

Отчёт по лабораторной работе № 5

НММбд-02-22

Паулу Антонью Жоау

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Программа Hello world!	6
3.2	Транслятор NASM	7
3.3	Расширенный синтаксис командной строки NASM	8
3.4	Компоновщик LD	8
3.5	Запуск исполняемого файла	9
3.6	Задание для самостоятельной работы	10
4	Выводы	12

Список иллюстраций

3.1	Созданный каталог	6
3.2	Переход в каталог	6
3.3	gedit	7
3.4	Компиляция	7
3.5	Созданный объектный файл	7
3.6	obj.o	8
3.7	Созданные файлы	8
3.8	Работа компоновщика	8
3.9	Созданный файл hello	8
3.10	Компоновка файла	9
3.11	Проверка названий файлов	9
3.12	ld -help	9
3.13	Выполнение файла	9
3.14	cp lab5.asm	10
3.15	ls lab05	10
3.16	Изменения в тексте программы	10
3.17	lab5.o	10
3.18	lab5.o	11
3.19	lab5 запуск	11
3.20	hello.asm	11
3.21	lab5.asm	11
3.22	Загрузка файлов на Github	11

1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на, машинноориентированный языке низкого уровня, ассемблере NASM.

2 Задание

1. Создать файлы расширения .asm.
2. Отредактировать .asm файлы.
3. Оттранслировать .asm файлы в объектные.
4. С помощью компоновщика создать исполняемые файлы и запустить.

3 Выполнение лабораторной работы

3.1 Программа Hello world!

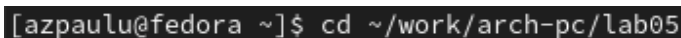
Рассмотрели пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит приветственное сообщение Hello world! на экран. Создали каталог для работы с программами на языке ассемблера NASM: (рис. 3.1)



```
[azpaulu@fedora ~]$ mkdir -p ~/work/arch-pc/lab05
```

Рис. 3.1: Созданный каталог

Перешли в созданный каталог. (рис. 3.2)



```
[azpaulu@fedora ~]$ cd ~/work/arch-pc/lab05
```

Рис. 3.2: Переход в каталог

Создали текстовый файл с именем hello.asm. Открыли этот файл с помощью текстового редактора gedit. Введите в него следующий текст: (рис. 3.3)

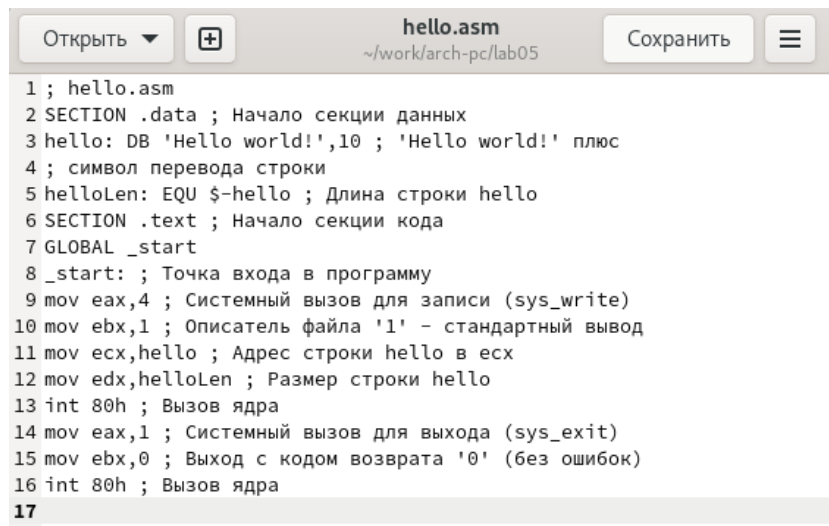


Рис. 3.3: gedit

3.2 Транслятор NASM

NASM превращает текст программы в объектный код. Для компиляции текста программы «Hello World» написали: (рис. 3.4)

```
[azpaulu@fedora lab05]$ nasm -f elf hello.asm
```

Рис. 3.4: Компиляция

С помощью транслятора преобразовали текст программы из файла hello.asm в объектный код, который записали в файл hello.o С помощью команды ls проверили, что объектный файл был создан. Объектный файл имеет имя hello.o (рис. 3.5)

```
[azpaulu@fedora lab05]$ ls
hello.asm  hello.o
```

