

# **Отчёта по лабораторной работе 5**

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

Цвелев С.А. НПИбд-02-22

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	12
6	Вопросы для самопроверки	13
	Список литературы	15

## Список иллюстраций

4.1	Файл hello.asm . . . . .	9
4.2	Работа программы hello . . . . .	10
4.3	Файл lab05.asm . . . . .	10
4.4	Работа программы lab05 . . . . .	11

## **Список таблиц**

# 1 Цель работы

Целью работы является освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

## 2 Задание

1. Изучите программу HelloWorld и скомпилируйте ее.
2. С помощью любого текстового редактора внесите изменения в текст программы так, чтобы вместо Hello world! на экран выводилась строка с вашими фамилией и именем.
3. Скомпилируйте новую программу и проверьте ее работу.
4. Загрузите файлы на GitHub.

### 3 Теоретическое введение

NASM (англ. Netwide Assembler) – это 80x86 ассемблер, разработанный исходя из принципов переносимости и модульности. Он поддерживает широкий диапазон форматов объектных файлов, включая форматы Linux.out и ELF, NetBSD/FreeBSD, COFF, Microsoft 16-bit OBJ и Win32. Он способен также создавать простые бинарные файлы. Синтакс NASM максимально упрощен для понимания и похож на синтакс Intel, но слегка посложнее. Он поддерживает инструкции Pentium, P6 и MMX, а также имеет макро-расширения.

NASM был создан Саймоном Тэтхемом совместно с Юлианом Холлом и в настоящее время развивается небольшой командой разработчиков на SourceForge.net. Первоначально он был выпущен согласно его собственной лицензии, но позже эта лицензия была заменена на GNU LGPL после множества проблем, вызванных выбором лицензии. Начиная с версии 2.07 лицензия заменена на «упрощённую BSD» (BSD из 2 пунктов).

NASM может работать на платформах, отличных от x86, таких как SPARC и PowerPC, однако код он генерирует только для x86 и x86-64.

NASM успешно конкурирует со стандартным в Linux- и многих других UNIX-системах ассемблером gas. Считается, что качество документации у NASM выше, чем у gas. Кроме того, ассемблер gas по умолчанию использует AT&T-синтаксис, ориентированный на процессоры не от Intel, в то время как NASM использует вариант традиционного для x86-ассемблеров Intel-синтаксиса; Intel-синтаксис используется всеми ассемблерами для под DOS/Windows, например, MASM, TASM, fasm.

В NASM используется Intel-синтаксис записи инструкций. Предложение языка ассемблера NASM (строка программы) может состоять из следующих элементов: Метка, Инструкция, Операнды, Комментарий.

Операнды отделяются между собой запятой. Перед строкой и после инструкции можно использовать любое количество пробельных символов. Комментарий начинается с точки с запятой, а концом комментария считается конец строки. В качестве инструкции может использоваться команда или псевдокоманда (директива компилятора). Если строка очень длинная, то её можно перенести на следующую, используя обратный слеш ( \ ), подобно тому, как это делается в языке Си.



## 4 Выполнение лабораторной работы

1. Создали каталог lab05 командой `mkdir`, перешел в него с помощью команды `cd`, скачал с ТУИС файл `hello.asm` и положил в папку. (рис. 4.1)
2. Открыли файл и изучили текст программы (рис. 4.1)



```
SECTION .data
hello:      db "Hello, world!",10
helloLen:   equ $ - hello

SECTION .text
global _start

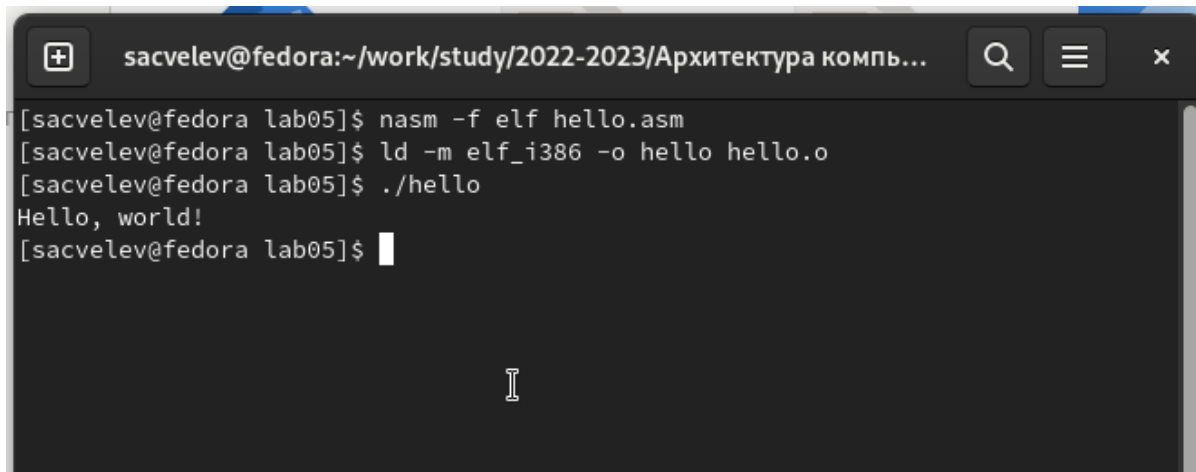
_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, hello
    mov edx, helloLen
    int 0x80

    mov eax, 1
    mov ebx, 0
    int 0x80
```

