

# **Лабораторная работа №5**

**Дисциплина: Архитектура компьютера**

Апареев Дмитрий Андреевич

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Задания для самостоятельной работы	11
5	Вывод	12

## Список иллюстраций

3.1	перемещение в каталог . . . . .	7
3.2	создание файла . . . . .	7
3.3	редактирование файла . . . . .	8
3.4	скачивание NASM . . . . .	8
3.5	превращение текст в объективный код . . . . .	9
3.6	компиляция в obj.o . . . . .	9
3.7	передача фалов на обработку . . . . .	9
3.8	передача файлов на обработку . . . . .	9
3.9	компиляция . . . . .	9

## **Список таблиц**

# 1 Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

## 2 Задание

1. Создание программы Hello world!
2. Работа с транслятором NASM
3. Работа с расширенным синтаксисом командной строки NASM
4. Работа с компоновщиком LD
5. Запуск исполняемого файла
6. Выполнение заданий для самостоятельной работы.

## 3 Выполнение лабораторной работы

С помощью `cd` перемещаюсь в каталог, в котором буду работать(рис.1 3.1)

```
[daApareev@fedora ~]$ cd ~/work/study/2022-2023/"Архитектура компьютеров"/arh-pc/labs/lab05
```

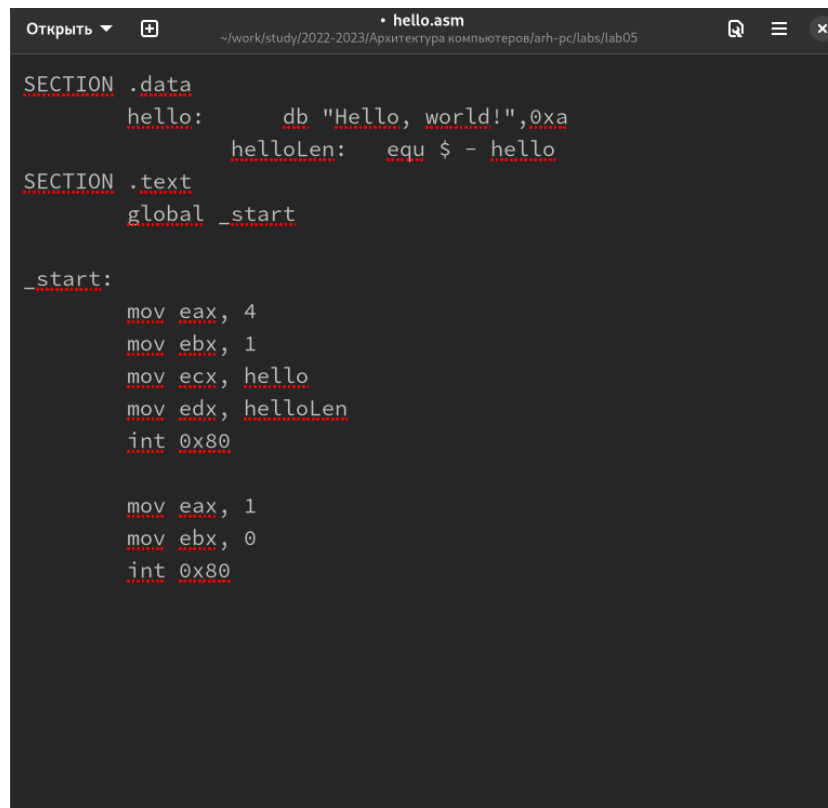
Рис. 3.1: перемещение в каталог

Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью `touch`(рис.2 3.2)

```
[daApareev@fedora lab05]$ touch hello.asm  
[daApareev@fedora lab05]$ ls  
hello.asm  presentation  report
```

Рис. 3.2: создание файла

Открываю созданный файл в текстовом редакторе `gedit` и вставляю в файл программу для вывода “Hello word!”(рис.3 3.3).