Рис. 3.5: Созданный объектный файл

3.3 Расширенный синтаксис командной строки NASM

Выполнили следующую команду: (рис. 3.6)

```
[azpaulu@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
```

Рис. 3.6: obj.o

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. (рис. 3.7)

```
[azpaulu@fedora lab05]$ ls
hello.asm  hello.o  list.lst  obj.o
```

Рис. 3.7: Созданные файлы

3.4 Компоновщик LD

Для получения исполняемой программы, объектный файл передали на обработку компоновщику: (рис. 3.8)

```
[azpaulu@fedora lab05]$ ld -m elf_i386 hello.o -o hello
```

Рис. 3.8: Работа компоновщика

С помощью команды `ls` проверьте, что исполняемый файл `hello` был создан. (рис. 3.9)

```
[azpaulu@fedora lab05]$ ls
hello  hello.asm  hello.o  list.lst  obj.o
```

Рис. 3.9: Созданный файл hello

Выполните следующую команду: (рис. 3.10)


```
[azpaulu@fedora lab05]$ ld -m elf_i386 obj.o -o main
```

Рис. 3.10: Компоновка файла

Исполняемый файл будет иметь имя `main`. Объектный файл, из которого собран этот исполняемый файл, будет иметь имя `main.o` (рис. 3.11)

```
[azpaulu@fedora lab05]$ ls
hello hello.asm hello.o list.lst main obj.o
```

Рис. 3.11: Проверка названий файлов

Формат командной строки LD увидели, набрав `ld --help`. (рис. 3.12)

```
[azpaulu@fedora lab05]$ ld --help
Использование ld [параметры] файл...
Параметры:
-a КЛЮЧЕВОЕ СЛОВО                Управление общей библиотекой для совместимости с HP/UX
-A АРХИТЕКТУРА, --architecture АРХИТЕКТУРА    Задать архитектуру
-b ЦЕЛЬ, --format ЦЕЛЬ            Задать цель для следующих входных файлов
-c ФАЙЛ, --mri-script ФАЙЛ        Прочитать сценарий компоновщика в формате MRI
-d, -dc, -dp                     Принудительно делать общие символы определёнными
--dependency-file ФАЙЛ           Write dependency file
--force-group-allocation         Принудительно удалить членов группы из групп
-e АДРЕС, --entry АДРЕС          Задать начальный адрес
-E, --export-dynamic             Экспортировать все динамические символы
--no-export-dynamic              Отменить действие --export-dynamic
--enable-non-contiguous-regions  Enable support of non-contiguous memory regions
--enable-non-contiguous-regions-warnings      Enable warnings when --enable-non-contiguous-regions ma
ause unexpected behaviour
-EB                               Компоновать объекты с прямым порядком байтов
-EL                               Компоновать объекты с обратным порядком байтов
-f SHLIB, --auxiliary SHLIB      Вспомогательный фильтр таблицы символов общих объектов
-F SHLIB, --filter SHLIB         Фильтр для таблицы символов общих объектов
```

Рис. 3.12: `ld --help`

3.5 Запуск исполняемого файла

Запустили на выполнение созданный исполняемый файл, находящийся в текущем каталоге. (рис. 3.13)

```
[azpaulu@fedora lab05]$ ./hello
Hello world!
```

Рис. 3.13: Выполнение файла

3.6 Задание для самостоятельной работы

1. В каталоге ~/work/arch-pc/lab05 с помощью команды cp создали копию файла hello.asm с именем lab5.asm (рис. 3.14), (рис. 3.15)

