

# **Отчет по лабораторной работе №4**

**дисциплина: Архитектура компьютера**

Михайлова Регина Алексеевна

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение заданий для самостоятельной работы	11
4	Вывод	14
5	Список литературы	15

# Список иллюстраций

2.1	Создание каталога . . . . .	6
2.2	Созданный каталог . . . . .	6
2.3	Переход в созданный каталог . . . . .	7
2.4	Создание файла hello.asm . . . . .	7
2.5	Созданный файл hello.asm . . . . .	7
2.6	Открытие файла hello.asm . . . . .	7
2.7	Компиляция программы «Hello World» . . . . .	8
2.8	hello.o успешно создан . . . . .	9
2.9	Компиляция hello.asm в obj.o . . . . .	9
2.10	obj.o и list.lst успешно созданы . . . . .	9
2.11	Передача объектного файла компоновщику . . . . .	10
2.12	Создание файла main . . . . .	10
2.13	Исполняемый файл успешно выполнен . . . . .	10
3.1	Копирование файла . . . . .	11
3.2	Копия файла создана . . . . .	11
3.3	Преобразование в объектный файл . . . . .	12
3.4	Компоновка объектного файла . . . . .	12
3.5	Запуск исполняемого файла . . . . .	13

## Список таблиц

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

1. 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 (рис. 2.1, 2.2): `mkdir -p ~/work/arch-pc/lab04`

```
ramikhalova@ramikhailova:~$ mkdir -p ~/work/arch-pc/lab04
ramikhalova@ramikhailova:~$
```

Рис. 2.1: Создание каталога

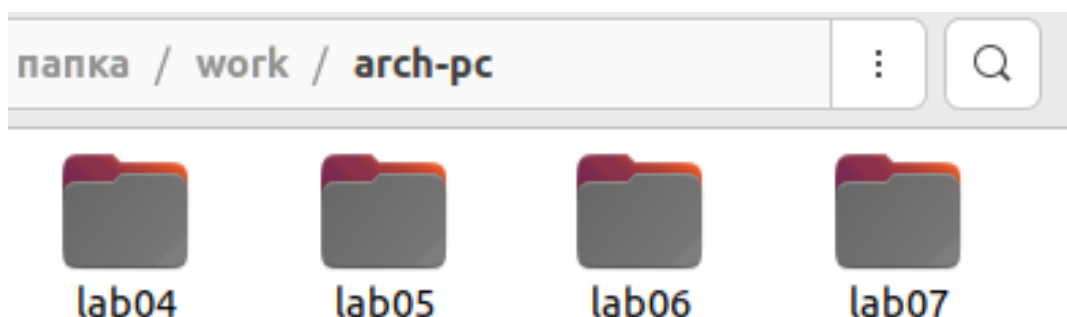


Рис. 2.2: Созданный каталог

Перейдем в созданный каталог (рис. 2.3): `cd ~/work/arch-pc/lab04`

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

Рис. 2.3: Переход в созданный каталог

Создадим текстовый файл с именем hello.asm (рис. 2.4, 2.5): touch hello.asm

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

Рис. 2.4: Создание файла hello.asm

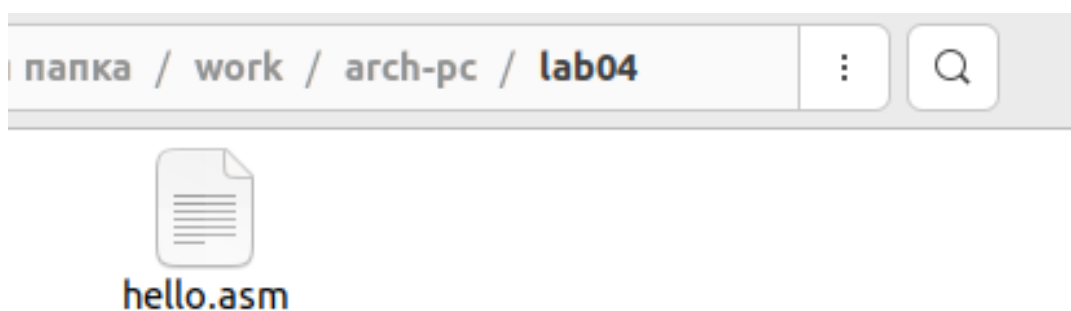


Рис. 2.5: Созданный файл hello.asm

Откроем этот файл с помощью любого текстового редактора, например, gedit (рис. 2.6): gedit hello.asm

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

Рис. 2.6: Открытие файла hello.asm

и введем в него следующий текст:

```
; hello.asm
SECTION .data ; Начало секции данных
```

```

hello: DB 'Hello world!',10 ; 'Hello world!' плюс
; символ перевода строки
helloLen: EQU $-hello ; Длина строки hello
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,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

```

2. NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World» (рис. 2.7) необходимо написать: `nasm -f elf hello.asm`



```

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

```

Рис. 2.7: Компиляция программы «Hello World»

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 (рис. 2.8).



