

Лабораторная работа №4

Седохин Даниил Алексеевич

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Задание для самостоятельной работы	11
4	Выводы	13

Список иллюстраций

2.1	Создание каталога для работы с программами на языке ассемблера NASM	5
2.2	Переходим в созданный каталог	5
2.3	Создаём текстовый файл	5
2.4	Открываем файл	6
2.5	Вводим текст	6
2.6	Компилирование текста программы “Hello World”	7
2.7	Компилируем исходный файл hello.asm в obj.o	7
2.8	Получение списка форматов объектного файла	8
2.9	Получение исполняемой программы	8
2.10	Проверка созданного файла	9
2.11	Задаём имя создаваемого исполняемого файла	9
2.12	Формат командной строки	9
2.13	Получение более подробной информации с помощью команды map ld.	10
2.14	Запуск созданного исполняемого файла	10
3.1	Создание копии файла hello.asm с именем lab4.asm	11
3.2	Изменение файла lab4.asm на вывод вместо Hello World - ФИ . . .	11
3.3	Копирование файлов в локальный репозиторий	12
3.4	Получение списка форматов объектного файла	12

1 Цель работы

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

2 Выполнение лабораторной работы

- 1) Создадим каталог для работы с программами на языке ассемблера NASM с помощью следующей команды. (рис. 2.1).

```
dasedokhin@dasedokhin-VirtualBox:~$ mkdir -p ~/work/arch-pc/lab04  
dasedokhin@dasedokhin-VirtualBox:~$ cd ~/work/arch-pc/lab04
```

Рис. 2.1: Создание каталога для работы с программами на языке ассемблера NASM

- 2) Перейдём в созданный каталог. (рис. 2.2).

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

Рис. 2.2: Переходим в созданный каталог

- 3) Создадим текстовый файл с именем “hello.asm”. (рис. 2.3).

```
dasedokhin@dasedokhin-VirtualBox:~$ cd ~/work/arch-pc/lab04  
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$ touch hello.asm  
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$
```

Рис. 2.3: Создаём текстовый файл

- 4) Откройте этот файл с помощью любого текстового редактора, например, gedit (рис. 2.4).

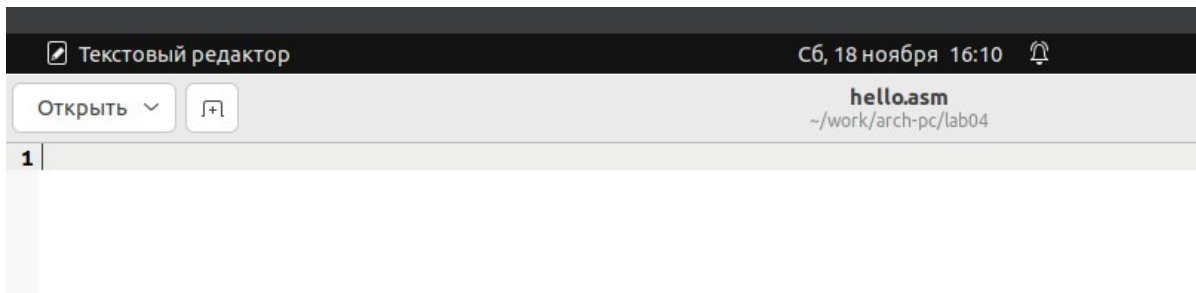


Рис. 2.4: Открываем файл

- 5) Введём в него следующий текст: (рис. 2.5).

