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: Леонов Никита

Группа: НКАбд-04-25

МОСКВА

2025 г.

Содержание

Снимки экрана.

1. Цель работы.

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

2. Порядок выполнения лабораторной работы

2.1. Программа Hello world!

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:

mkdir -p ~/work/arch-pc/lab04

Перейдите в созданный каталог

cd ~/work/arch-pc/lab04

Создайте текстовый файл с именем hello.asm

touch hello.asm

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

gedit hello.asm

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

В отличие от многих современных высокоуровневых языков программирования, в ассемблерной программе каждая команда располагается на отдельной строке. Размещение нескольких команд на одной строке недопустимо.

Создал каталог для работы с программами на языке ассемблера NASM и перешел в него. Создал текстовый файл с именем hello.asm и открыл этот файл с помощью текстового редактора gedit

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

1. Создание текстового файла.

ввел в него 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 0x80         ; Вызов ядра
```

2. Текст программы.

2.2. Транслятор NASM

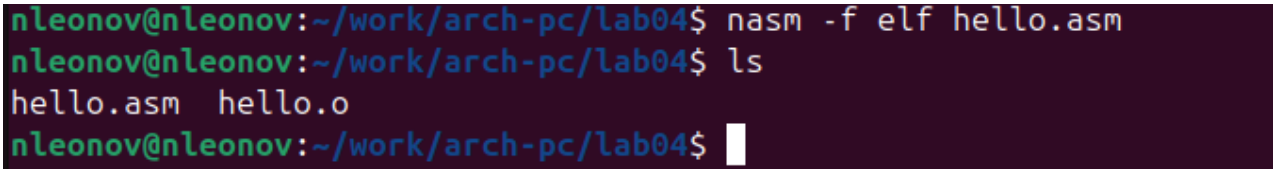
NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World» необходимо написать:

nasm -f elf hello.asm

Если текст программы набран без ошибок, то транслятор преобразует текст программы из файла hello.asm в объектный код, который запишется в файл hello.o.

Таким образом, имена всех файлов получаются из имени входного файла и расширения по умолчанию. При наличии ошибок объектный файл не создаётся, а после запуска транслятора появятся сообщения об ошибках или предупреждения. С помощью команды ls проверьте, что объектный файл был создан. Какое имя имеет объектный файл? NASM не запускают без параметров. Ключ -f указывает транслятору, что требуется создать бинарные файлы в формате ELF. Следует отметить, что формат elf64 позволяет создавать исполняемый код, работающий под 64-битными версиями Linux. Для 32-битных версий ОС указываем в качестве формата просто elf. NASM всегда создаёт выходные файлы в текущем каталоге.

Компилировал приведённый выше текст программы «Hello World» и проверил, что объектный файл был создан. Имя объектного файла “hello.o”.



3. Компилляция текста.

2.3. Расширенный синтаксис командной строки NASM

Полный вариант командной строки nasm выглядит следующим образом:

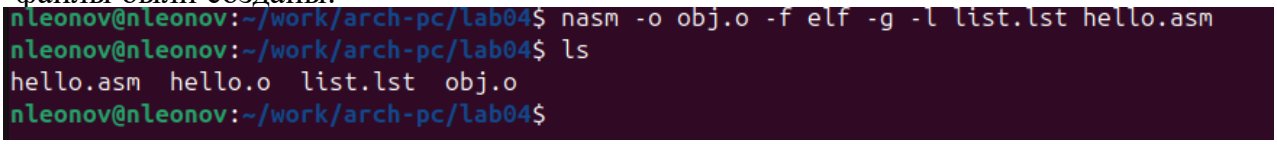
nasm [-@ косвенный_файл_настроек] [-o объектный_файл] [-f ↪ формат_объектного_файла] [-l листинг] [параметры...] [--] исходный_файл

Выполните следующую команду:

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

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). С помощью команды ls проверьте, что файлы были созданы. Для более подробной информации см. man nasm. Для получения списка форматов объектного файла см. nasm -hf.

Скомпилировала исходный файл hello.asm в obj.o с форматом elf и проверила, что файлы были созданы.



4. Компиляция файла hello.asm в obj.o

2.4. Компоновщик LD

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: • o – для объектных файлов; • без расширения – для исполняемых файлов; • map – для файлов схемы программы; • lib – для библиотек. Ключ -o с последующим значением задаёт в данном случае имя создаваемого исполняемого 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?

Передал программу на обработку компоновщику и проверил, что исполняемый файл hello был создан

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

5. Передача файла на обработку компоновщику

Выполнил указанную команду.

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

6. Передача файла на обработку компоновщику 2

Имя исполняемого файла main, а имя объектного файла obj.o.

2.5. Запуск исполняемого файла

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

./hello

Запустила на выполнение созданный исполняемый файл.

```
nleonov@nleonov:~/work/arch-pc/lab04$ ./hello
Hello world!
Segmentation fault (core dumped)
nleonov@nleonov:~/work/arch-pc/lab04$
```

7. Запуск файла.

2.6. Задание для самостоятельной работы.

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

Создал копию файла hello.asm с именем lab4.asm в каталоге ~/work/arch-pc/lab04

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

8. Копирование файла.

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.

С помощью текстового редактора внес изменения в текст программы в файле lab4.asm.

```
 GNU nano 7.2                                    lab4.asm *
; hello.asm
SECTION .data;
   hello: DB 'Leonov Nikita' , 10;
   helloLen: EQU $-hello;

SECTION .text;
    GLOBAL _start

_start:
    mov eax, 4;
    mov ebx, 1;
    mov ecx, hello;
    mov edx, helloLen;
    int 0x80
```

9. Изменение файла

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

Оттранслировал полученный текст программы lab4.asm в объектный файл, выполните компоновку объектного файла и запустила получившийся исполняемый файл.

```
nleonov@nleonov:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
nleonov@nleonov:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
nleonov@nleonov:~/work/arch-pc/lab04$ ./lab4
Leonov Nikita
Segmentation fault (core dumped)
nleonov@nleonov:~/work/arch-pc/lab04$
```

11. Вывод ФИ с помощью исполняемого файла.

4. Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study/2025-2026/"Архитектура компьютера"/arch-pc/labs/lab04/. Загрузите файлы на Github.

Скопировал файлы hello.asm и lab4.asm в свой локальный репозиторий

```
nleonov@nleonov:~/work/study/2025-2026/Arch_comp/arch-pc$ cp ~/work/arch-pc/lab04/hello.asm labs/lab04/
nleonov@nleonov:~/work/study/2025-2026/Arch_comp/arch-pc$ cp ~/work/arch-pc/lab04/lab4.asm labs/lab04/
```

12. Копирование файлов в репозиторий

Загрузил файлы на Github.

```
nleonov@nleonov:~/work/study/2025-2026/Arch_comp/arch-pc$ git add labs/lab04/
nleonov@nleonov:~/work/study/2025-2026/Arch_comp/arch-pc$ git commit -m "Add lab4: Assembly programs with personal name
[master (root-commit) 12b9610] Add lab4: Assembly programs with personal name
```

13. Загрузка файлов на Github

3. Вывод.

Я освоил процедуры компиляции и сборки программ, написанных на ассемблере NASM.