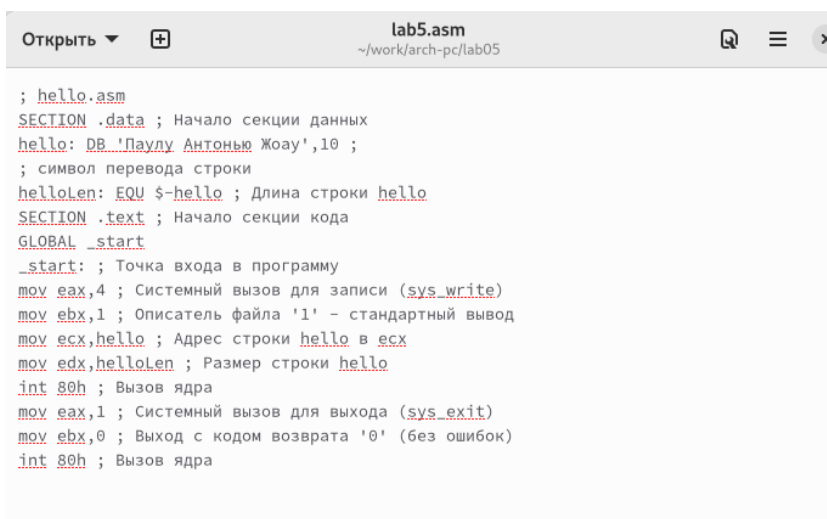
```
[azpaulu@fedora lab05]$ cp hello.asm lab5.asm
```

Рис. 3.14: cp lab5.asm

```
[azpaulu@fedora lab05]$ ls  
hello hello.asm hello.o lab5.asm list.lst main obj.o
```

Рис. 3.15: ls lab05

2. С помощью текстового редактора внесли изменения в текст программы в файле lab5.asm так, чтобы вместо Hello world! на экран выводилась строка с фамилией и именем. (рис. 3.16)



```
Открыть ▾ + lab5.asm  
~/work/arch-pc/lab05  
; hello.asm  
SECTION .data ; Начало секции данных  
hello: DB 'Паулу Анто́нью Жоау',10 ;  
; 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
```

Рис. 3.16: Изменения в тексте программы

3. Оттранслировали полученный текст программы lab5.asm в объектный файл. (рис. 3.17)

```
[azpaulu@fedora lab05]$ nasm -f elf lab5.asm
```

Рис. 3.17: lab5.o

Выполнили компоновку объектного файла и запустили получившийся исполняемый файл. (рис. 3.18), (рис. 3.19)

```
[azpaulu@fedora lab05]$ ld -m elf_i386 lab5.o -o lab5
```

Рис. 3.18: lab5.o

```
[azpaulu@fedora lab05]$ ./lab5  
Паулу Антонию Жоау
```

Рис. 3.19: lab5 запуск

4. Скопируйте файлы hello.asm и lab5.asm в Ваш локальный репозиторий в каталог ~/work/study/2022-2023/“Архитектура компьютера”/archpc/labs/lab05/. (рис. 3.20), (рис. 3.21)

```
[azpaulu@fedora lab05]$ cp lab5.asm ~/work/study/2022-2023_2023_arh-pc/labs/lab05
```

Рис. 3.20: hello.asm

```
[azpaulu@fedora lab05]$ cp hello.asm ~/work/study/2022-2023_2023_arh-pc/labs/lab05
```

Рис. 3.21: lab5.asm

Загрузите файлы на Github. (рис. 3.22)

```
[azpaulu@fedora study_2022-2023_arh-pc]$ git add .  
[azpaulu@fedora study_2022-2023_arh-pc]$ git commit -am 'feat(main): make course structure'  
[master 5f62c3b] feat(main): make course structure  
3 files changed, 236 insertions(+), 119 deletions(-)  
create mode 100644 labs/lab05/hello.asm  
create mode 100644 labs/lab05/lab5.asm  
rewrite labs/lab05/report/report.md (72%)  
[azpaulu@fedora study_2022-2023_arh-pc]$ git push  
Перечисление объектов: 13, готово.  
Подсчет объектов: 100% (13/13), готово.  
При сжатии изменений используется до 3 потоков  
Сжатие объектов: 100% (8/8), готово.  
Запись объектов: 100% (8/8), 3.76 КиБ | 3.76 МиБ/с, готово.  
Всего 8 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0  
remote: Resolving deltas: 100% (4/4), completed with 3 local objects.  
To github.com:antoniopaulo11/study_2022-2023_arh-pc.git  
79d7cd0..5f62c3b master -> master
```

Рис. 3.22: Загрузка файлов на Github

4 Выводы

В ходе лабораторной работы были освоены процедуры компиляции и сборки программ, написанных на машинноориентированном языке низкого уровня, ассемблере NASM.