

Отчет по Лабораторной работе №4

Архитектура компьютеров и операционные системы

Бекауов Артур Тимурович НКАбд-01-23

Содержание

1	Цель работы	5
2	Ход лабораторной работы	6
3	Ход самостоятельной работы	10
4	Выводы	12

Список иллюстраций

2.1	Подготовка к началу работы	6
2.2	Текст программы hello.asm	7
2.3	Создание объектного файла из текстовой программы с помощью NASM	7
2.4	Создание объектного файла с дополнительными опциями	8
2.5	Создание исполняемой программы из объектного кода . .	8
2.6	Создание исполняемой программы из объектного кода с переименованием	8
2.7	Запуск исполняемого файла	9
3.1	CP1: Копирование текстовой программы с переименованием	10
3.2	CP2: Изменение текстовой программы “lab04.asm”	10
3.3	Процесс получения исполняемой программы “lab04” и её вывод	11
3.4	Копирование файлов и Загрузка изменений на github . . .	11

Список таблиц

1 Цель работы

Целью работы является освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM

2 Ход лабораторной работы

А. Программа Hello world!

Сначала открываю терминал и создаю в каталоге курса папку для работы с программами на языке ассемблера NASM, перехожу в созданный каталог и создаю текстовый файл с именем “hello.asm”. Открываю созданный файл с помощью gedit (Рис. 2.1) ввожу в него предложенный текст программы (Рис. 2.2)

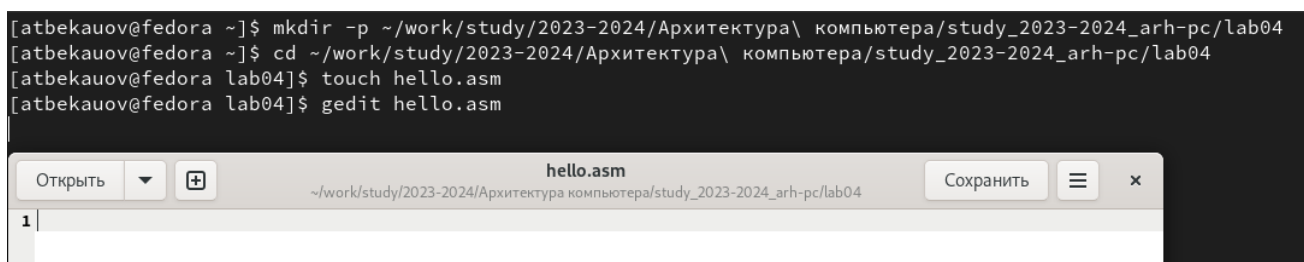


Рис. 2.1: Подготовка к началу работы

```
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
```

Рис. 2.2: Текст программы hello.asm

В. Транслятор NASM

Затем превращу текст программы в объектный код. Проверю содержимое папки командой `ls`, вижу, что из текстового файла “hello.asm” был создан объектный файл “hello.o” (Рис. 2.3).

```
[atbekauov@fedora lab04]$ nasm -f elf hello.asm
[atbekauov@fedora lab04]$ ls
hello.asm  hello.o
[atbekauov@fedora lab04]$ |
```

Рис. 2.3: Создание объектного файла из текстовой программы с помощью NASM

С. Расширенный синтаксис командной строки NASM

Далее ввожу команду “`nas, -o obj.o -f elf -g -l list.lst hello.asm`”. Проверю результаты её выполнения с помощью `ls`. Итак, команда создала из текстового файла “hello.asm” объектный файл, названный “obj.o” (-o obj.o)

с форматом elf (-f elf), в который будут также включены символы для отладки (-g). Помимо этого был создан файл листинга list.lst (-l list.lst) (Рис. 2.4).

```
[atbekauov@fedora lab04]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[atbekauov@fedora lab04]$ ls
hello.asm  hello.o  list.lst  obj.o
[atbekauov@fedora lab04]$ |
```

Рис. 2.4: Создание объектного файла с дополнительными опциями

D. Компоновщик LD

Передам объектный файл “hello.o” на обработку компоновщику, чтобы получить исполняемую программу “hello”. Проверю результаты работы компоновщика с помощью ls. (Рис. 2.5).

```
[atbekauov@fedora lab04]$ ld -m elf_i386 hello.o -o hello
[atbekauov@fedora lab04]$ ls
hello  hello.asm  hello.o  list.lst  obj.o
[atbekauov@fedora lab04]$
```

Рис. 2.5: Создание исполняемой программы из объектного кода

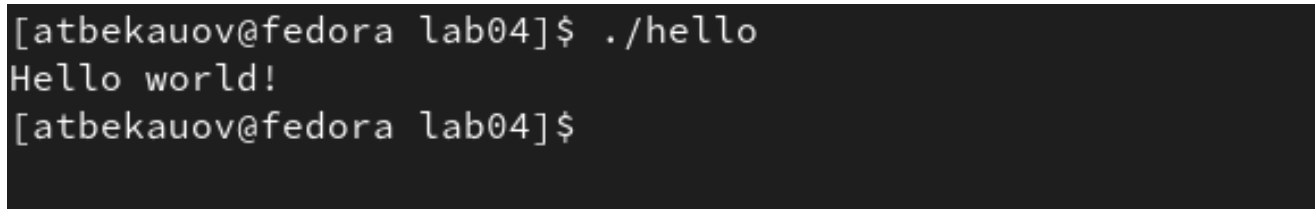
Далее выполню следующую команду “ld -m elf_i386 obj.o -o main”, которая создаст исполняемый файл “main” (-o main) из объектного файла “obj.o”. Проверю с помощью ls (Рис. 2.6).

```
[atbekauov@fedora lab04]$ ld -m elf_i386 obj.o -o main
[atbekauov@fedora lab04]$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
[atbekauov@fedora lab04]$
```

Рис. 2.6: Создание исполняемой программы из объектного кода с переименованием

Е. Запуск исполняемого файла

Запущу на исполнение созданный исполняемый файл. В результате на экран выведено сообщение “Hello world!” (Рис. 2.7).

