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: Абдуллахи Шугофа

Группа: НПИБД – 03 -23

МОСКВА

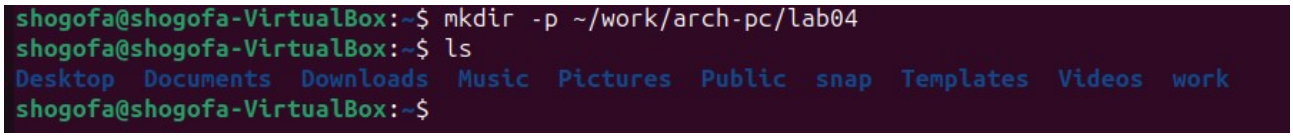
2023_ г.

Цель работы

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

Рассмотрим пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит приветственное сообщение Hello world! на экран.Создайте каталог для работы с программами на языке ассемблера NASM:

Создал каталог для работы с программами на языке ассемблера NASM.



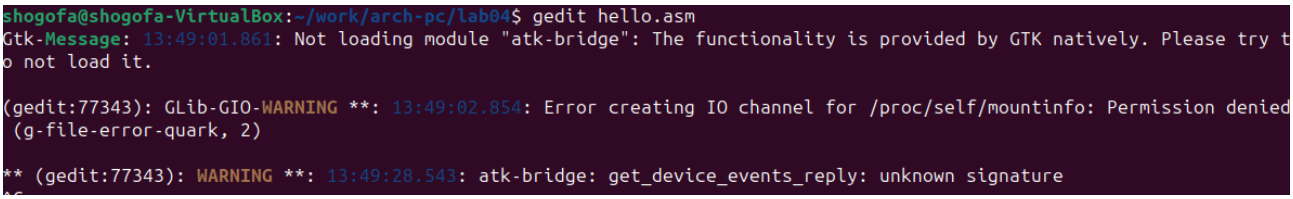
Перейдите в созданный каталог:

Создайте текстовый файл с именем hello.asm

Создал текстовый файл с именем hello.asm

откройте этот файл с помощью любого текстового редактора, например, gedit

Открыл этот файл с помощью 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)

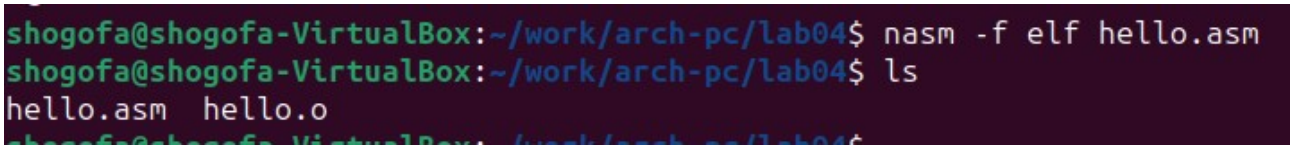
mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)

int 80h ; Вызов ядра

```
;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
```

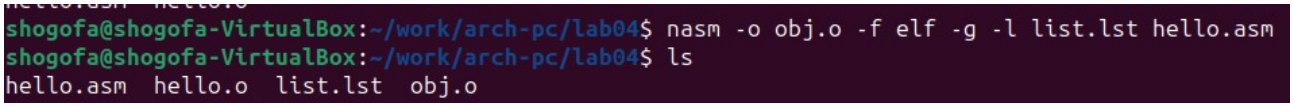
NASM превращает текст программы в объектный код. Например, для компиляции приве-дённого выше текста программы «Hello World» необходимо написать:

nasm -f elf hello.asm



Выполните следующую команду:

nasm -o obj.o -f elf -g -l list.lst hello.asm

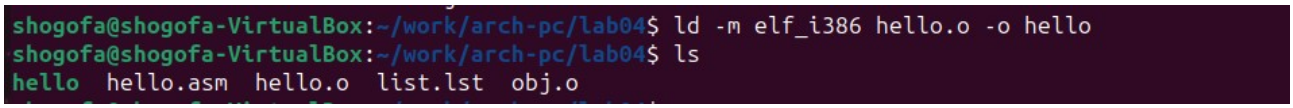


Как видно из схемы на рис. 4.3, чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику:

ld -m elf_i386 hello.o -o hello

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

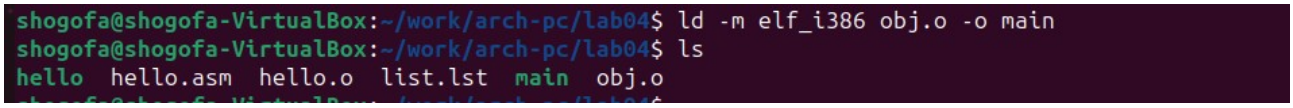
Компоновщик ld не предполагает по умолчанию расширений для файлов, но принято использовать следующие расширения:



Ключ -o с последующим значением задаёт в данном случае имя создаваемого исполняе-мого файла.

Выполните следующую команду:

ld -m elf_i386 obj.o -o main



Какое имя будет иметь исполняемый файл? Какое имя имеет объектный файл из которого собран этот исполняемый файл?

Формат командной строки LD можно увидеть, набрав ld --help. Для получения более подробной информации см. man ld.

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

./hello

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

Задание для самостоятельной работы :

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

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

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.

```
shogofa@shogofa-VirtualBox:~/work/arch-pc/lab04$ gedit hello.asm lab4.asm
Gtk-Message: 14:00:29.136: Not loading module "atk-bridge": The functionality is provided by GTK natively. Please try t
o not load it.

(gedit:78413): GLib-GIO-WARNING **: 14:00:30.071: Error creating IO channel for /proc/self/mountinfo: Permission denied
 (g-file-error-quark, 2)

** (gedit:78413): WARNING **: 14:00:55.712: atk-bridge: get_device_events_reply: unknown signature
```

```
;hello.asm

SECTION .data                       ; Начало секции данных

      hello:   DB 'shogofa Abdullahi',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
```

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

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

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.

```
shogofa@shogofa-VirtualBox:~/work/arch-pc/lab04$ cd
shogofa@shogofa-VirtualBox:~$ cp ~/work/arch-pc/lab04/hello.asm ~/work/study/2023-2024/Архитектура\ компьютера/study-20
23-2024-arh-pc/labs/lab04
shogofa@shogofa-VirtualBox:~$ cp ~/work/arch-pc/lab04/lab4.asm ~/work/study/2023-2024/Архитектура\ компьютера/study-202
3-2024-arh-pc/labs/lab04
shogofa@shogofa-VirtualBox:~$
```

```
shogofa@shogofa-VirtualBox:~$ cd ~/work/study/2023-2024/Архитектура\ компьютера/study-2023-2024-arh-pc/labs/lab04
shogofa@shogofa-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/study-2023-2024-arh-pc/labs/lab04$ git add .
shogofa@shogofa-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/study-2023-2024-arh-pc/labs/lab04$ git commit
-am 'shogofa'
[master 3d27a8c] shogofa
 2 files changed, 50 insertions(+)
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab4.asm
shogofa@shogofa-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/study-2023-2024-arh-pc/labs/lab04$ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 909 bytes | 227.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:Shogofa21/study-2023-2024-arh-pc.git
   0434f75..3d27a8c  master -> master
```