

Отчёта по лабораторной работе 5

Создание и процесс обработки программ на языке ассемблера NASM

Понамарев Алексей Михайлович НПИбд-02-22

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	11
6	Вопросы для самопроверки	12
	Список литературы	14

Список иллюстраций

4.1	Файл hello.asm	8
4.2	Работа программы hello	9
4.3	Файл lab05.asm	10
4.4	Работа программы lab05	10

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

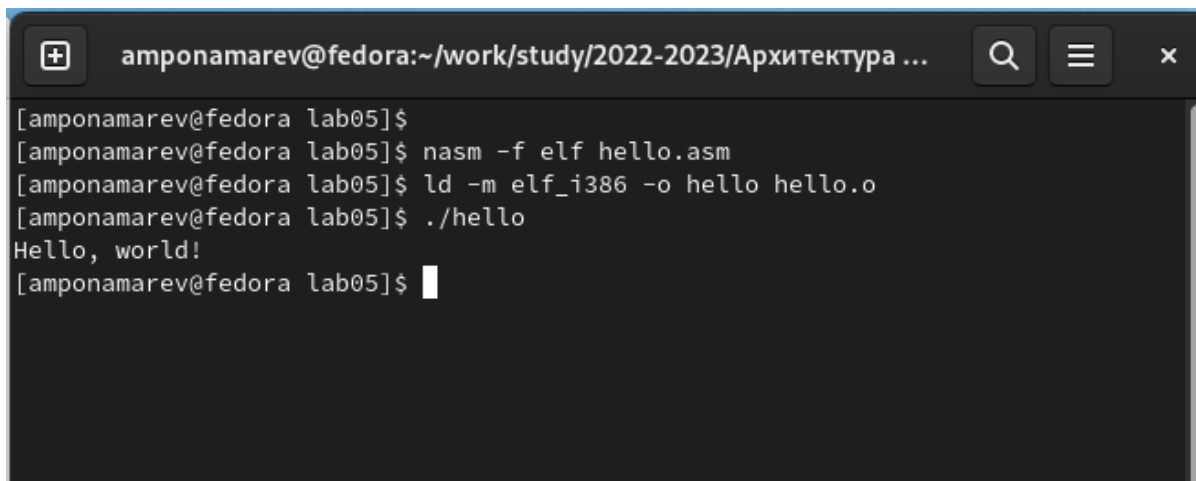
SECTION .text
global _start

_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`

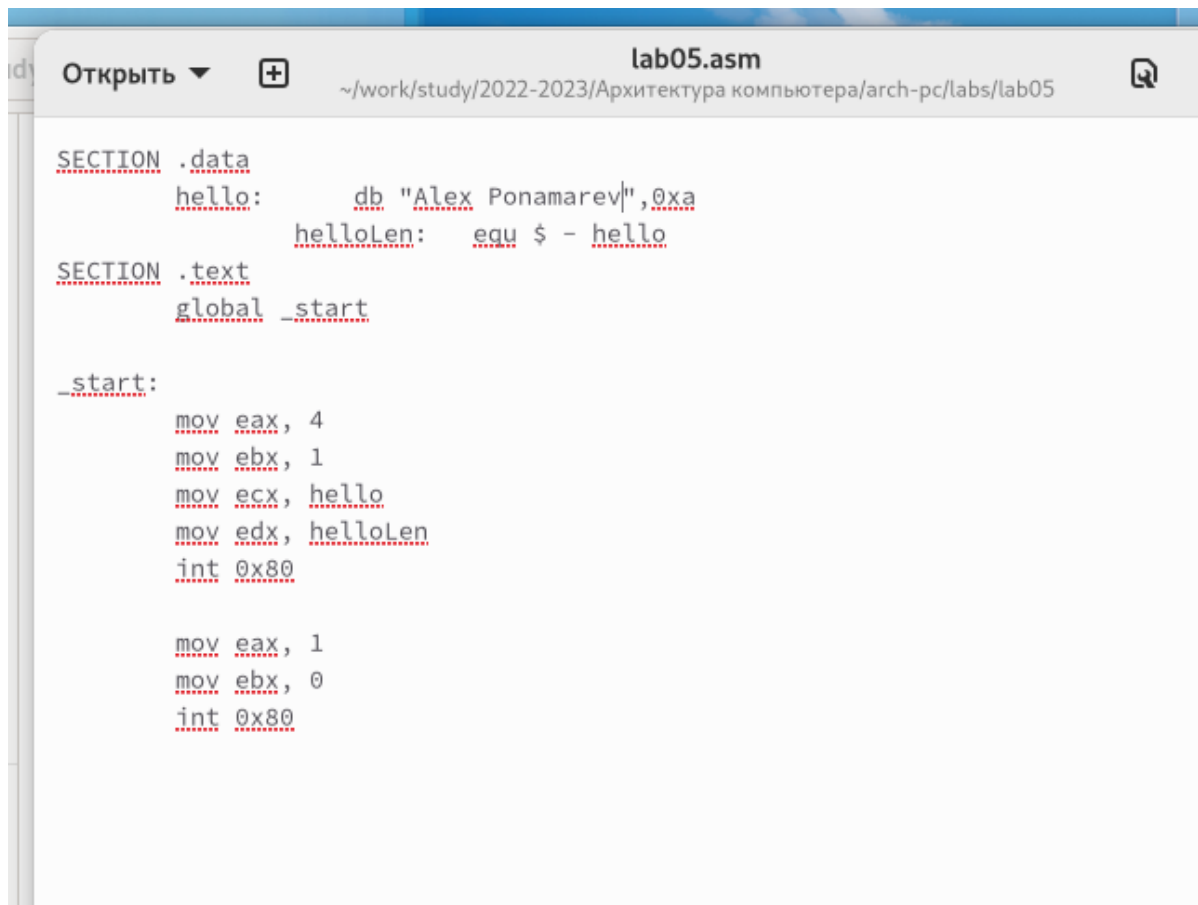
2. Транслировали файл командой `nasm`
3. Выполнили линковку командой `ld` и получили исполняемый файл и запустили его (рис. 4.2)