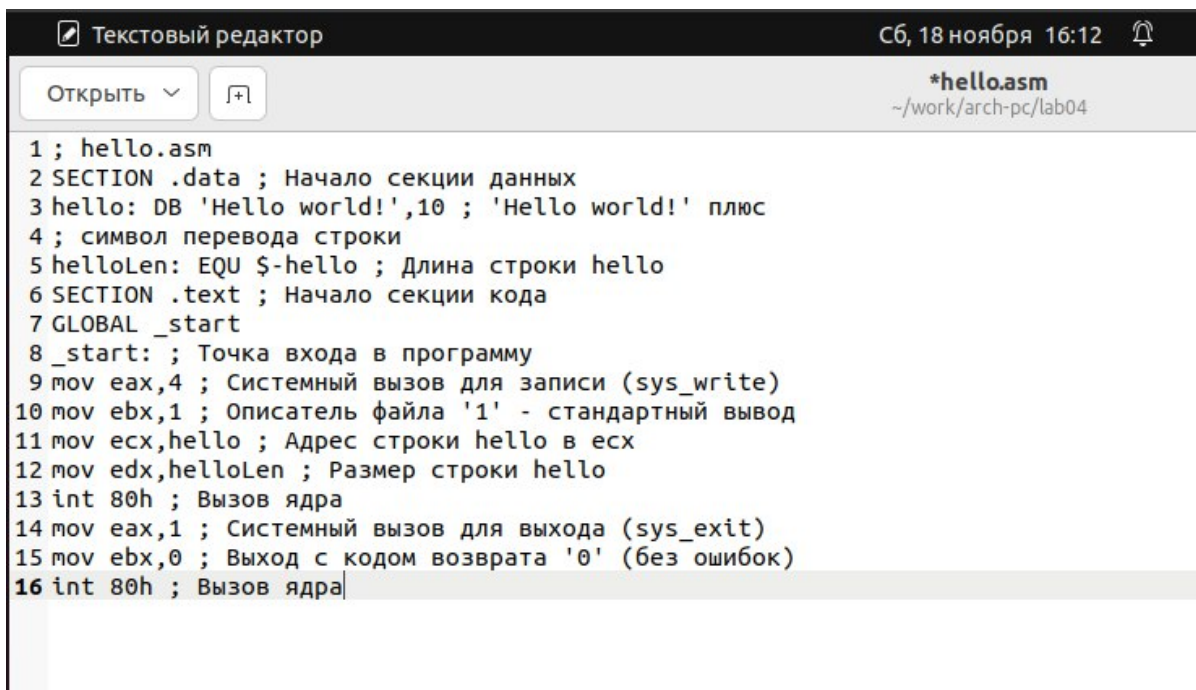


Рис. 2.5: Вводим текст

- 6) Для компиляции приведённого выше текста программы «Hello World» на-пишем следующую команду и проверим с помощью “ls” (рис. 2.6).

```
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$ nasm -f elf hello.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
```

Рис. 2.6: Компилирование текста программы “Hello World”

- 7) Выполним следующую команду. 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`). Проверим созданные файлы с помощью команды `“ls”`. (рис. 2.7).

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

Рис. 2.7: Компилируем исходный файл `hello.asm` в `obj.o`

- 8) Для получения списка форматов объектного файла см. `nasm -hf`. (рис. 2.8).

```

dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$ nasm -hf
Usage: nasm [-@ response_file] [options...] [--] filename
        nasm -v (or --v)

Options (values in brackets indicate defaults):

  -h                show this text and exit (also --help)
  -v (or --v)       print the NASM version number and exit
  -@ file           response file; one command line option per line

  -o outfile        write output to outfile
  --keep-all        output files will not be removed even if an error happens

  -Xformat          specify error reporting format (gnu or vc)
  -s                redirect error messages to stdout
  -Zfile            redirect error messages to file

  -M                generate Makefile dependencies on stdout
  -MG               d:o, missing files assumed generated
  -MF file          set Makefile dependency file
  -MD file          assemble and generate dependencies
  -MT file          dependency target name
  -MQ file          dependency target name (quoted)
  -MP               emit phony targets

  -f format         select output file format
    bin             Flat raw binary (MS-DOS, embedded, ...) [default]
    ith             Intel Hex encoded flat binary

```

Рис. 2.8: Получение списка форматов объектного файла

- 9) Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику с помощью следующей команды: (рис. 2.9).

```

lines          total source lines processed [2000000000]
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$

```

Рис. 2.9: Получение исполняемой программы

- 10) Проверим созданные файлы с помощью команды “ls”. (рис. 2.10)


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

Рис. 2.10: Проверка созданного файла

- 11) Ключ `-o` с последующим значением задаёт в данном случае имя создаваемого исполняемого файла. Выполним следующую команду: (рис. 2.11).

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

Рис. 2.11: Задаём имя создаваемого исполняемого файла

- 12) Формат командной строки LD можно увидеть, набрав `ld -help`. (рис. 2.12).

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

Рис. 2.12: Формат командной строки

- 13) Для получения более подробной информации см. `man ld`. (рис. 2.13).

```
dasedokhin@dasedokhin-VirtualBox: ~/work/arch-pc/lab04
LD(1) GNU Development Tools

NAME
    ld - The GNU linker

SYNOPSIS
    ld [options] objfile ...

DESCRIPTION
    ld combines a number of object and archive files, relocates their data and ties up symbol references. One of the first steps in compiling a program is to run ld.

    ld accepts Linker Command Language files written in a superset of AT&T's Link Editor Command Language. It gives the user explicit and total control over the linking process.

    This man page does not describe the command language; see the ld entry in "info" for full details and on other aspects of the GNU linker.

    This version of ld uses the general purpose BFD libraries to operate on object files. This allows ld to read and write object files in many different formats---for example, COFF or "a.out". Different formats are used together to produce any available kind of object file.

    Aside from its flexibility, the GNU linker is more helpful than other linkers in providing diagnostics. Unlike other linkers, ld abandons execution immediately upon encountering an error; whenever possible, ld continues to identify other errors (or, in some cases, to get an output file in spite of the error).

    The GNU linker ld is meant to cover a broad range of situations, and to be as compatible as possible with other linkers. As a result, you have many choices to control its behavior.

OPTIONS
    The linker supports a plethora of command-line options, but in actual practice few of them are used in a typical context. For instance, a frequent use of ld is to link standard Unix object files on a stand-alone system, to link a file "hello.o":
```

Рис. 2.13: Получение более подробной информации с помощью команды `man ld`.

