

# Лабораторная работа №4

Создание и процесс обработки программ на языке ассемблера NASM

Замула Егор Сергеевич

# Содержание

Цель работы	5
Теоретическое введение	6
Выполнение лабораторной работы	8
Выводы	13
Список литературы	14

## Список таблиц

## Список иллюстраций

1	Создание каталога . . . . .	8
2	Перешёл в созданный каталог . . . . .	8
3	Создание текстового файла . . . . .	8
4	Открыт файл . . . . .	9
5	Ввожу текст . . . . .	9
6	Компиляция текста . . . . .	9
7	Проверка файла . . . . .	10
8	Компиляция в obj.o . . . . .	10
9	Обработка компоновщика . . . . .	10
10	Задаю имя файла . . . . .	10
11	Запуск файла . . . . .	10
12	Создание копии . . . . .	11
13	Редактирование файла . . . . .	11
14	Выполнение компоновки и запуск файла . . . . .	11
15	Копирование файлов в локальный репозиторий . . . . .	12
16	Загрузка файлов на Гитхаб . . . . .	12
17	Проверка на гитхаб . . . . .	12

## Цель работы

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

# Теоретическое введение

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня, таких как C/C++, Perl, Python и пр. Заметим, что получить полный доступ к ресурсам компьютера в современных архитектурах нельзя, самым низким уровнем работы прикладной программы является обращение напрямую к ядру операционной системы. Именно на этом уровне и работают программы, написанные на ассемблере. Но в отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввёл программист. Таким образом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора. Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — машинные коды. До появления языков ассемблера программистам приходилось писать программы, используя только лишь машинные коды, которые были крайне сложны для запоминания, так как представляли собой числа, записанные в двоичной или шестнадцатеричной системе счисления. Преобразование или трансляция команд с Демидова А. В. 33 Архитектура ЭВМ языка ассемблера в исполняемый машинный код осуществляется специальной программой транслятором — Ассемблер. Программы, написанные на языке ассемблера, не уступают в качестве и скорости программам, написанным на машинном языке, так как транслятор просто переводит мнемонические обозначения команд в последовательности бит (нулей и единиц). Используемые мнемоники обыч-

но одинаковы для всех процессоров одной архитектуры или семейства архитектур (среди широко известных — мнемоники процессоров и контроллеров x86, ARM, SPARC, PowerPC, M68k). Таким образом для каждой архитектуры существует свой ассемблер и, соответственно, свой язык ассемблера. Наиболее распространёнными ассемблерами для архитектуры x86 являются:

- для DOS/Windows: Borland Turbo Assembler (TASM), Microsoft Macro Assembler (MASM) и Watcom assembler (WASM);
- для GNU/Linux: gas (GNU Assembler), использующий AT&T-синтаксис, в отличие от большинства других популярных ассемблеров, которые используют Intel-синтаксис. Более подробно о языке ассемблера см., например, в [10].

В нашем курсе будет использоваться ассемблер NASM (Netwide Assembler) [7; 12; 14]. NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64. Типичный формат записи команд NASM имеет вид: [метка:] мнемокод [операнд {, операнд}] [; комментарий] Здесь мнемокод — непосредственно мнемоника инструкции процессору, которая является обязательной частью команды. Операндами могут быть числа, данные, адреса регистров или адреса оперативной памяти. Метка — это идентификатор, с которым ассемблер ассоциирует некоторое число, чаще всего адрес в памяти. Т.о. метка перед командой связана с адресом данной команды. Допустимыми символами в метках являются буквы, цифры, а также следующие символы: `,`, `$`, `#`, `@`, `~`, `.` и `?`. Начинаться метка или идентификатор могут с буквы, `.` и `?`. Перед идентификаторами, которые пишутся как зарезервированные слова, нужно писать `$`, чтобы компилятор трактовал его верно (так называемое экранирование). Максимальная длина идентификатора 4095 символов. Программа на языке ассемблера также может содержать директивы — инструкции, не переводящиеся непосредственно в машинные команды, а управляющие работой транслятора. Например, директивы используются для определения данных (констант и переменных) и обычно пишутся большими буквами.