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

Рис. 2.8: hello.o успешно создан

3. 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 (рис. 2.9): `nasm -o obj.o -f elf -g -l list.lst hello.asm`

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

Рис. 2.9: Компиляция 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 (рис. 2.10)

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

Рис. 2.10: obj.o и list.lst успешно созданы

4. Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику (рис. 2.11): `ld -m elf_i386 hello.o -o hello` С помощью команды `ls` проверим, что исполняемый файл `hello` был создан (рис. 2.11).

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

Рис. 2.11: Передача объектного файла компоновщику

Выполним следующую команду (рис. 2.12): `ld -m elf_i386 obj.o -o main`  
Исполняемый файл будет иметь имя `main` (рис. 2.12)

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

Рис. 2.12: Создание файла `main`

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

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

Рис. 2.13: Исполняемый файл успешно выполнен

### 3 Выполнение заданий для самостоятельной работы

1. В каталоге ~/work/arch-pc/lab04 с помощью команды cp создадим копию файла hello.asm с именем lab4.asm (рис. 3.1), после чего копия файла будет успешно создана (рис. 3.2)

```
ramikhalova@ramikhalova:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report$ cd ~/work/arch-pc/lab04
ramikhalova@ramikhalova:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
```

Рис. 3.1: Копирование файла

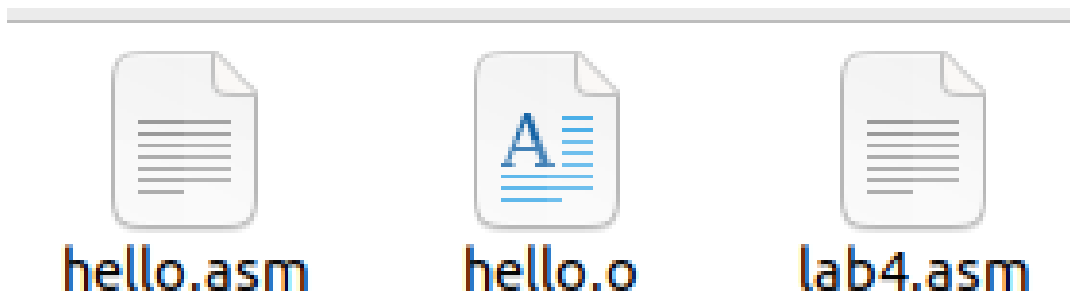


Рис. 3.2: Копия файла создана


2. С помощью текстового редактора внесем изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с фамилией и именем:

```
SECTION .data
name: DB 'Mikhaylova Regina',10
nameLen: EQU $-name
```

```
SECTION .text
GLOBAL _start
```

```
_start:
mov eax,4
mov ebx,1
mov ecx,name
mov edx,nameLen
int 80h
mov eax,1
mov ebx,0
int 80h
```

3. Оттранслируем полученный текст программы lab4.asm в объектный файл (рис. 3.3). Выполним компоновку объектного файла (рис. 3.4) и запустим получившийся исполняемый файл (рис. 3.5).



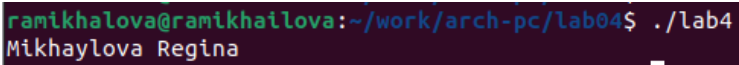
```
ramikhalova@ramikhailova:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
```

Рис. 3.3: Преобразование в объектный файл



```
ramikhalova@ramikhailova:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
```

Рис. 3.4: Компоновка объектного файла

A terminal window with a dark background. The prompt is 'ramikhalova@ramikhailova:~/work/arch-pc/lab04\$'. The command './lab4' has been entered. Below the command, the text 'Mikhaylova Regina' is displayed.

```
ramikhalova@ramikhailova:~/work/arch-pc/lab04$ ./lab4
Mikhaylova Regina
```

Рис. 3.5: Запуск исполняемого файла

## 4 Вывод

В ходе лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

## 5 Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-

- е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).