

# **Отчёт по лабораторной работе №4**

**дисциплина: Архитектура компьютера**

Байдина Елизавета Дмитриевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Создание программы Hello world! . . . . .	7
3.2	Работа с транслятором NASM . . . . .	8
3.3	Работа с расширенным синтаксисом командной строки NASM . . .	8
3.4	Работа с компоновщиком LD . . . . .	9
3.5	Запуск исполняемого файла . . . . .	9
3.6	Выполнение заданий для самостоятельной работы . . . . .	10
<b>4</b>	<b>Выводы</b>	<b>12</b>
	<b>Список литературы</b>	<b>13</b>

# Список иллюстраций

3.1	Перемещение между директориями . . . . .	7
3.2	Создание пустого файла . . . . .	7
3.3	Открытие файла в текстовом редакторе . . . . .	7
3.4	Заполнение файла . . . . .	8
3.5	Компиляция текста программы.png . . . . .	8
3.6	Компиляция текста программы.png . . . . .	9
3.7	Передача объектного файла на обработку компоновщику . . . . .	9
3.8	Передача объектного файла на обработку компоновщику . . . . .	9
3.9	Запуск исполняемого файла . . . . .	9
3.10	Создание копии файла . . . . .	10
3.11	Изменение программы . . . . .	10
3.12	Компиляция текста программы . . . . .	10
3.13	Передача объектного файла на обработку компоновщику . . . . .	10
3.14	Запуск исполняемого файла . . . . .	11
3.15	Создание копии файлов в новом каталоге . . . . .	11
3.16	Добавление файлов на GitHub . . . . .	11
3.17	Отправка файлов . . . . .	11

# Список таблиц

2.1	Описание некоторых каталогов файловой системы GNU Linux . . .	6
-----	---------------------------------------------------------------	---

# 1 Цель работы

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

## 2 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 2.1 приведено краткое описание стандартных каталогов Unix.

Таблица 2.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую систему
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно про Unix см. в [1–4].

## 3 Выполнение лабораторной работы

### 3.1 Создание программы Hello world!

С помощью команды `cd` переходим в каталог, в котором будем работать. (рис. 3.1).

```
edbayjdina@dk3n38 ~ $ mkdir -p ~/work/arch-pc/lab04
edbayjdina@dk3n38 ~ $ cd ~/work/arch-pc/lab04
```

Рис. 3.1: Перемещение между директориями

В текущем каталоге создаю пустой текстовый файл `hello.asm` с помощью утилиты `touch` (рис. 3.2).

```
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ touch hello.asm
```

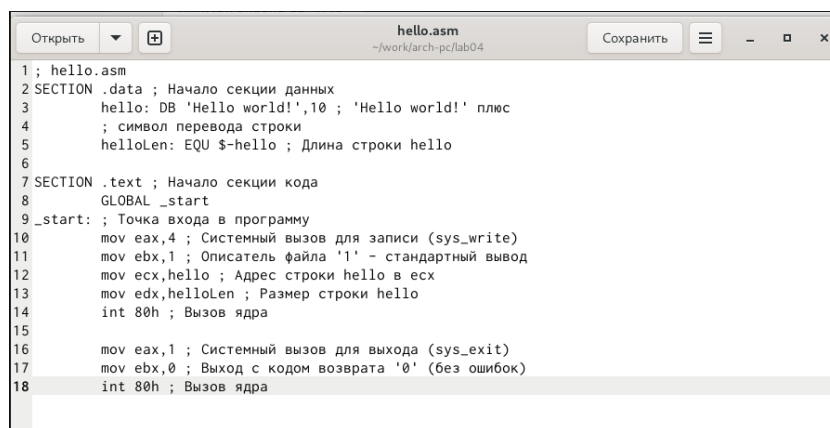
Рис. 3.2: Создание пустого файла

Открываю созданный файл в текстовом редакторе. (рис. 3.3).

```
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 3.3: Открытие файла в текстовом редакторе

Заполняю файл, вставляя в него программу для вывода “Hello word!” (рис. 3.4).



