

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

### ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

дисциплина:     *Архитектура компьютера*

Студент: Павличенко Родион Андреевич

Группа: НПИбд-02-24

МОСКВА

2024 г.

## Цель работы.

Цель работы: научиться писать базовые программы на языке ассемблера NASM, компилировать их в объектные файлы и собирать из них исполняемые программы с помощью компоновщика.

## Выполнение лабораторной работы

- 1) Перед выполнением лабораторной работы необходимо создать нужную директорию с помощью команды `mkdir`

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

- 2) Теперь переместимся в созданный нами каталог

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


- 3) Теперь создадим файл `hello` с расширением `.asm`, в котором мы будем писать код на ассемблере

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

- 4) Для того, чтобы редактировать созданный файл, воспользуемся текстовым редактором `gedit`

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

- 5) Вставим в открытый файл следующий код



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

- 6) Теперь нам необходимо превратить наш файл в объектный. Этим занимается транслятор NASM. Введём следующую команду

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

- 7) Здесь мы говорим создать из файла `hello.asm` объектный, указывая при

этом формат файла elf (с помощью аргумента -f), то есть формат, работающий в системах семейства Linux. Далее проверим, создался ли объектный файл с помощью команды ls

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

8) Теперь попробуем использовать полный вариант команды NASM

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

9) Здесь мы указываем, что файл hello.asm должен быть скомпилирован в файл с названием obj.o (название указывается с помощью аргумента -o) в формате elf (аргументом -f) и включить туда символы для отладки (аргумент -g). Кроме того, мы укажем, что необходимо создать файл листинга list.lst (аргументом -l). Проверим, создался ли файл с помощью команды ls

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

10) Для создания исполняемого файла необходимо использовать компоновщик ld, который соберёт объектный файл. Напишем следующую команду

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

11) Здесь мы указываем формат elf\_i386 (с помощью аргумента -m) и файл для сборки, а аргументом -o указываем имя выходного файла. Мы назовём его hello. Проверим, создался ли файл с помощью команды ls

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

12) Теперь соберём файл obj.o в файл main

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

13) Теперь проверим, создался ли файл. Снова пропишем команду ls

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

14) Теперь запустим файл hello, для этого мы должны написать ./ и название

файла

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

## Выполнение задания для самостоятельной работы

1) Скопируем файл hello.asm в каталог ~/work/arch-pc/lab04 под названием lab4.asm

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
rapavlichenko@rapavlichenko:~/work/arch-pc/lab04$
```

2) Внесём изменения в скопированный файл. Для этого откроем его в gedit

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

3) Теперь изменим третью строку, заменив фразу Hello world! на фамилию и имя



```
*lab4.asm
~/work/arch-pc/lab04
Сохранить
1; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Pavlichenko Rodioncr',10 ; 'Hello world!' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для записи (sys_write)
10 mov ebx,1 ; Описатель файла '1' - стандартный вывод
11 mov ecx,hello ; Адрес строки hello в ecx
12 mov edx,helloLen ; Размер строки hello
13 int 80h ; Вызов ядра
14 mov eax,1 ; Системный вызов для выхода (sys_exit)
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16 int 80h ; Вызов ядра
```

4) Теперь скомпилируем полученный файл в объектный. Для этого воспользуемся командой nasm и укажем формат elf и нужный файл для компиляции

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
```

5) Теперь соберём полученный объектный файл. Укажем формат elf\_i386 и объектный файл для сборки (lab4.o). Укажем, что выходной файл должен быть назван lab4

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
rapavlichenko@rapavlichenko:~/work/arch-pc/lab04$
```

6) Убедимся в том, что сделали всё правильно. Для этого запустим собранный файл

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab04$ ./lab4
Pavlichenko Rodioncr
```

7) Теперь скопируем файл hello.asm в каталог 4 лабораторной работы

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab04$ cp hello.asm ~/work/study/2023-2024/
"Архитектура компьютера"/arch-pc/labs/lab04/
rapavlichenko@rapavlichenko:~/work/arch-pc/lab04$
```

8) Эту же операцию проведем для файла lab4.asm

```
rapavlichenko@rapavlichenko:~/work/arch-pc/lab04$ cp lab4.asm ~/work/study/2023-2024/
"Архитектура компьютера"/arch-pc/labs/lab04/
rapavlichenko@rapavlichenko:~/work/arch-pc/lab04$
```

9) Теперь загрузим результат проделанной лабораторной работы на GitHub

```
rapavlichenko@rapavlichenko:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git add .
rapavlichenko@rapavlichenko:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git commit -am "feat(main):
added files hello.asm and lab4.asm
>
>
> cd
> "
[master d99ad2c] feat(main): added files hello.asm and lab4.asm
3 files changed, 32 insertions(+)
create mode 100644 labs/lab03/report/image.zip
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
rapavlichenko@rapavlichenko:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 14, готово.
Подсчет объектов: 100% (14/14), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (9/9), готово.
Запись объектов: 100% (9/9), 1.59 МиБ | 1.48 МиБ/с, готово.
Total 9 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 3 local objects.
To github.com:Rodion-77/study_2023-2024_arhpc.git
59b6cfd..d99ad2c master -> master
rapavlichenko@rapavlichenko:~/work/study/2023-2024/Архитектура компьютера/arch-pc$
```

## Вывод

В результате выполнения лабораторной работы появилось понимание того, как работает алгоритм создания исполняемого файла из кода на ассемблере, а также появились навыки работы с языком `asm`, компиляции кода в объектный файл и сборкой исполняемых программ.