

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

Студент: Барханоева Раина Магометовна

Ст.билет: 1032252468

Группа: НКАбд-01-25

МОСКВА

2025 г

Содержание

1. Цель работы.....	3
2. Выполнение работы.....	4
2.1. Программа Hello world!.....	4
2.2. Транслятор NASM.....	5
2.3. Расширенный синтаксис командной строки NASM.....	6
2.4. Компоновщик LD.....	7
3. Самостоятельная работа.....	8
4. Вывод.....	10

1. Цель работы

Изучить процесс создания, трансляции и компоновки программ на языке ассемблера NASM, а также освоить практические навыки работы с системными вызовами и регистрами процессора при получении исполняемого файла в среде Linux.

2. Выполнение работы

2.1. Программа Hello world!

С помощью команды mkdir мы создали папку lab4 (рисунок 1).

```
rmbarkhanoeva1@Redmi-Book:~$ mkdir -p ~/work/arch-pc/lab04  
rmbarkhanoeva1@Redmi-Book:~$
```

Рисунок 1.

Перешли в папку lab4 с помощью команды cd (рисунок 2).

```
rmbarkhanoeva1@Redmi-Book:~$ mkdir -p ~/work/arch-pc/lab04  
rmbarkhanoeva1@Redmi-Book:~$ cd ~/work/arch-pc/lab04  
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ █
```

Рисунок 2.

Создание файла hello.asm(рисунок 3).

```
rmbarkhanoeva1@Redmi-Book:~$ mkdir -p ~/work/arch-pc/lab04  
rmbarkhanoeva1@Redmi-Book:~$ cd ~/work/arch-pc/lab04  
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ touch hello.asm  
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ █
```

Рисунок 3.

Используя команду touch, которая может создавать файлы разных расширений мы создали файл hello.asm.

Открыла файл текстового редактора с помощью nano (рисунок 4).

```
GNU nano 7.2                                     hello.asm *
; hello.asm
SECTION .data
; Начало секции данных
hello:DB 'Hello world!',10 ; 'Hello world!' плюс
helloLen:EQU $-hello
; символ перевода строки
SECTION .text
; Длина строки hello
; 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
```

Рисунок 4.

С помощью редактора nano можно открывать файлы разных расширений. Мы вставили туда код, написанный на языке ассемблера NASM. Сохранили изменения с помощью команды Ctrl+O, подтвердили с помощью Enter и вышли из редактора Ctrl+X.

2.2. Транслятор NASM

Скомпелируем наш код (Рисунок 5).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ nasm -f elf hello.asm
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$
```

Рисунок 5.

Транслятор преобразует текст программы из файла hello.asm в объектный код, который запишется в файл hello.o. Ключ -f указывает транслятору, что требуется создать бинарные файлы в формате ELF.

2.3. Расширенный синтаксис командной строки NASM

Проверка объектного файла (рисунок 6).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ nasm -f elf hello.asm
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ █
```

Рисунок 6.

Проверили с помощью команды ls наличие объектного файла, который создали выше

Скомпелируем исходный файл hello.asm(рисунок 7).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l
list.lst hello.asm
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ █
```

Рисунок 7.

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).

2.4. Компоновщик LD

Получение исполняемого файла (рисунок 8).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$
```

Рисунок 8.

Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику ld, что мы и сделали. Ключ -о с последующим значением задаёт в данном случае имя создаваемого исполняемого файла.

Получение исполняемого файла (рисунок 9).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$
```

Рисунок 9.

С помощью компоновщика ld создали исполняемый файл main из объектного obj.o

2.4.1. Запуск исполняемого файла (рисунок 10).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ ./hello
Hello world!
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$
```

Рисунок 10.

С помощью команды ./ мы запустили исполняемый файл hello, который вывел hello world!

3. Самостоятельная работа

Создание копии (рисунок 11).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm list.lst main obj.o
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ █
```

Рисунок 11.

С помощью команды cp создали копию файла hello.asm и назвали ее lab4.asm. Команда cp перенесла скопирована все содержимое файла.

Редактирование файла(рисунок 12).

```
GNU nano 7.2                                     lab4.asm *
; hello.asm
SECTION .data
; Начало секции данных
hello:DB 'Барханоева Раяна!',10 ; 'Барханоева Раяна!' плюс
helloLen:EQU $-hello
; символ перевода строки
SECTION .text
; Длина строки hello
; 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 еcx
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
```

Рисунок 12.

С помощью редактора nano, я отредактировала файл lab4.asm так, чтобы он выводил Барханоева Раюна!.

Скомпилировали код (рисунок 13).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$
```

Рисунок 13.

Получили объектный файл lab4.o

Компиляция и компоновка (рисунок 14).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l
list.lst lab4.asm
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$
```

Рисунок 14.

Скомпилировали исходный файл lab4.asm в obj.o и передали объектный файл lab4.o компоновщику.

Запуск программы (рисунок 15).

```
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$ ./lab4
Барханоева Раюна!
rbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab04$
```

Рисунок 15.

Запустили программу и проверили, что она правильно выводит.

4. Вывод

В ходе выполнения лабораторной работы были изучены основы низкоуровневого программирования и процесс создания программ на языке ассемблера NASM. Освоены этапы получения исполняемого файла в среде Linux, включая написание исходного кода, трансляцию программы в объектный файл с помощью транслятора NASM и последующую компоновку объектного кода компоновщиком LD.