

отчёт по лабораторной работе № 5

Архитектура компьютера

Толстых Максим Алексеевич

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	7
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	8
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.

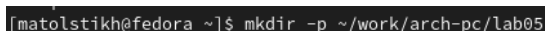
2 Задание

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!

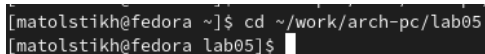
Рассмотрели пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит приветственное сообщение Hello world! на экран. Создали каталог для работы с программами на языке ассемблера NASM: (рис. 3.1)



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

Рис. 3.1: Созданный каталог

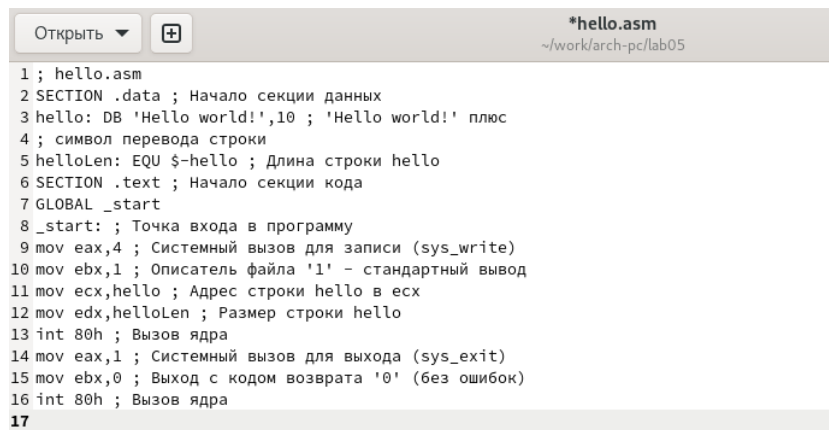
Перешли в созданный каталог. (рис. 3.2)



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

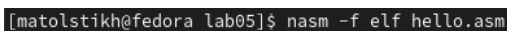


```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для 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
17
```

Рис. 3.3: gedit

3.2 Транслятор NASM

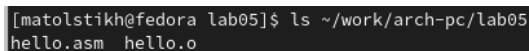
NASM превращает текст программы в объектный код. Для компиляции текста программы «Hello World» написали: (рис. 3.4)



```
[matolstikh@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)



```
[matolstikh@fedora lab05]$ ls ~/.work/arch-pc/lab05
hello.asm hello.o
```

Рис. 3.5: Созданный объектный файл

3.3 Расширенный синтаксис командной строки NASM

Выполнили следующую команду: (рис. 3.6)

```
[matolstikh@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)

```
[matolstikh@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)

```
[matolstikh@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)

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

Рис. 3.9: Созданный файл hello

Выполните следующую команду: (рис. 3.10)

```
[matolstikh@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)


```
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)

```
[matolstikh@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
may cause 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 объек
ов
-F SHLIB, --filter SHLIB                          Фильтр для таблицы символов общих объектов
-g                                                 Игнорируется
-G РАЗМЕР, --gpsize РАЗМЕР                         Размер маленьких данных (если не указан, то берётся
из --shared)
-h ИМЯ_ФАЙЛА, -soname ИМЯ_ФАЙЛА                  Задать внутреннее имя общей библиотеки
-I ПРОГРАММА, --dynamic-linker ПРОГРАММА          Назначить ПРОГРАММУ в качестве используемого динамич
еского компоновщика
```

Рис. 3.12: `ld --help`

3.5 Запуск исполняемого файла

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

```
[matolstikh@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)

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

Рис. 3.14: `cp lab5.asm`

```
[matolstikh@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 и скромной самооценкой. (рис. 3.16)

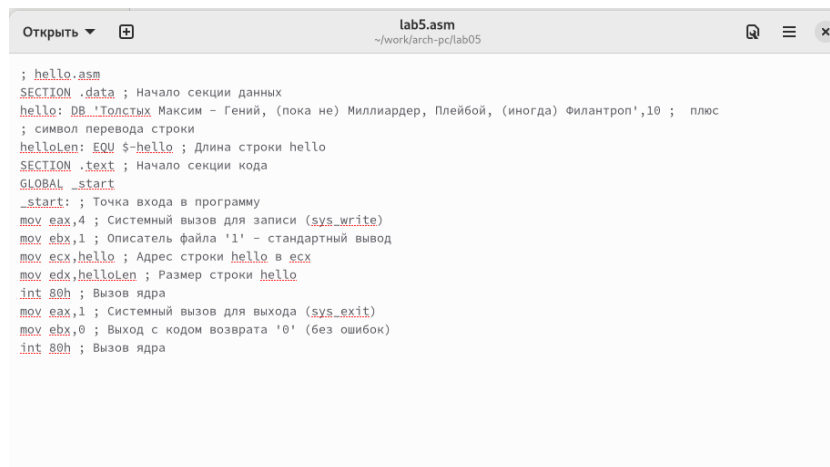


Рис. 3.16: Изменения в тексте программы

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

```
[matolstikh@fedora lab05]$ nasm -o lab5.o -f elf -g -l list.lst lab5.asm
```

Рис. 3.17: `lab5.o`

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

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

Рис. 3.18: lab5.o

```
[matolstikh@fedora lab05]$ ./lab5  
Толстых Максим – Гений, (пока не) Миллиардер, Плейбой, (иногда) Филантроп
```

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)

```
[matolstikh@fedora lab05]$ cp hello.asm ~/work/study/2022-2023/"Архитектура  
а компьютера"/study_2022-2023_arh-pc/labs/lab05/
```

Рис. 3.20: hello.asm

```
[matolstikh@fedora lab05]$ cp lab5.asm ~/work/study/2022-2023/"Архитектура  
компьютера"/study_2022-2023_arh-pc/labs/lab05/
```

Рис. 3.21: lab5.asm

Загрузите файлы на Github. (рис. 3.22)

```
[matolstikh@fedora report]$ cd ~/work/study/2022-2023/"Архитектура компьт  
ера"/study_2022-2023_arh-pc/labs/lab05  
[matolstikh@fedora lab05]$ git add .  
[matolstikh@fedora lab05]$ git commit -am 'feat(main): add files lab-5'  
[master e109538] feat(main): add files lab-5  
2 files changed, 34 insertions(+)  
create mode 100644 labs/lab05/hello.asm  
create mode 100644 labs/lab05/lab5.asm  
[matolstikh@fedora lab05]$ git push  
Перечисление объектов: 9, готово.  
Подсчет объектов: 100% (9/9), готово.  
При сжатии изменений используется до 4 потоков  
Сжатие объектов: 100% (6/6), готово.  
Запись объектов: 100% (6/6), 1.02 КиБ | 349.00 КиБ/с, готово.  
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно исп  
ользовано пакетов 0  
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.  
To github.com:Frostoslav/study_2022-2023_arh-pc.git  
8737891..e109538 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.