A terminal window with a dark background. The prompt is [atbekauov@fedora lab04]\$. The user enters ./hello. The output is Hello world!. The prompt returns to [atbekauov@fedora lab04]\$.

```
[atbekauov@fedora lab04]$ ./hello
Hello world!
[atbekauov@fedora lab04]$
```

Рис. 2.7: Запуск исполняемого файла

3 Ход самостоятельной работы

1

Командой `ls` покажу изначальное содержание папки. С помощью команды `cp` создам копию файла `hello.asm` с именем `lab4.asm`. Продемонстрирую результаты копирования командой `ls` (Рис. 3.1)

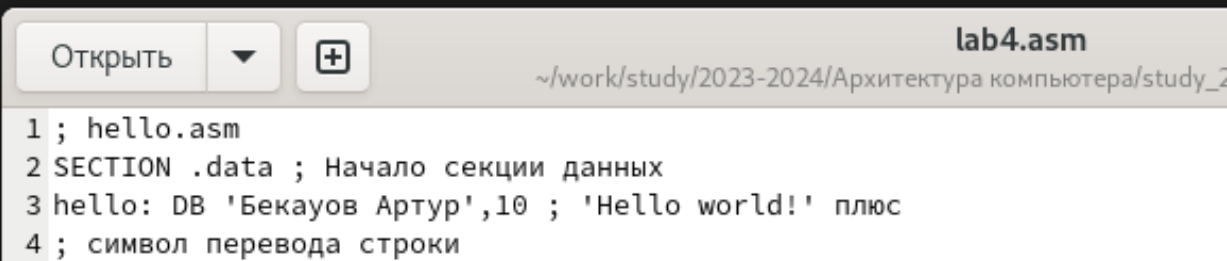
```
[atbekauov@fedora lab04]$ ls
hello hello.asm hello.o list.lst main obj.o
[atbekauov@fedora lab04]$ cp hello.asm lab4.asm
[atbekauov@fedora lab04]$ ls
hello hello.asm hello.o lab4.asm list.lst main obj.o
[atbekauov@fedora lab04]$ |
```

Рис. 3.1: CP1: Копирование текстовой программы с переименованием

2

Открою `lab4.asm` в `gedit`. Изменяю текст программы так, чтобы программа выводила моё имя и фамилию. (Рис. 3.2)

```
[atbekauov@fedora lab04]$ gedit lab4.asm
```



```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Бекаюв Артур',10 ; 'Hello world!' плюс
4 ; символ перевода строки
```

Рис. 3.2: CP2: Изменение текстовой программы `lab04.asm`

3

Текстовую программу “lab4.asm” с помощью NASM переведу в объектный код, который передам на обработку компоновщику LD. На выходе получил исполняемую программу lab4, которую запущу через терминал. Программа вывела на экран моё имя и фамилию (Рис 3.3)

```
[atbekauov@fedora lab04]$ nasm -f elf lab4.asm
[atbekauov@fedora lab04]$ ld -m elf_i386 lab4.o -o lab4
[atbekauov@fedora lab04]$ ./lab4
Бекауов Артур
[atbekauov@fedora lab04]$ |
```

Рис. 3.3: Процесс получения исполняемой программы “lab04” и её вывод

4

Скопирую файлы “hello.asm” и “lab4.asm” в свой локальный репозиторий папку для лабораторной работы №4 (Рис. 3.4) Затем загружаю все сделанные изменения на github (Рис. 3.4).

```
[atbekauov@fedora lab04]$ cp hello.asm ~/work/study/2023-2024/Архитектура\ компьютера/study_2023-2024_arh-pc/labs/lab04
[atbekauov@fedora lab04]$ cp lab4.asm ~/work/study/2023-2024/Архитектура\ компьютера/study_2023-2024_arh-pc/labs/lab04
[atbekauov@fedora lab04]$ git add .
[atbekauov@fedora lab04]$ git commit -am 'Lab4 - first commit'
[master 0cb87de] Lab4 - first commit
9 files changed, 49 insertions(+)
create mode 100755 lab04/hello
create mode 100644 lab04/hello.asm
create mode 100644 lab04/hello.o
create mode 100755 lab04/lab4
create mode 100644 lab04/lab4.asm
create mode 100644 lab04/lab4.o
create mode 100644 lab04/list.lst
create mode 100755 lab04/main
create mode 100644 lab04/obj.o
[atbekauov@fedora lab04]$ git push
Перечисление объектов: 13, готово.
Подсчет объектов: 100% (13/13), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (12/12), готово.
Запись объектов: 100% (12/12), 3.19 КиБ | 3.19 МиБ/с, готово.
Всего 12 (изменений 6), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (6/6), completed with 1 local object.
To github.com:atbekauov/study_2023-2024_arh-pc.git
8022ec0..0cb87de master -> master
[atbekauov@fedora lab04]$ |
```

Рис. 3.4: Копирование файлов и Загрузка изменений на github

4 Выводы

В ходе лабораторной работы я освоил процедуры компиляции и сборки программ, написанных на ассемблере NASM.