# Выполнение лабораторной работы

1. Создал каталог для работы с программами на языке ассемблера NASM:

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

Рис. 1: Создание каталога

2. Перешли в созданный каталог

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

Рис. 2: Перешёл в созданный каталог

3. Создал текстовый файл с именем hello.asm

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

Рис. 3: Создание текстового файла

4. Открыл этот файл с помощью любого текстового редактора, например, gedit



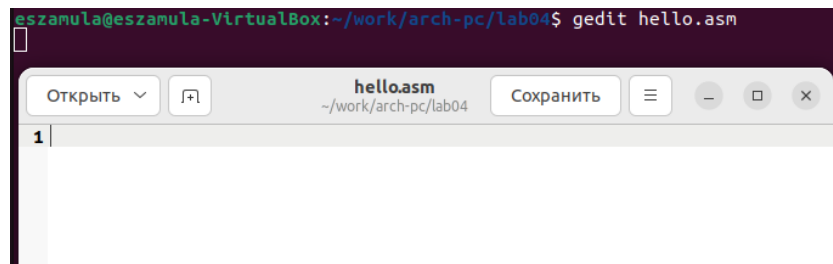


Рис. 4: Открыт файл

5. И ввел в него следующий текст:

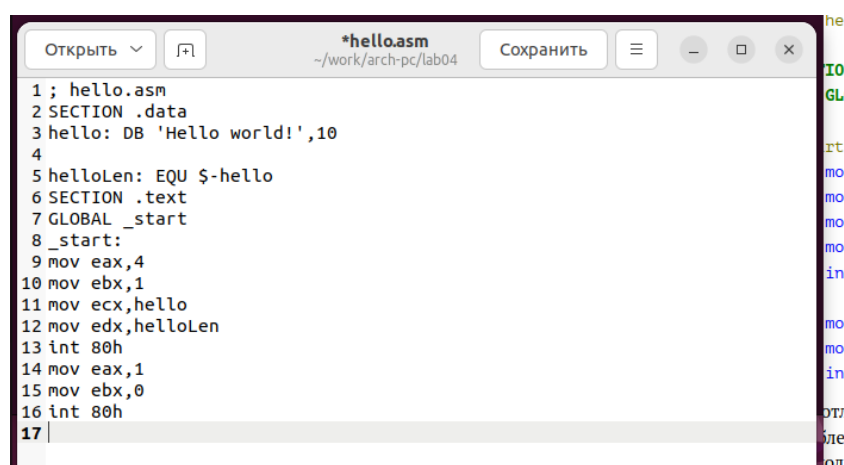


Рис. 5: Ввожу текст

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

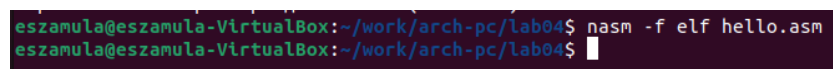


Рис. 6: Компиляция текста

7. Проверяю наличие нового файла

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

Рис. 7: Проверка файла

8. Выполняю следующую команду для компиляции файл hello.asm в obj.o и проверяю его наличие в папки

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

Рис. 8: Компиляция в obj.o

9. Передаю объектный файл на обработку компоновщика и проверяю его в папке

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

Рис. 9: Обработка компоновщика

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

```
eszmula@eszmula-VirtualBox:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
eszmula@eszmula-VirtualBox:~/work/arch-pc/lab04$
```

Рис. 10: Задаю имя файла

11. Запустил на выполнение созданный исполняемый файл, находящийся в текущем каталоге

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

Рис. 11: Запуск файла

12. В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создал копию файла `hello.asm` с именем `lab4.asm`

```
eszamula@eszamula-VirtualBox:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
eszamula@eszamula-VirtualBox:~/work/arch-pc/lab04$
```

Рис. 12: Создание копии

13. С помощью любого текстового редактора внес изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с моей фамилией и именем.



```
eszamula@eszamula-VirtualBox:~/work/arch-pc/lab04$ gedit hello.asm
es
es
-g
Открыть [иконка] *hello.asm Сохранить [иконка] [иконка] [иконка]
~/work/arch-pc/lab04
1 ; hello.asm
2 SECTION .data
3 hello: DB 'Zamula Egor',10
4
5 helloLen: EQU $-hello
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
17
```