Рис. 4.1: Файл `hello.asm`

2. Транслировали файл командой `nasm`

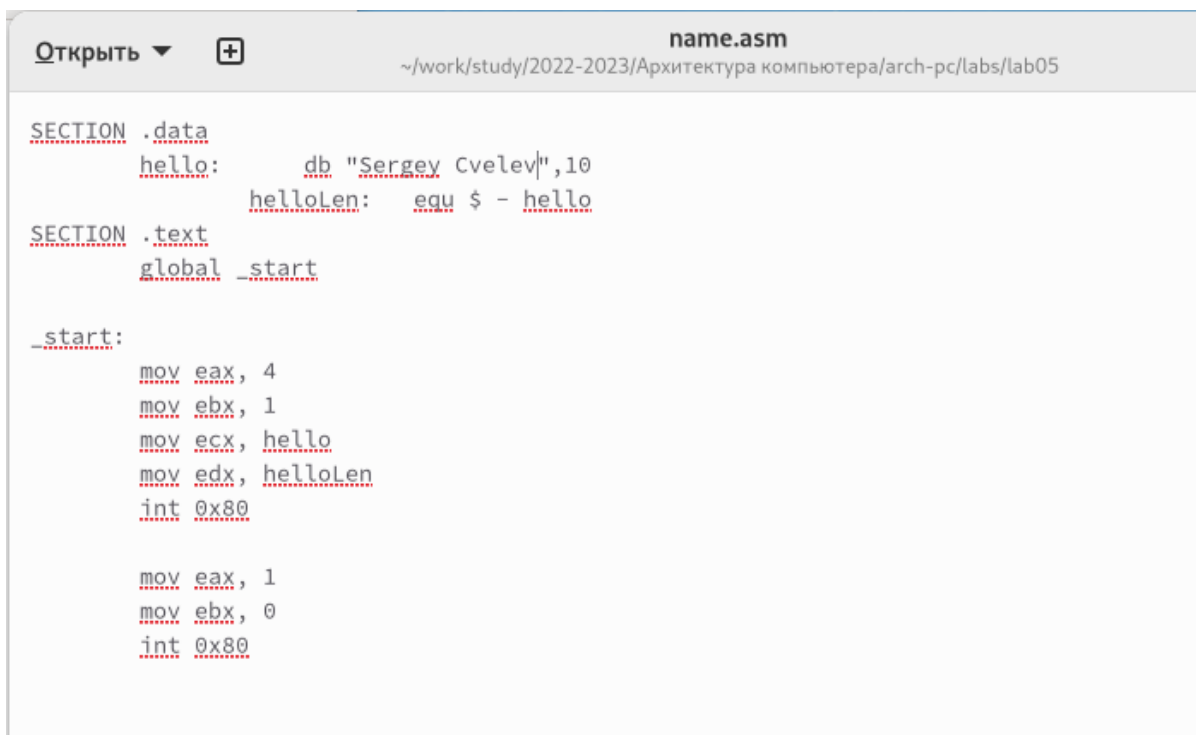
3. Выполнили линковку командой `ld` и получили исполняемый файл и запустили его (рис. 4.2)



```
sacvelev@fedora:~/work/study/2022-2023/Архитектура компь...
[sacvelev@fedora lab05]$ nasm -f elf hello.asm
[sacvelev@fedora lab05]$ ld -m elf_i386 -o hello hello.o
[sacvelev@fedora lab05]$ ./hello
Hello, world!
[sacvelev@fedora lab05]$
```

