помощью команды nasm -f elf hello.asm

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` проверим, что объектный файл был создан:



```
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
```

Рис. 2.7: Рис 2.2.2: Проверка созданного файла

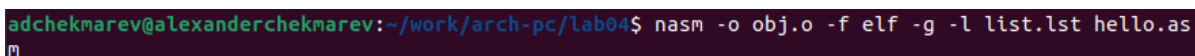
NASM не запускают без параметров. Ключ `-f` указывает транслятору, что требуется создать бинарные файлы в формате **ELF**. Следует отметить, что формат **elf64** позволяет создавать исполняемый код, работающий под 64-битными версиями Linux. Для 32-битных версий ОС указываем в качестве формата просто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

### 2.3 Расширенный синтаксис командной строки NASM

Полный вариант командной строки `nasm` выглядит следующим образом:

`nasm [-@ косвенный_файл_настроек] [-o объектный_файл] [-f формат_объектного_файла] [-l листинг] [параметры...] [-] исходный_файл`

Выполним следующую команду:



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

Рис. 2.8: Рис 2.3.1: Компиляция исходного файла `hello.asm` в `obj.o`

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* проверим, что файлы были созданы:

```
adchekmarev@alexandercchekmarev:~/work/arch-pc/lab04$ ls
hello.asm hello.o list.lst obj.o
```

Рис. 2.9: Рис 2.3.2: Проверка созданных файлов

Для более подробной информации см. *man nasm*. Для получения списка форматов объектного файла см. *nasm -hf*.

## 2.4 Компоновщик LD

Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику:

```
adchekmarev@alexandercchekmarev:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
```

Рис. 2.10: Рис 2.4.1: Передача объектного файла на обработку компоновщику

С помощью команды *ls* проверим, что исполняемый файл *hello* был создан:

```
adchekmarev@alexandercchekmarev:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst obj.o
```

Рис. 2.11: Рис 2.4.2: Проверка созданного файла *hello*

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.

Выполним следующую команду:

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

Рис. 2.12: Рис 2.4.3: Создание исполняемого файла main и его проверка

Объектный файл obj.o был передан на обработку компоновщику для создания исполняемого файла main.

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

## 2.5 Запуск исполняемого файла

Запустить на выполнение созданный исполняемый файл, находящийся в текущем каталоге, можно, набрав в командной строке:

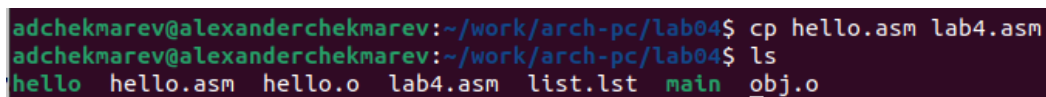
```
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$ ./hello
Hello world!
```

Рис. 2.13: Рис 2.5.1: Запуск исполняемого файла hello с помощью команды ./hello

### 3 Самостоятельная работа

*Задание№1 В каталоге ~/work/arch-pc/lab04 с помощью команды cp создайте копию файла hello.asm с именем lab4.asm*

Создадим копию файла hello.asm с именем lab4.asm

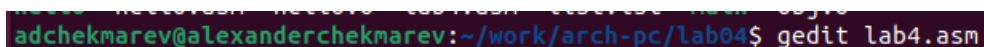
A terminal window with a dark purple background. The prompt is 'adcchekmarev@alexanderchekmarev:~/work/arch-pc/lab04\$'. The first command is 'cp hello.asm lab4.asm'. The second command is 'ls', which outputs 'hello hello.asm hello.o lab4.asm list.lst main obj.o'.

```
adcchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
adcchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm list.lst main obj.o
```

Рис. 3.1: Рис 3.1.1: Копирование файла

*Задание№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.*

С помощью редактора markdown внесём изменения в текст в файле lab4.asm

A terminal window with a dark purple background. The prompt is 'adcchekmarev@alexanderchekmarev:~/work/arch-pc/lab04\$'. The command 'gedit lab4.asm' is entered.

```
adcchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$ gedit lab4.asm
```

