

Отчёта по лабораторной работе 4

Создание и процесс обработки программ на языке ассемблера NASM

Зырянов Артём Алексеевич НБИбд-01-22

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	12
	Список литературы	13

Список иллюстраций

4.1	Файл hello.asm	8
4.2	Работа программы hello	9
4.3	Файл lab04.asm	10
4.4	Работа программы lab04	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

Язык ассемблера (assembly language, сокращённо asm) — машинноориентированный 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

4 Выполнение лабораторной работы

1. Создали каталог lab04 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!",0xa
helloLen:   equ $ - hello

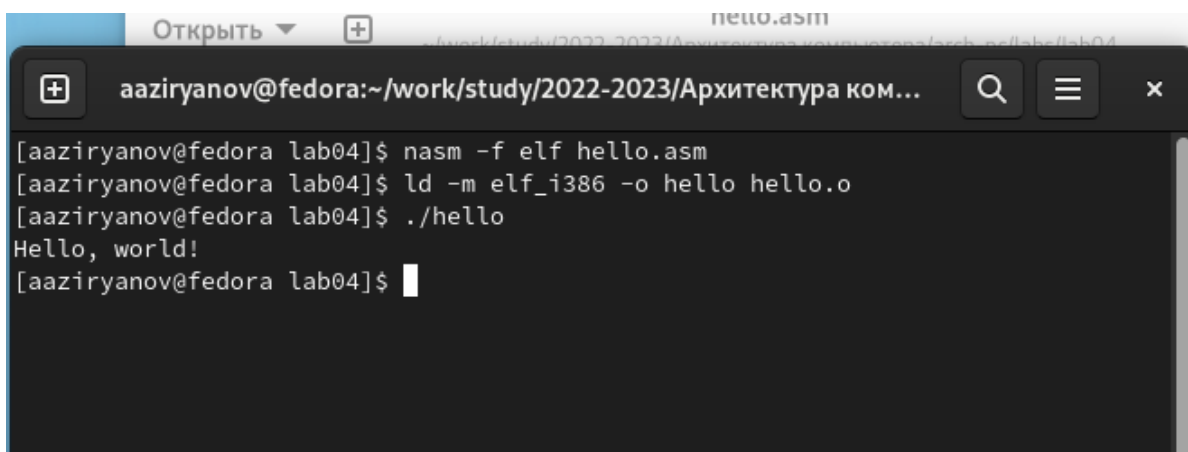
SECTION .text
global _start

_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)



The image shows a terminal window with a dark background. The window title bar includes a search icon, a menu icon, and a close icon. The terminal text is as follows:

```
hello.asm
[aaiziryanov@fedora lab04]$ nasm -f elf hello.asm
[aaiziryanov@fedora lab04]$ ld -m elf_i386 -o hello hello.o
[aaiziryanov@fedora lab04]$ ./hello
Hello, world!
[aaiziryanov@fedora lab04]$
```

Рис. 4.2: Работа программы hello

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

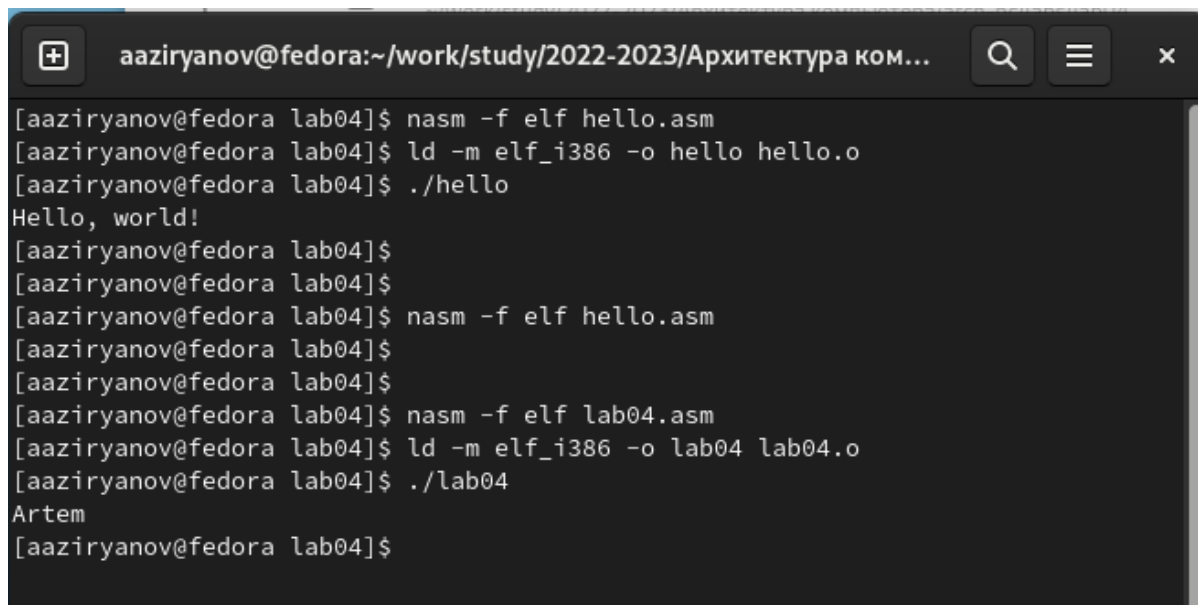
```
SECTION .data
hello:      db "Artem",0xa
helloLen:   equ $ - hello

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 lab04.asm

A terminal window with a dark background and light text. The window title bar shows the user 'aaziryanov@fedora' and the current directory '~/work/study/2022-2023/Архитектура ком...'. The terminal contains the following commands and output:

```
[aaziryanov@fedora lab04]$ nasm -f elf hello.asm
[aaziryanov@fedora lab04]$ ld -m elf_i386 -o hello hello.o
[aaziryanov@fedora lab04]$ ./hello
Hello, world!
[aaziryanov@fedora lab04]$
[aaziryanov@fedora lab04]$
[aaziryanov@fedora lab04]$ nasm -f elf hello.asm
[aaziryanov@fedora lab04]$
[aaziryanov@fedora lab04]$
[aaziryanov@fedora lab04]$ nasm -f elf lab04.asm
[aaziryanov@fedora lab04]$ ld -m elf_i386 -o lab04 lab04.o
[aaziryanov@fedora lab04]$ ./lab04
Artem
[aaziryanov@fedora lab04]$
```

Рис. 4.4: Работа программы lab04

5 Выводы

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

Список литературы

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