Рис. 4.2: Работа программы hello

4. Изменили сообщение Hello world на свое имя и запустили файл еще раз (рис. 4.3, 4.4)



```
name.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab05

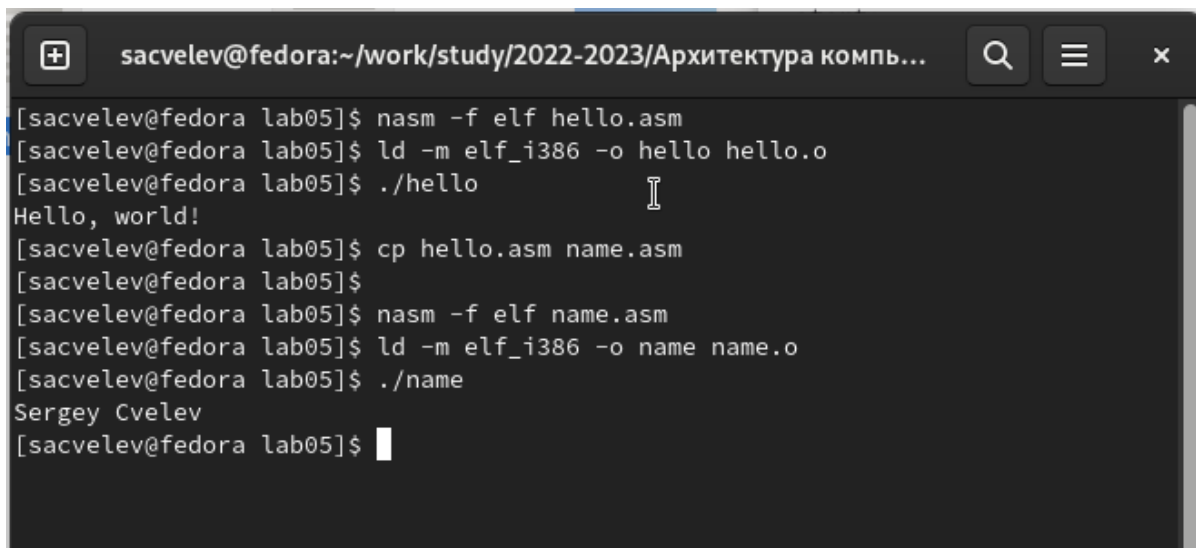
SECTION .data
hello:      db "Sergey Cvelev",10
            helloLen: equ $ - hello

SECTION .text
global _start

_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, hello
    mov edx, helloLen
    int 0x80

    mov eax, 1
    mov ebx, 0
    int 0x80
```

Рис. 4.3: Файл lab05.asm

A terminal window with a dark background and light text. The window title is "sacvelev@fedora:~/work/study/2022-2023/Архитектура компь...". The terminal shows a series of commands and their outputs. The commands are: "nasm -f elf hello.asm", "ld -m elf\_i386 -o hello hello.o", "./hello", "cp hello.asm name.asm", "nasm -f elf name.asm", "ld -m elf\_i386 -o name name.o", and "./name". The outputs are "Hello, world!" and "Sergey Cvelev".

```
sacvelev@fedora lab05]$ nasm -f elf hello.asm
[sacvelev@fedora lab05]$ ld -m elf_i386 -o hello hello.o
[sacvelev@fedora lab05]$ ./hello
Hello, world!
[sacvelev@fedora lab05]$ cp hello.asm name.asm
[sacvelev@fedora lab05]$
[sacvelev@fedora lab05]$ nasm -f elf name.asm
[sacvelev@fedora lab05]$ ld -m elf_i386 -o name name.o
[sacvelev@fedora lab05]$ ./name
Sergey Cvelev
[sacvelev@fedora lab05]$
```

Рис. 4.4: Работа программы lab05

## 5 Выводы

Освоили процесс компиляции и сборки программ, написанных на ассемблере `nasm`.

## 6 Вопросы для самопроверки

1. Какие основные отличия ассемблерных программ от программ на языках высокого уровня? - Ассемблер позволяет работать с ресурсами компьютера на уровне ядра ОС. Это язык низкого уровня, в котором с помощью кодовых инструкций пишутся команды прямо для процессора и регистров.
2. В чём состоит отличие инструкции от директивы на языке ассемблера? - Инструкции выполняются прямо процессором как машинные команды. Директивы не выполняются как команды, а обрабатываются транслятором в инструкции
3. Перечислите основные правила оформления программ на языке ассемблера.  
- Типичный формат записи команд NASM имеет вид: [метка:] мнемокод [операнд {, операнд}] [; комментарий]
4. Каковы этапы получения исполняемого файла? - Написание кода программы, трансляция кода в объектный файл, линковка объектного файла в исполняемый.
5. Каково назначение этапа трансляции? - — преобразование с помощью транслятора, например `nasm`, текста программы в машинный код, называемый объектным
6. Каково назначение этапа компоновки? - этап обработки объектного кода компоновщиком (`ld`), который принимает на вход объектные файлы и собирает по ним исполняемый файл.

7. Какие файлы могут создаваться при трансляции программы, какие из них создаются по умолчанию? - Создается объектный файл .o и можно получить файл листинга .lst.
8. Каковы форматы файлов для nasm и ld? - для nasm на вход подается текст программы в формате .asm. для ld подается объектный файл, полученный от nasm, в формате .o

# Список литературы

1. Расширенный ассемблер: NASM
2. MASM, TASM, FASM, NASM под Windows и Linux