

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4**

дисциплина: Архитектура компьютера

Студент: Бондаренко София

Группа: НБИбд-01-25

**МОСКВА**

2025 г.

# **Оглавление**

<b>Оглавление .....</b>	<b>Ошибка! Закладка не определена.</b>
<b>1. Цель работы.....</b>	<b>3</b>
<b>2. Задание.....</b>	<b>4</b>
<b>3. Теоретическое введение .....</b>	<b>5</b>
<b>4. Выполнение задания .....</b>	<b>7</b>
1. Программа Hello world! .....	7
2. Транслятор NASM.....	8
3. Расширенный синтаксист командной строки NASM .....	8
4. Компоновщик LD .....	8
5. Запуск исполняемого файла .....	9
6. Задание для самостоятельной работы.....	9
<b>5. Вывод .....</b>	<b>11</b>
<b>6. Список литературы.....</b>	<b>12</b>

## **1. Цель работы**

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

## **2. Задание**

1. Программа Hello world!
2. Транслятор NASM
3. Расширенный синтаксис командной строки NASM
4. Компоновщик LD
5. Запуск исполняемого файла
6. Задание для самостоятельной работы

### **3. Теоретическое введение**

Любая электронно-вычислительная машина состоит из трех ключевых элементов, которые взаимодействуют через общую шину. Шина — это набор проводников (в современных ПК это дорожки на материнской плате), служащий магистралью для обмена данными между устройствами.

1. Центральный процессор (ЦП). Его главные задачи — обработка информации и координация работы всех узлов компьютера. В состав процессора входят:

- Арифметико-логическое устройство (АЛУ): выполняет математические и логические операции над данными.
- Устройство управления (УУ): управляет работой всех устройств компьютера.
- Регистры: это сверхбыстрая память внутри процессора, используемая для временного хранения промежуточных результатов. Они делятся на регистры общего назначения и специальные.

2. Оперативное запоминающее устройство (ОЗУ). Это быстрая, но энергозависимая память, предназначенная для хранения программ и данных, с которыми процессор работает в текущий момент. Данные в ОЗУ хранятся в пронумерованных ячейках, где номер ячейки является ее адресом.

3. Периферийные устройства. К ним относятся:

- Устройства внешней памяти (для долговременного хранения данных).
- Устройства ввода-вывода (для взаимодействия с внешним миром).

Компьютер работает по принципу программного управления, то есть выполняет задачу как последовательность команд (программу). Каждая машинная команда — это двоичный код, состоящий из двух частей:

- Операционная часть: указывает, какую операцию выполнить (например, сложить, переместить).
- Адресная часть: содержит данные или адреса данных, над которыми выполняется операция.

Для выполнения команды процессор следует командному циклу:

1. Формирование адреса следующей команды.
2. Считывание и дешифрация команды из памяти.
3. Выполнение команды.
4. Переход к следующей команде.

Язык Ассемблера — это низкоуровневый язык, тесно связанный с архитектурой процессора. Большинство команд в программах на Ассемблере используют регистры в качестве операндов (например, пересылка данных между регистрами или арифметические операции над ними). В отличие от ячеек оперативной памяти, доступ к регистрам осуществляется по именам, а не по адресам. В архитектуре x86 регистры общего назначения имеют иерархическую структуру и разные разрядности:

- 64-битные: RAX, RCX, RDX, RBX, RSI, RDI
- 32-битные: EAX, ECX, EDX, EBX, ESI, EDI
- 16-битные: AX, CX, DX, BX, SI, DI
- 8-битные: AH, AL, CH, CL, DH, DL, BH, BL

NASM — это популярный кроссплатформенный ассемблер с открытым исходным кодом. Он использует интеловский синтаксис и поддерживает современные 64-битные инструкции (x86-64), что делает его мощным инструментом для низкоуровневого программирования.

