

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**  
**Факультет физико-математических и естественных наук**  
**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4**

**дисциплина: Архитектура компьютеров и  
операционные системы**

Студент: Райко М. В.

Группа: НБИбд-03–23

МОСКВА

2023 г.

# Содержание

<b>1.Цель работы.....</b>	<b>3</b>
<b>2.Задание.....</b>	<b>4</b>
<b>3.Теоритическое введение.....</b>	<b>5</b>
<b>4.Выполнение лабораторной работы.....</b>	<b>6-13</b>
○ <b>4.1</b> Программа Hello world!.....	<b>6-7</b>
○ <b>4.2</b> Транслятор NASM.....	<b>8</b>
○ <b>4.3</b> Расширенный синтаксис командной строки NASM.....	<b>9</b>
○ <b>4.4</b> Компоновщик LD.....	<b>10</b>
○ <b>4.5</b> Запуск исполняемого файла.....	<b>11</b>
○ <b>4.6</b> Задание для самостоятельной работы.....	<b>12-14</b>
<b>5.Заключение.....</b>	<b>15</b>

# **1.Цель работы:**

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

## **2.Задание:**

- **4.1** Программа Hello world!
- **4.2** Транслятор NASM
- **4.3** Расширенный синтаксис командной строки NASM
- **4.4** Компоновщик LD
- **4.5** Запуск исполняемого файла
- **4.6** Задание для самостоятельной работы

### 3. Теоретическое введение:

ЭВМ состоит из центрального процессора (ЦП), памяти и периферийных устройств, связанных общей шиной. ЦП включает арифметико-логическое устройство (выполняет операции), устройство управления (контролирует устройства) и регистры (хранят данные). В программировании на ассемблере регистры используются для операций с данными, таких как перемещение и арифметика.

ЭВМ включает периферийные устройства для хранения данных (например, жёсткие диски) и устройства ввода-вывода для взаимодействия с внешней средой. Программное управление осуществляется машинными командами, состоящими из операционной и адресной частей. Команды выполняются в командном цикле процессора, включающем формирование адреса, считывание, дешифрацию, выполнение и переход к следующей команде.

**Язык ассемблера (asm):** это машинно-ориентированный язык низкого уровня, близкий к архитектуре ЭВМ. В отличие от языков высокого уровня, ассемблер позволяет более полный контроль над аппаратными ресурсами. Программы на ассемблере состоят из команд, понятных процессору, записанных с использованием мнемоник. Ассемблерные программы транслируются в машинный код транслятором (ассемблером), обеспечивая высокую производительность. На языке ассемблера программисты указывают инструкции, которые прямо соответствуют операциям процессора, сохраняя контроль над ресурсами.

Создание ассемблерной программы:

**Написание и сохранение:** Программист пишет код на ассемблере и сохраняет в файл (.asm).

**Трансляция:** Транслятор (например, nasm) преобразует текст программы в машинный код, создавая объектный файл (.o) и листинг (.lst).

**Компоновка:** Компоновщик (ld) обрабатывает объектные файлы, создавая исполняемый файл без расширения, возможно, с картой загрузки (.tar).

**Запуск программы:** Исполняемый файл запускается. Ошибки требуют отладки с использованием отладчика, после чего цикл повторяется.

## 4.Выполнение лабораторной работы:

### 4.1 Программа Hello world!:

Создайте каталог для работы с программами на языке ассемблера NASM:

```
matvei@MatveiRay:~$ mkdir -p ~/work/arch-pc/lab04  
matvei@MatveiRay:~$
```

(4.1.1 Создание каталога лаб4)

Перейдите в созданный каталог:

```
matvei@MatveiRay:~$ cd ~/work/arch-pc/lab04  
matvei@MatveiRay:~/work/arch-pc/lab04$
```

(4.1.2 Перешел в созданный каталог)

Создайте текстовый файл с именем hello.asm:

```
matvei@MatveiRay:~/work/arch-pc/lab04$ touch hello.asm  
matvei@MatveiRay:~/work/arch-pc/lab04$
```

(4.1.3 Создал текстовый файл)

Откройте этот файл с помощью любого текстового редактора, например, gedit:



(4.1.4 Открыл текстовый файл с помощью команды gedit)

