

# **Отчёт по лабораторной работе №4**

**Дисциплина: Архитектура компьютера**

Будник Александра Олеговна

# Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выполнение заданий для самостоятельной работы	10
5	Выводы	12

# Список иллюстраций

3.1	Создание каталога . . . . .	7
3.2	Создание файла . . . . .	7
3.3	Код программы . . . . .	8
3.4	Трансляция . . . . .	8
3.5	Проверка . . . . .	8
3.6	Расширенный синтаксис . . . . .	9
3.7	Линковка . . . . .	9
3.8	Линковка другого файла . . . . .	9
3.9	Запуск программы . . . . .	9
4.1	Копирование . . . . .	10
4.2	Изменение кода . . . . .	10
4.3	Трансляция, линковка, запуск . . . . .	11

## Список таблиц

# 1 Цель работы

Освоить процедуры компиляции и сборки программ, которые написаны на ассемблере NASM.

## 2 Теоретическое введение

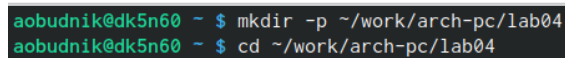
Основные функциональные элементы любой ЭВМ - центральный процессор (ЦП), память, периферийные устройства. В состав ЦП входят арифметико-логическое устройство (АЛУ), устройство управления (УУ) и регистры. Периферийные устройства можно разделить на устройства внешней памяти и устройства ввода-вывода.

Язык ассемблера - это машинно-ориентированный язык низкого уровня. Считается, что он наиболее приближен к архитектуре ЭВМ. Ассемблер - это специальная программа транслятор.

Процесс создания и обработки программы на языке ассемблера: 1. Текст программы набирается в текстовом редакторе (файл с расширением .asm). 2. При помощи NASM (транслятор) текст превращается в объектный код. 3. При помощи компоновщика ld код обрабатывается, создается исполняемый файл. 4. Производится запуск программы

### 3 Выполнение лабораторной работы

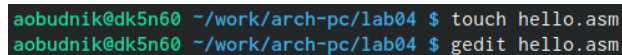
При помощи команды `mkdir` создаю каталог для работы с программами на языке ассемблера NASM. Перехожу в него через команду `cd` (рис. 3.1).



```
aobudnik@dk5n60 ~ $ mkdir -p ~/work/arch-pc/lab04
aobudnik@dk5n60 ~ $ cd ~/work/arch-pc/lab04
```

Рис. 3.1: Создание каталога

Создаю файл `hello.asm` командой `touch` и открываю его в текстовом редакторе (рис. 3.2).



```
aobudnik@dk5n60 ~/work/arch-pc/lab04 $ touch hello.asm
aobudnik@dk5n60 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 3.2: Создание файла

В текстовом редакторе ввожу будущий код программы, соблюдая синтаксис языка (рис. 3.3).

```

1 ; hello.asm
2 SECTION .data
3     hello:          DB 'Hello world!',10
4
5     helloLen:       EQU $-hello
6
7 SECTION .text
8     GLOBAL _start
9
10 _start:
11     mov eax,4
12     mov ebx,1
13     mov ecx,hello
14     mov edx,helloLen
15     int 80h
16
17     mov eax,1
18     mov ebx,0
19     int 80h

```

Рис. 3.3: Код программы

При помощи транслятора NASM превращаю текст в объектный код, вводя следующую команду (рис. 3.4).

```

aobudnik@dk5n60 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm

```

Рис. 3.4: Трансляция

Проверяю, что файл расширения .o с кодом создан (рис. 3.5).

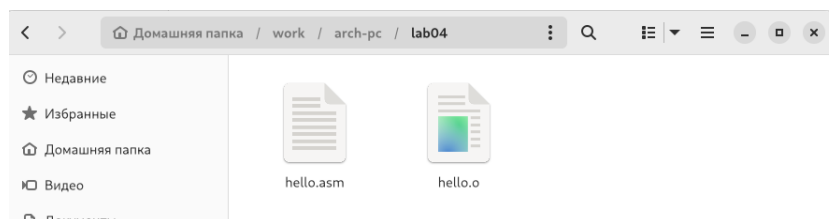


Рис. 3.5: Проверка

Выполняю следующую последовательность команд, чтобы узнать расширенный синтаксис командной строки. Она компилирует файл hell.asm в obj.o, так чтобы формат выходного файла будет elf, в него будут включены символы для отладки и создан файл листинга. При помощи команды ls проверяю, все ли файлы были созданы (рис. 3.6).



```
aobudnik@dk5n60 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
aobudnik@dk5n60 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
```

Рис. 3.6: Расширенный синтаксис

С помощью компоновщика LD обрабатываю объектный файл и получаю исполняемый файл (рис. 3.7).

```
aobudnik@dk5n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
```

Рис. 3.7: Линковка

Ввожу аналогичную последовательность команд с другими исходными данными (рис. 3.8). При этом исполняемый файл будет называться main. Объектный файл, из которого он собран, называется obj.o.

```
aobudnik@dk5n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
```

Рис. 3.8: Линковка другого файла

Запускаю получившуюся программу. Она выводит текст “Hello World!” (рис. 3.9).

```
aobudnik@dk5n60 ~/work/arch-pc/lab04 $ ./hello
Hello world!
```

Рис. 3.9: Запуск программы

## 4 Выполнение заданий для самостоятельной работы

В каталоге создаю копию файла `hello.asm` с названием `lab4.asm` при помощи команды `cp` (рис. 4.1).

```
aobudnik@dk5n60 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
aobudnik@dk5n60 ~/work/arch-pc/lab04 $ gedit lab4.asm
```

Рис. 4.1: Копирование

Открываю получившийся файл в текстовом редакторе и изменяю код так, чтобы при запуске программа выводила моё имя и фамилию (рис. 4.2).

```
1; hello.asm
2 SECTION .data
3     hello:      DB 'Budnik Alexandra',10
4
5     helloLen:   EQU $-hello
6
7 SECTION .text
8     GLOBAL _start
9
10 _start:
11     mov eax,4
12     mov ebx,1
13     mov ecx,hello
14     mov edx,helloLen
15     int 80h
16
17     mov eax,1
18     mov ebx,0
19     int 80h
```

Рис. 4.2: Изменение кода

Произвожу трансляцию текста в объектный файл, затем его линковку (компоновку). Получаю исполняемый файл. Запускаю программу (рис. 4.3).

```
aobudnik@dk5n60 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
aobudnik@dk5n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
aobudnik@dk5n60 ~/work/arch-pc/lab04 $ ./lab4
Budnik Alexandra
```

Рис. 4.3: Трансляция, линковка, запуск

Все получившиеся файлы копирую в локальный репозиторий и загружаю на GitHub.

## **5 Выводы**

По итогам выполнения лабораторной работы №4 я научилась работать с машинно-ориентированным языком низкого уровня ассемблера NASM и создала первые две программы.