Рис. 13: Редактирование файла

14. Оттранслировал полученный текст программы `lab4.asm` в объектный файл. Выполнял компоновку объектного файла и запустил получившийся исполняемый файл

```
eszamula@eszamula-VirtualBox:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
eszamula@eszamula-VirtualBox:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
eszamula@eszamula-VirtualBox:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst lab4.asm
eszamula@eszamula-VirtualBox:~/work/arch-pc/lab04$ nasm -o Zamula.o -f elf -g -l list.lst lab4.asm
eszamula@eszamula-VirtualBox:~/work/arch-pc/lab04$ ld -m elf_i386 Zamula.o -o Zamula
eszamula@eszamula-VirtualBox:~/work/arch-pc/lab04$ ld -m elf_i386 Zamula.o -o Zamula
eszamula@eszamula-VirtualBox:~/work/arch-pc/lab04$ ./Zamula
Zamula Egor
eszamula@eszamula-VirtualBox:~/work/arch-pc/lab04$
```

Рис. 14: Выполнение компоновки и запуск файла

15. Скопировал файлы hello.asm и lab4.asm в локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/ с помощью утилиты cp и проверил наличие файлов с помощью утилиты ls

```
eszanulageszanula-VirtualBox:~/work/arch-pc/lab04$ cp hello.asm ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/report/
eszanulageszanula-VirtualBox:~/work/arch-pc/lab04$ cp lab4.asm ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/report/
eszanulageszanula-VirtualBox:~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/report$ ls
bib hello.asm image lab4.asm Makefile pandoc report.md
eszanulageszanula-VirtualBox:~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/report$
```

Рис. 15: Копирование файлов в локальный репозиторий

16. Загружаю файлы на Github.

```
eszanulageszanula-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report$ git add .
eszanulageszanula-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report$ git commit -am "feat(main): make course structure"
[main de7e457] feat(main): make course structure
5 files changed, 34 insertions(+), 238 deletions(-)
delete mode 100644 labs/lab02/report/L02_Zamula_Report
delete mode 100644 labs/lab02/report/report.md
delete mode 100644 labs/lab03/report/report.md
create mode 100644 labs/lab04/report/hello.asm
create mode 100644 labs/lab04/report/lab4.asm
eszanulageszanula-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report$ git push
Перечисление объектов: 19, готово.
Подсчет объектов: 100% (19/19), готово.
При сжатии изменений используется до 3 потоков
Сжатие объектов: 100% (11/11), готово.
Запись объектов: 100% (11/11), 1.09 Киб | 223.00 Киб/с, готово.
Всего 11 (изменений 5), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (5/5), completed with 4 local objects.
To github.com:egorzan21/study_2023-2024_arh-pc.git
 f9257e6..de7e457 master -> master
eszanulageszanula-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report$
```

Рис. 16: Загрузка файлов на Гитхаб

17. Проверяю наличие файлов в репозитории

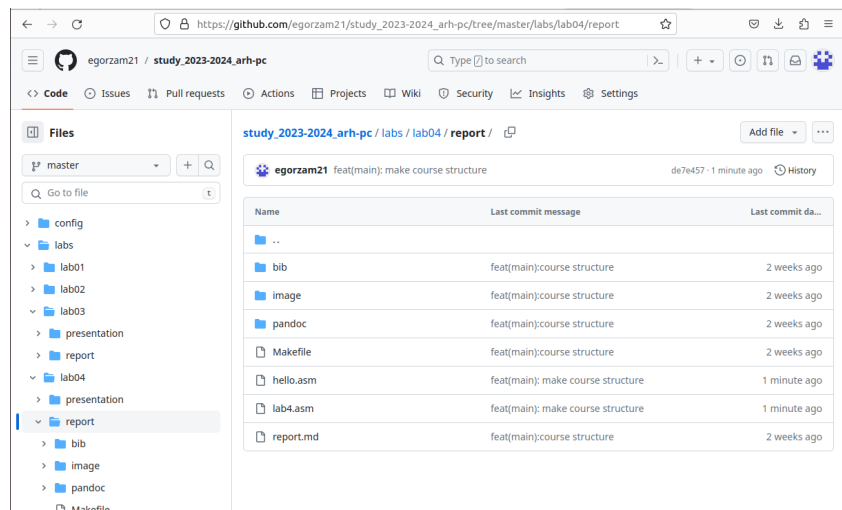


Рис. 17: Проверка на гитхаб

## Выводы

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

## Список литературы