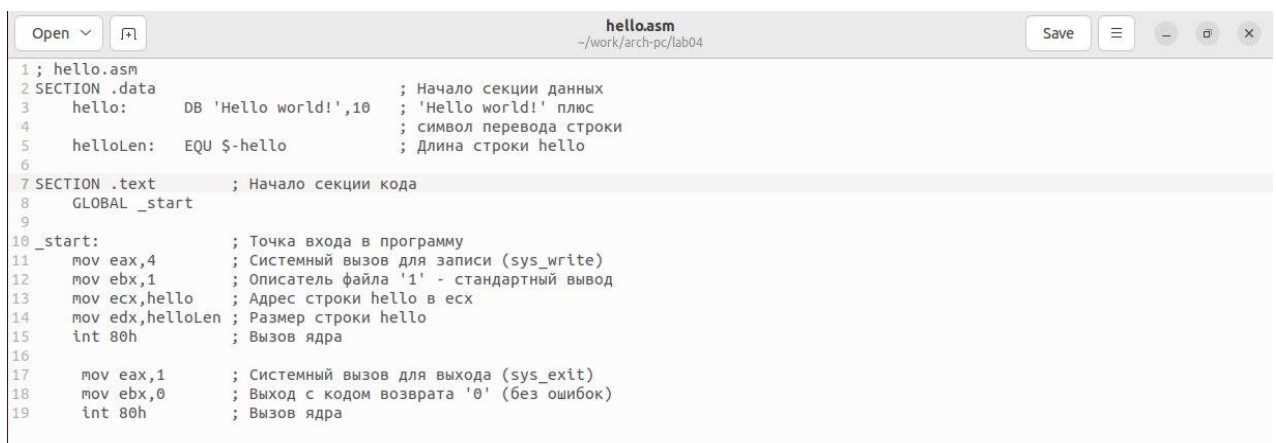
и введите в него следующий текст:

```
; hello.asm
SECTION .data                ; Начало секции данных
    hello:      DB 'Hello world!',10 ; 'Hello world!' плюс
                                ; символ перевода строки
    helloLen:   EQU $-hello      ; Длина строки hello

SECTION .text                ; Начало секции кода
    GLOBAL _start

_start:                      ; Точка входа в программу
    mov eax,4                ; Системный вызов для записи (sys_write)
    mov ebx,1                ; Описатель файла '1' - стандартный вывод
    mov ecx,hello            ; Адрес строки hello в ecx
    mov edx,helloLen         ; Размер строки hello
    int 80h                  ; Вызов ядра

    mov eax,1                ; Системный вызов для выхода (sys_exit)
    mov ebx,0                ; Выход с кодом возврата '0' (без ошибок)
    int 80h                  ; Вызов ядра
```



(4.1.5 Ввел синтаксис, который выводит в ассемблере программу Hello World)

## 4.2 Транслятор NASM:

Для компиляции приведённого выше текста программы «Hello World» необходимо написать:

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

(4.2.1 Преобразование текстового файла в объективный код с помощью транслятора NASM)

С помощью команды `ls` проверьте, что объектный файл был создан:

```
matvei@MatveiRay:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
matvei@MatveiRay:~/work/arch-pc/lab04$
```

(4.2.2 NASM транслировал в объективный код с названием `hello.o`)

## 4.3 Расширенный синтаксис командной строки NASM:

Выполните следующую команду



(данная команда скомпилирует исходный файл `hello.asm` в `obj.o` (опция `-o` позволяет задать имя объектного файла, в данном случае `obj.o`), при этом формат выходного файла будет `elf`, и в него будут включены символы для отладки (опция `-g`), кроме того, будет создан файл листинга `list.lst` (опция `-l`).) :

```
matvei@MatveiRay:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
matvei@MatveiRay:~/work/arch-pc/lab04$
```

(4.3.1 Компиляция файла)

## 4.4 Компоновщик LD:

Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику:

```
matvei@MatveiRay:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
matvei@MatveiRay:~/work/arch-pc/lab04$
```

(4.4.1 Передаем файл компоновщику, чтобы исполняемый файл Hello был создан)

С помощью команды `ls` проверьте, что исполняемый файл `hello` был создан:

```
hello hello.asm hello.o list.lst obj.o
matvei@MatveiRay:~/work/arch-pc/lab04$
```

(4.4.2 Утверждаемся в этом)

Выполните следующую команду:

```
matvei@MatveiRay:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
matvei@MatveiRay:~/work/arch-pc/lab04$
```

(4.4.3 Объединение объектного файла в исполняемый файл ELF под архитектуру x86)

## 4.5 Запуск исполняемого файла:

Запустить на выполнение созданный исполняемый файл, находящийся в текущем каталоге, можно, набрав в командной строке:

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

(4.5.1 Вывод программы Hello world!)