- 14) Запустить на выполнение созданный исполняемый файл, находящийся в текущем каталоге, можно, набрав в командной строке: (рис. 2.14).

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

Рис. 2.14: Запуск созданного исполняемого файла

3 Задание 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).

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

Рис. 3.1: Создание копии файла hello.asm с именем lab4.asm

- 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. 3.2).

```
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$ ./lab4.asm
bash: ./lab4.asm: Отказано в доступе
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$ ./lab4
Седохин Даниил!
dasedokhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$
```

Рис. 3.2: Изменение файла lab4.asm на вывод вместо Hello World - ФИ

- 3) Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/. (рис. 3.3).

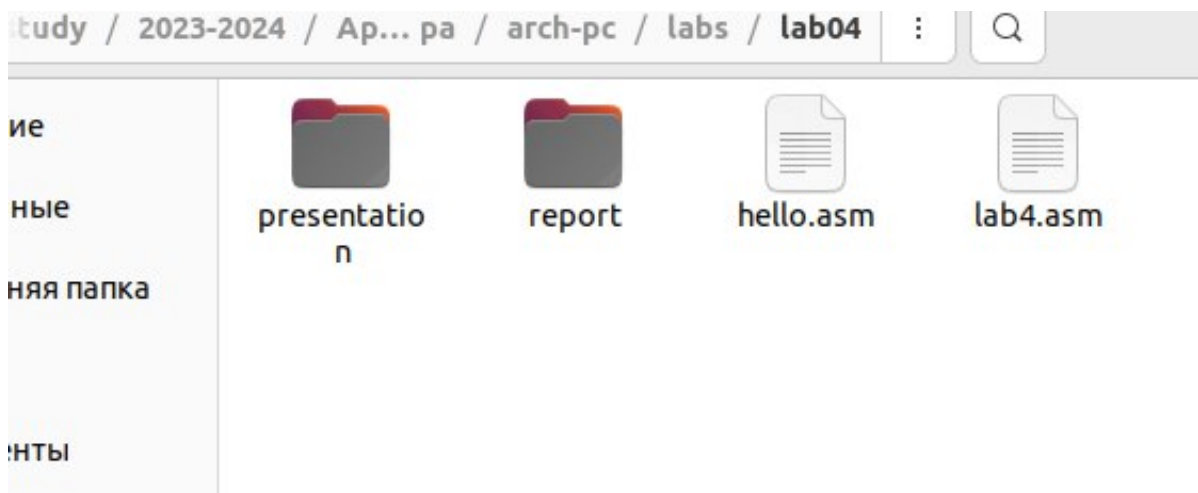


Рис. 3.3: Копирование файлов в локальный репозиторий

4) Загрузим файлы на Github. (рис. ??).

```
okhin@dasedokhin-VirtualBox:~/work/arch-pc/lab04$ cd
okhin@dasedokhin-VirtualBox:~$ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc
okhin@dasedokhin-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git add .
okhin@dasedokhin-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git commit -am 'feat(main):
: pathspec '-' did not match any file(s) known to git
: pathspec 'am' did not match any file(s) known to git
: pathspec 'feat(main): add files lab-4' did not match any file(s) known to git
okhin@dasedokhin-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git commit -am 'feat(main):
er 0d0b661] feat(main): add files lab-4
les changed, 32 insertions(+)
te mode 100644 labs/lab04/hello.asm
te mode 100644 labs/lab04/lab4.asm
okhin@dasedokhin-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git push
исление объектов: 9, готово.
ет объектов: 100% (9/9), готово.
жати изменений используется до 5 потоков
е объектов: 100% (6/6), готово.
ь объектов: 100% (6/6), 968 байтов | 968.00 КиБ/с, готово.
6 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
e: Resolving deltas: 100% (3/3), completed with 2 local objects.
thub.com:Danil2234/study_2023-2024_arh-pc.git
79686...0d0b661 master -> master
```

Рис. 3.4: Получение списка форматов объектного файла

4 Выводы

Мы освоили процедуры компиляции и сборки программ, написанных на ассемблере NASM.