Рис. 3.2: Рис 3.2.1: Применение команды gedit

```

1 ; lab4.asm
2 SECTION .data                                ; Начало секции данных
3     lab4: DB 'Чекмарев Александр',10        ; 'Чекмарев Александр' плюс
4                                             ; символ перевода строки
5     lab4Len: EQU $-lab4                     ; Длина строки lab4
6
7 SECTION .text                                ; Начало секции кода
8     GLOBAL _start
9
10 _start:                                     ; Точка входа в программу
11     mov eax,4                               ; Системный вызов для записи (sys_write)
12     mov ebx,1                               ; Описатель файла '1' - стандартный вывод
13     mov ecx,lab4                            ; Адрес строки lab4 в ecx
14     mov edx,lab4Len                        ; Размер строки lab4
15     int 80h                                ; Вызов ядра
16
17     mov eax,1                               ; Системный вызов для выхода (sys_exit)
18     mov ebx,0                               ; Выход с кодом возврата '0' (без ошибок)
19     int 80h                                ; Вызов ядра

```

Рис. 3.3: Рис 3.2.2: Демонстрация изменённого текста

*Задание №3 Оттранслируйте полученный текст программы lab4.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл.*

Скомпилируем файл lab4.asm

```

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

```

Рис. 3.4: Рис 3.3.1: Компиляция файла

Передадим объектный файл lab4.o на обработку компоновщику

```

adchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o LF
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm lab4.o LF list.lst main obj.o

```

Рис. 3.5: Рис 3.3.2: Передача объектного файла на обработку компоновщику

Запустим получившийся исполняемый файл LF

```
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$ ./LF
Чекмарев Александр
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$
```

Рис. 3.6: Рис 3.3.3: Запуск исполняемого файла LF с помощью команды ./LF

*Задание №4* Скопируйте файлы *hello.asm* и *lab4.asm* в Ваш локальный репозиторий в каталог `~/work/study2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/`. Загрузите файлы на Github.

Скопируем файлы в локальный репозиторий

```
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$ cp {hello.asm,lab4.asm} /home/adchekmarev/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab04$
```

Рис. 3.7: Рис 3.4.1: Копирование файлов *hello.asm* и *lab4.asm*

```
adchekmarev@alexanderchekmarev:~$ cd ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04
adchekmarev@alexanderchekmarev:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm lab4.asm presentation report
adchekmarev@alexanderchekmarev:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$
```

Рис. 3.8: Рис 3.4.2: Проверка

Загрузим файлы на Github

```
adchekmarev@alexanderchekmarev:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report$ git add .
adchekmarev@alexanderchekmarev:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report$ git commit -am 'feat(main): make course structure'
[master dec3df6] feat(main): make course structure
25 files changed, 228 insertions(+), 119 deletions(-)
delete mode 100644 labs/lab04/report/image/placeimg_800_600_tech.jpg
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.1.1.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.1.2.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.1.3.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.1.4.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.1.5.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.2.1.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.2.2.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.3.1.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.3.2.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.4.1.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.4.2.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.4.3.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 2.5.1.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 3.1.1.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 3.2.1.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 3.2.2.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 3.3.1.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 3.3.2.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 3.3.3.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 3.4.1.png"
create mode 100644 "labs/lab04/report/image/\320\240\320\270\321\201 3.4.2.png"
create mode 100644 labs/lab04/report/report.docx
rewrite labs/lab04/report/report.md (71%)
create mode 100644 labs/lab04/report/report.pdf
```

Рис. 3.9: Рис 3.4.3: (1)Загрузка файлов на гитхаб

```
alexanderchekmarev@alexanderchekmarev:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report$ gedit report.md
alexanderchekmarev@alexanderchekmarev:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report$ git push
Перечисление объектов: 58, готово.
Подсчет объектов: 100% (58/58), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (48/48), готово.
Запись объектов: 100% (48/48), 1.55 МиБ | 2.41 МиБ/с, готово.
Всего 48 (изменений 14), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (14/14), completed with 6 local objects.
To github.com:nenokxd/study_2023-2024_arh-pc.git
   e0ba18f..c291605  master -> master
```

Рис. 3.10: Рис 3.4.4: (2)Загрузка файлов на гитхаб

## 4 Выводы

Я освоил процедуры компиляции и сборки программ, написанных на ассемблере NASM.