A screenshot of a code editor window titled 'hello.asm'. The editor shows assembly code for a program that prints 'Hello, world!'. The code is divided into two sections: .data and .text. The .data section defines a string 'hello' and its length 'helloLen'. The .text section starts at '\_start' and contains two identical blocks of assembly instructions: 'mov eax, 4', 'mov ebx, 1', 'mov ecx, hello', 'mov edx, helloLen', 'int 0x80', followed by 'mov eax, 1', 'mov ebx, 0', and 'int 0x80'.

```
SECTION .data
    hello:      db "Hello, world!",0xa
                helloLen: equ $ - hello

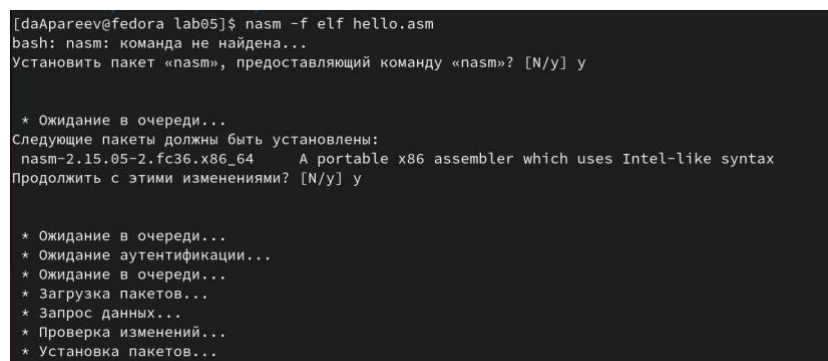
SECTION .text
    global _start

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

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

Рис. 3.3: редактирование файла

Скачиваю необходимые файлы (рис.4 3.4).

A screenshot of a terminal window showing the installation of NASM. The user runs 'nasm -f elf hello.asm', which results in an error 'bash: nasm: команда не найдена...'. The user then runs 'yum install nasm', which prompts for confirmation to install the package. The user responds 'y'. The terminal shows the progress of the installation, including downloading and installing the package.

```
[daApareev@fedora lab05]$ nasm -f elf hello.asm
bash: nasm: команда не найдена...
Установить пакет «nasm», предоставляющий команду «nasm»? [N/y] y

* Ожидание в очереди...
Следующие пакеты должны быть установлены:
nasm-2.15.05-2.fc36.x86_64    A portable x86 assembler which uses Intel-like syntax
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...
```

Рис. 3.4: скачивание NASM

Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду `nasm -f elf hello.asm`(рис.5 3.5).



```
[daApareev@fedora lab05]$ nasm -f elf hello.asm
[daApareev@fedora lab05]$ ls
hello.asm hello.o presentation report
[daApareev@fedora lab05]$
```

Рис. 3.5: превращение текст в объективный код

Ввожу команду, которая скомпилирует файл hello.asm в файл obj.o, при этом в файл будут включены символы для отладки, также с помощью ключа -l будет создан файл list.lst (рис.6 3.6).

```
[daApareev@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[daApareev@fedora lab05]$ ls
hello.asm list.lst presentation
hello.o obj.o report
```

Рис. 3.6: компиляция в obj.o

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello

```
[daApareev@fedora lab05]$ ld -m elf_i386 hello.o -o hello
[daApareev@fedora lab05]$ ls
hello hello.o obj.o report
hello.asm list.lst presentation
```

Рис. 3.7: передача фалов на обработку

Ввожу следующую команду:(рис.8 3.8). Исполняемый файл будет иметь имя main, т.к. после ключа -o было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o

```
[daApareev@fedora lab05]$ ld -m elf_i386 obj.o -o main
[daApareev@fedora lab05]$ ls
hello hello.asm hello.o list.lst main obj.o presentation report
```

Рис. 3.8: передача файлов на обработку

Запускаю на выполнение созданный исполняемый файл, находящийся в текущем каталоге (рис.9 3.9).

```
[daApareev@fedora lab05]$ ./hello
Hello, world!
```

Рис. 3.9: компиляция

Открываю файл report.md с помощью любого текстового редактора gedit. Компилирую файл с отчетом. Загружаю отчет на GitHub.

## 4 Задания для самостоятельной работы

С помощью утилиты `cp` создаю в текущем каталоге копию файла `hello.asm` с именем `lab5.asm` (рис.10 ??).

```
[daApareev@fedora lab05]$ cp hello.asm lab5.asm
```

Редактирую текст файла(рис.12 ??).

```
lab5Len: equ $ - lab5

SECTION .text
global _start

_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, lab5
    mov edx, [lab5Len]
    int 0x80

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

Компилирую текст программы в объектный файл (рис.12 ??).

```
[daApareev@fedora lab05]$ nasm -f elf lab5.asm
[daApareev@fedora lab05]$ ls
hello.o  lab5.o  main  presenta
hello.asm lab5.asm list.lst obj.o  report
```

Передаю объектный файл `lab5.o` на обработку компоновщику LD (рис.13 ??).

```
[daApareev@fedora lab05]$ ./lab5
Dmitry Apareev
```

Запускаю исполняемый файл `lab5` (рис.14 ??).

```
[daApareev@fedora lab05]$ ./lab5
Dmitry Apareev
```

Добавляю файлы в `git` , Сохраняю файлы в `git` , Отправляю файлы на сервер

## **5 Вывод**

При выполнении данной лабораторной работы я освоил процедуры компиляции и сборки программ, написанных на ассемблере NASM.