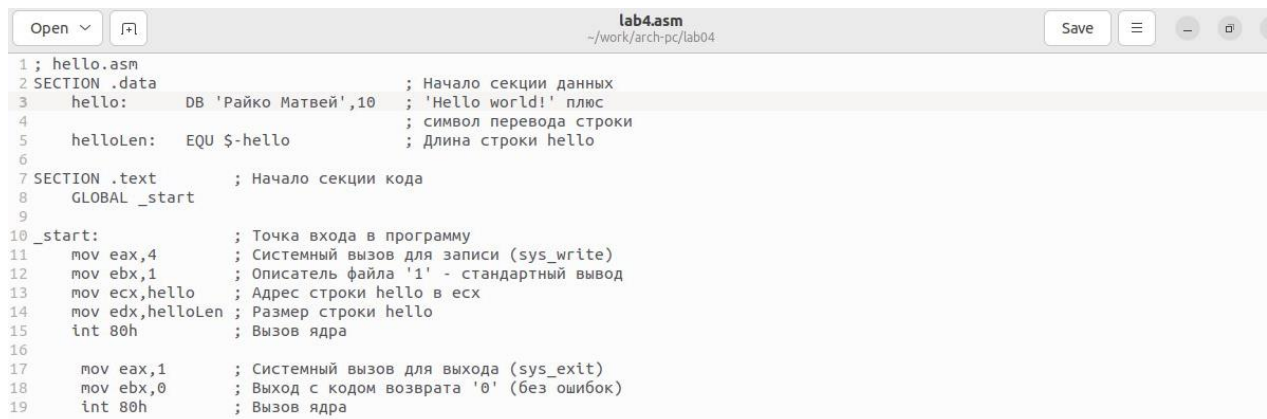
## 4.6 Задание для самостоятельной работы:

1. В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создайте копию файла `hello.asm` с именем `lab4.asm`

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

(4.6.1 Создал копию с помощью команды cp)

2. С помощью любого текстового редактора внесите изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с вашими фамилией и именем.



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

(4.6.2 Ввел свою Фамилию и свое имя в текстовый редактор)

3. Оттранслируйте полученный текст программы lab4.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл.

```

matvei@MatveiRay:~/work/arch-pc/lab04$ nasm -f elf hello.asm
matvei@MatveiRay:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
matvei@MatveiRay:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
matvei@MatveiRay:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o
matvei@MatveiRay:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst lab4.asm
matvei@MatveiRay:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o
matvei@MatveiRay:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
matvei@MatveiRay:~/work/arch-pc/lab04$
matvei@MatveiRay:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o
matvei@MatveiRay:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
matvei@MatveiRay:~/work/arch-pc/lab04$ ./lab4
Райко Матвей
matvei@MatveiRay:~/work/arch-pc/lab04$

```

(4.6.3 Компановка объектного файла lab4 и последующий его запуск)

4. Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/. Загрузите файлы на Github:

```

matvei@MatveiRay:~/work/arch-pc/lab04$ cp hello.asm ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/
matvei@MatveiRay:~/work/arch-pc/lab04$ cp lab4.asm ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/
matvei@MatveiRay:~/work/arch-pc/lab04$ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04
matvei@MatveiRay:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm  lab4.asm  presentation  report
matvei@MatveiRay:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ git add hello.asm lab4.asm
matvei@MatveiRay:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ git commit -m "Добавлены файлы hello.asm и lab4.asm"
[master c1cafef] Добавлены файлы hello.asm и lab4.asm
2 files changed, 38 insertions(+)
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab4.asm
matvei@MatveiRay:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ git push -u origin master
To github.com:MatveiRay/study_2023-2024_arch-pc.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'github.com:MatveiRay/study_2023-2024_arch-pc.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

```

(4.6.4 Скопировал файлы,загрузил в локал каталог,а после загрузил на гитхаб)

MatveiRay Add files via upload 442fbb2 · now History		
Name	Last commit message	Last commit date
..		
presentation	feat(main): make course structure	last week
report	feat(main): make course structure	last week
hello.asm	Add files via upload	now
lab4.asm	Add files via upload	now

(4.6.5 Проверил на гитхабе загруженные файлы)

## **5. Заключение:**

Освоение ассемблера способствует развитию навыков оптимизации кода, что приводит к повышению производительности программ. Кроме того, данное изучение открывает путь к освоению более высокоуровневых языков программирования, предоставляя более высокий уровень абстракции и удобства при написании кода. Приобретенные знания по языку ассемблера также облегчают изучение работы компилятора и понимание того, какие инструкции выполняются на более низком уровне.