## 4. Выполнение задания

### 1. Программа *Hello world!*

Создаю каталог, в котором буду работать с программами. (рис.1)

```
snbondarenko@virtualbox:~$ mkdir -p ~/work/arch-pc/lab04
```

рис.1 Создание каталога

После создания перехожу в него с помощью утилиты cd. (рис.2)

```
snbondarenko@virtualbox:~$ cd ~/work/arch-pc/lab04
```

рис.2 Переход в директорию

Создаю файл hello.asm. (рис.3)

```
snbondarenko@virtualbox:~/work/arch-pc/lab04$ touch hello.asm
```

рис.3 Создание файла

Открываю созданный ранее файл с помощью текстового редактора gedit. (рис.4)

```
snbondarenko@virtualbox:~/work/arch-pc/lab04$ gedit hello.asm
```

рис.4 Открытие файла в текстовом редакторе

Ввожу в него предложенную программу из теоретического файла к лабораторной работе. (рис.5)

```
1 ; hello.asm
2 section .data
3 ; Начало секции данных
4 hello:DB 'Hello world!',10 ; 'Hello world!' плюс
5 helloLen:EQU $-hello
6 ; символ строки перевода
7 SECTION .text
8 ; Длина строки hello
9 ; Начало секции кода
10 GLOBAL _start
11 _start:
12 ; Точка входа в программу
13 mov eax,4; Системный вызов для записи (sys_write)
14 mov ebx,1; Описатель файла '1' - стандартный вывод
15 mov ecx,hello; Адрес строки hello в ecx
16 mov edx,helloLen; Размер строки hello
17 int 80h; Вызов ядра
18 mov eax,1; Системный вызов для выхода (sys_exit)
19 mov ebx,0; Выход с кодом возврата '0' (без ошибок)
20 int 80h; Вызов ядра
```

рис.5 Ввод программы в текстовом редакторе

## 2. Транслятор NASM

Превращаю текст программы в код с помощью транслятора NASM, использую команду nasm -f elf hello.asm , ключ –f указывает на то, что требуется создать бинарные файлы формата ELF. (рис.6)

```
snbondarenko@virtualbox:~/work/arch-pc/lab04$ nasm -f elf hello.asm
```

рис.6 Компиляция

## 3. Расширенный синтаксист командной строки NASM

Ввожу команду, которая скомпилирует исходный файл hello.asm в obj.o, при этом формат выходного файла будет elf, в него будут включены символы для откладки (опция –g), также будет создан файл листинга list.lst (опция -l). И проверяю правильность выполнения программы с помощью утилиты ls. (рис.7)

```
snbondarenko@virtualbox:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
snbondarenko@virtualbox:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst obj.o
```

рис.7 Компиляция текста программы и проверка

## 4. Компоновщик LD

Передаю объектный файл hello.o компоновщику LD, чтобы получить исполняемую программу. Ключ –o со следующими значениями задает имя создаваемого исполняемого файла. С помощью команды ls проверяю корректность выполнения команды. (рис.8)

```
snbondarenko@virtualbox:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
snbondarenko@virtualbox:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst main obj.o
```

рис.8 Передача объектного файла на обработку компоновщику и проверка

Выполняю следующую команду, исполняемый файл будет иметь имя main, потому что после ключа –o указано значение main. Объектный файл, из которого был собран исполняемый файл, имеет название obj.o. (рис.9)

```
snbondarenko@virtualbox:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
snbondarenko@virtualbox:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst main obj.o
```

рис.9 . Передача объектного файла на обработку компоновщику и проверка

## 5. Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello.o. (рис.10)

```
snbondarenko@virtualbox:~/work/arch-pc/lab04$ ./hello
Hello world!
```

рис.10 Запуск исполняемого файла

## 6. Задание для самостоятельной работы

1. С помощью утилиты cp в текущем каталоге создаю копию файла hello.asm с названием lab4.asm. С помощью ls проверяю выполнение команды. (рис.11)

```
snbondarenko@virtualbox:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
snbondarenko@virtualbox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
```