```
1; hello.asm
2SECTION .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
7SECTION .text ; Начало секции кода
8    GLOBAL _start
9_start: ; Точка входа в программу
10       mov eax,4 ; Системный вызов для записи (sys_write)
11       mov ebx,1 ; Описатель файла '1' - стандартный вывод
12       mov ecx,hello ; Адрес строки hello в ecx
13       mov edx,helloLen ; Размер строки hello
14       int 80h ; Вызов ядра
15
16       mov eax,1 ; Системный вызов для выхода (sys_exit)
17       mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
18       int 80h ; Вызов ядра
```

Рис. 3.4: Заполнение файла

## 3.2 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. Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “hello.o”(рис. 3.5).

```
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ ls
hello.asm hello.o list.lst obj.o
```

Рис. 3.5: Компиляция текста программы.png

## 3.3 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.(рис. 3.6).



```
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o
```

Рис. 3.6: Компиляция текста программы.png

### 3.4 Работа с компоновщиком 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. (рис. 3.7).

```
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o
```

Рис. 3.7: Передача объектного файла на обработку компоновщику

Выполняю следующую команду. 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. 3.8).

```
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
```

Рис. 3.8: Передача объектного файла на обработку компоновщику

### 3.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. 3.9).

```
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ ./hello
Hello world!
```

Рис. 3.9: Запуск исполняемого файла

## 3.6 Выполнение заданий для самостоятельной работы

С помощью утилиты `cp` создаю в текущем каталоге копию файла `hello.asm` с именем `lab4.asm` (рис. 3.10).

```
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $
```

Рис. 3.10: 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 (рис. 3.11).

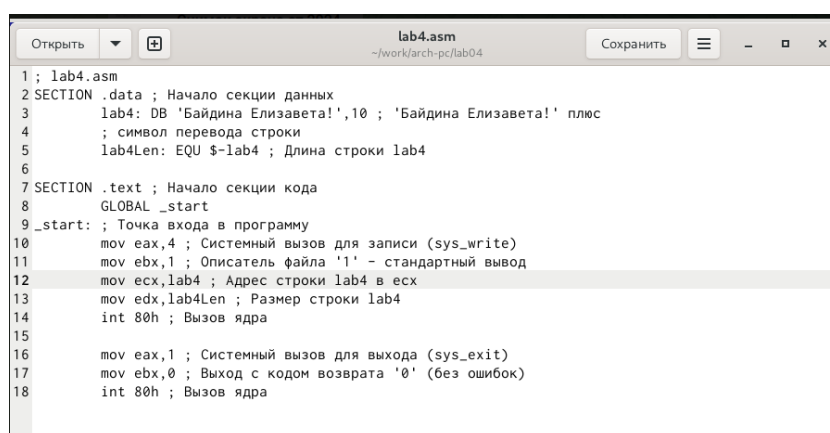


Рис. 3.11: Изменение программы

Компилирую текст программы в объектный файл. Проверяю с помощью утилиты `ls`, что файл `lab4.o` создан (рис. 3.12).

```
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o
```

Рис. 3.12: Компиляция текста программы

Передаю объектный файл `lab4.o` на обработку компоновщику `LD`, чтобы получить исполняемый файл `lab4` (рис. 3.13).

```
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o
```

Рис. 3.13: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab5, на экран действительно выводятся мои имя и фамилия (рис. 3.14).

```
edbayjdina@dk8n64 ~/work/arch-pc/lab04 $ ./lab4
Байдина Елизавета!
```

Рис. 3.14: Запуск исполняемого файла

Создаю копии файлов в новом каталоге(рис. 3.15).

```
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ cp lab4.asm ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/lab04
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ cp hello.asm ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/lab04
edbayjdina@dk3n38 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
```

Рис. 3.15: Создание копии файлов в новом каталоге

С помощью команд `git add .` и `git commit` добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №4 (рис. 3.16).

```
edbayjdina@dk3n38 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git add .
edbayjdina@dk3n38 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git commit -m "Add fales for lab04"
[master b7f2245] Add fales for lab04
1 file changed, 18 insertions(+)
create mode 100644 lab04
```

Рис. 3.16: Добавление файлов на GitHub

Отправляю файлы на сервер с помощью команды `git push` (рис. 3.17).

```
edbayjdina@dk3n38 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git push
Перечисление объектов: 4, готово.
Подсчет объектов: 100% (4/4), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 715 байтов | 715.00 КиБ/с, готово.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:edbayjdina/edbayjdina-study_2023-2024_arh-pc.git
 8b9e6ce..b7f2245 master -> master
```

Рис. 3.17: Отправка файлов

## **4 Выводы**

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

## Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.
2. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.