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

```
[amponamarev@fedora lab05]$  
[amponamarev@fedora lab05]$ nasm -f elf hello.asm  
[amponamarev@fedora lab05]$ ld -m elf_i386 -o hello hello.o  
[amponamarev@fedora lab05]$ ./hello  
Hello, world!  
[amponamarev@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)



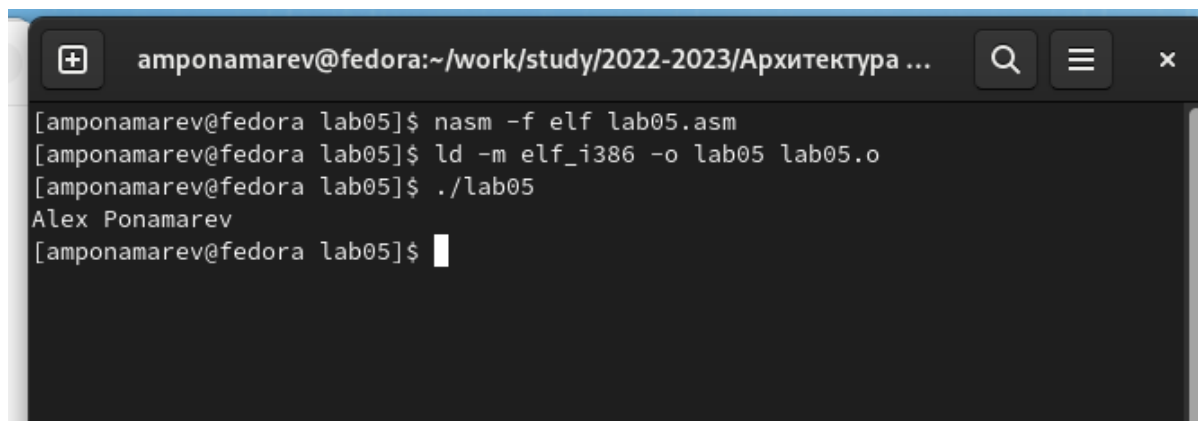
```
SECTION .data
hello:      db "Alex Ponamarev",0xa
helloLen:   equ $ - hello

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 lab05.asm



```
amponamarev@fedora:~/work/study/2022-2023/Архитектура ...
[amponamarev@fedora lab05]$ nasm -f elf lab05.asm
[amponamarev@fedora lab05]$ ld -m elf_i386 -o lab05 lab05.o
[amponamarev@fedora lab05]$ ./lab05
Alex Ponamarev
[amponamarev@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