рис.11 Копирование файла и проверка

2. С помощью текстового редактора gedit ввожу изменения в файл lab4.asm, чтобы программа выводила мои имя и фамилию. (рис.12, рис.13)

```
snbondarenko@virtualbox:~/work/arch-pc/lab04$ gedit lab4.asm
```

рис.12 Открываю текстовый редактор

```
1 ; lab4.asm
2 section .data
3 ; Начало секции данных
4 lab4:DB 'Sofiiia Bondarenko',10 ; 'Sofiiia Bondarenko' плюс
5 lab4Len:EQU $-lab4
6 ; символ строки перевода
7 SECTION .text
8 ; Длина строки lab4
9 ; Начало секции кода
0 GLOBAL _start
1 _start:
2 ; Точка входа в программу
3 mov eax,4; Системный вызов для записи (sys_write)
4 mov ebx,1; Описатель файла '1' - стандартный вывод
5 mov ecx,lab4; Адрес строки lab в ecx
6 mov edx,lab4Len; Размер строки lab4
7 int 80h; Вызов ядра
8 mov eax,1; Системный вызов для выхода (sys_exit)
9 mov ebx,0; Выход с кодом возврата '0' (без ошибок)
0 int 80h; Вызов ядра
```

рис.13 Редактирование текста программы

3. Компилирую текст программы в объектный файл и сразу проверяю правильность выполнения. (рис.14)

```
snbondarenko@virtualbox:~/work/arch-pc/lab04$ nasm -f elf lab4.asm  
snbondarenko@virtualbox:~/work/arch-pc/lab04$ ls  
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o
```

рис.14 Компиляция текста программы

Передаю объектный файл компоновщику LD, чтобы получить исполняемую программу. (рис.15)

```
snbondarenko@virtualbox:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4  
snbondarenko@virtualbox:~/work/arch-pc/lab04$ ls  
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
```

рис.15 Передача объектного файла на обработку компоновщику

Запускаю полученный исполняемый файл. (рис. 16)

```
snbondarenko@virtualbox:~/work/arch-pc/lab04$ ./lab4  
Sofilia Bondarenko
```

рис.16 Запуск исполняемого файла

4. Копирую файлы hello.asm и lab4.asm в личный репозиторий. (рис.17)

```
snbondarenko@virtualbox:~$ cp ~/work/arch-pc/lab04/hello.asm ~/work/study/2025-2026/"Архитектура компьютера"/arch-pc/labs/lab04  
snbondarenko@virtualbox:~$ cp ~/work/arch-pc/lab04/lab4.asm ~/work/study/2025-2026/"Архитектура компьютера"/arch-pc/labs/lab04  
snbondarenko@virtualbox:~$ ls ~/work/study/2025-2026/"Архитектура компьютера"/arch-pc/labs/lab04  
hello.asm lab4.asm presentation report
```

рис.17 Копирование файлов

В каталог ~/work/study/2025-2026/"Архитектура компьютеров"/arch-pc/labs/lab04/report загружаю файл отчета в формате .docx и .pdf. С помощью команды git add . и git commit “...” добавляю файлы на GitHub, комментируя изменения как добавление файлов. Затем отправляю файлы на сервер с помощью команды git push.

## **5. Вывод**

В ходе выполнения работы, я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

## **6. Список литературы**

[https://esystem.rudn.ru/pluginfile.php/2091234/mod\\_resource/content/0/Лабораторна я%20работа%20№4.%20Создание%20и%20процесс%20обработки%20программ %20на%20языке%20ассемблера%20NASM.pdf](https://esystem.rudn.ru/pluginfile.php/2091234/mod_resource/content/0/Лабораторна я%20работа%20№4.%20Создание%20и%20процесс%20обработки%20программ %20на%20языке%20ассемблера%20NASM.pdf)