

Отчет по лабораторной работе №4

Архитектура компьютера и операционные системы

Александр Дмитриевич Собко

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Программа Hello world!	8
4.2	Транслятор NASM	9
4.3	Расширенный синтаксис командной строки NASM	9
4.4	Компоновщик LD	10
4.5	Запуск исполняемого файла	11
5	Задание для самостоятельной работы	12
6	Выводы	14
	Список литературы	15

Список иллюстраций

4.1	Рисунок1	8
4.2	Рисунок2	9
4.3	Рисунок3	10
4.4	Рисунок4	10
4.5	Рисунок5	11
5.1	Рисунок6	12
5.2	Рисунок7	12
5.3	Рисунок8	13

Список таблиц

1 Цель работы

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

2 Задание

Написание, компиляция и запуск первых программ написанных на ассемблере

3 Теоретическое введение

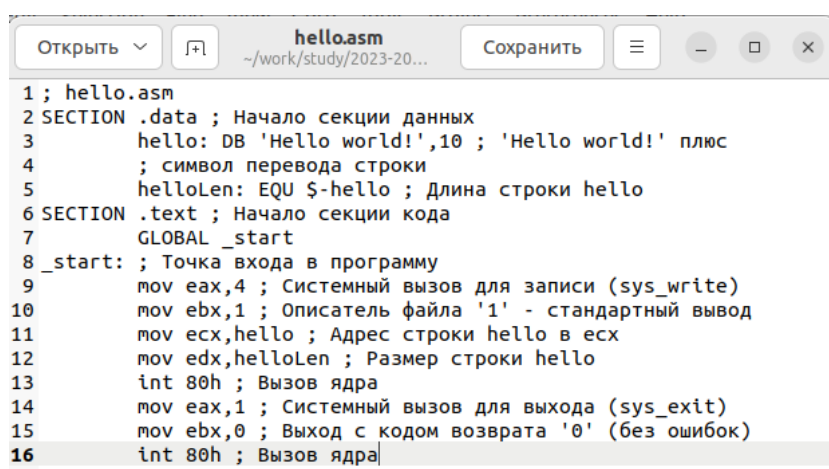
Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства (рис. 4.1). Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих дорожек на материнской (системной) плате.

Язык ассемблера (*assembly language*, сокращённо *asm*) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня, таких как C/C++, Perl, Python и пр. Заметим, что получить полный доступ к ресурсам компьютера в современных архитектурах нельзя, самым низким уровнем работы прикладной программы является обращение напрямую к ядру операционной системы. Именно на этом уровне и работают программы, написанные на ассемблере. Но в отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввёл программист. Таким образом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора

4 Выполнение лабораторной работы

4.1 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. Перейдите в созданный каталог Создайте текстовый файл с именем hello.asm откройте этот файл с помощью любого текстового редактора, например, gedit и введите в него следующий текст:



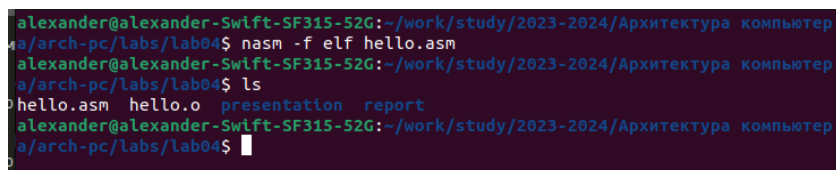
```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3     hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4             ; символ перевода строки
5     hellolen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7     GLOBAL _start
8 _start: ; Точка входа в программу
9         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. 4.1: Рисунок1

4.2 Транслятор NASM

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

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?



```
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютера
a/arch-pc/labs/lab04$ nasm -f elf hello.asm
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютера
a/arch-pc/labs/lab04$ ls
hello.asm hello.o presentation report
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютера
a/arch-pc/labs/lab04$
```

Рис. 4.2: Рисунок2

4.3 Расширенный синтаксис командной строки NASM

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`.

```
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/labs/lab04$ ls
hello.asm hello.o list.lst obj.o presentation report
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/labs/lab04$
```

Рис. 4.3: Рисунок3

4.4 Компоновщик LD

Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику:

```
ld -m elf_i386 hello.o -o hello
```

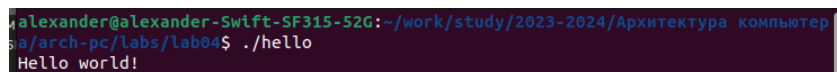
С помощью команды `ls` проверьте, что исполняемый файл `hello` был создан. Компоновщик `ld` не предполагает по умолчанию расширений для файлов, но принято использовать следующие расширения: • `o` – для объектных файлов; • без расширения – для исполняемых файлов; • `map` – для файлов схемы программы; • `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
```

```
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/labs/lab04$ ls
hello hello.asm hello.o list.lst obj.o presentation report
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/labs/lab04$ ld -m elf_i386 obj.o -o main
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютер
a/arch-pc/labs/lab04$ ls
hello hello.asm hello.o list.lst main obj.o presentation report
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютер
```

Рис. 4.4: Рисунок4

4.5 Запуск исполняемого файла

A terminal window with a dark background. The prompt is 'alexander@alexander-Swift-SF315-52G:~/work/study/2023-2024/Архитектура_компьютер'. The user enters 'a/arch-pc/labs/lab04\$./hello'. The output is 'Hello world!'.

```
alexander@alexander-Swift-SF315-52G:~/work/study/2023-2024/Архитектура_компьютер
a/arch-pc/labs/lab04$ ./hello
Hello world!
```

Рис. 4.5: Рисунок5

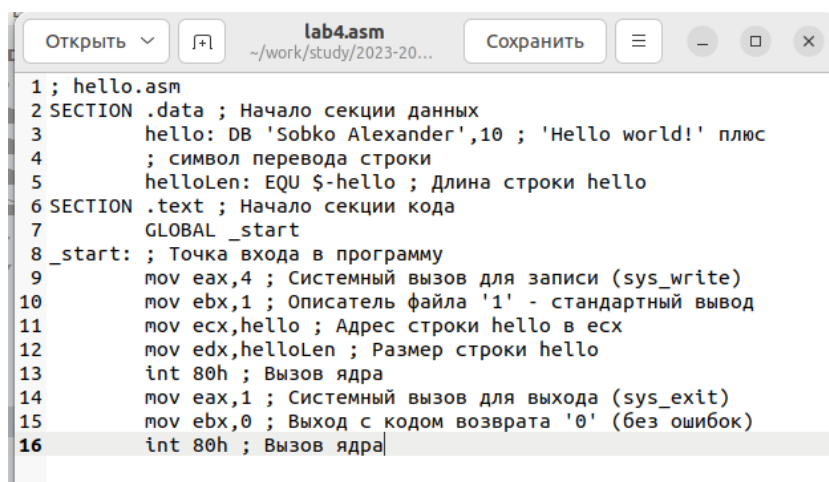
5 Задание для самостоятельной работы

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

```
a/arch-pc/labs/lab04$ cp hello.asm lab4.asm
alexander@alexander-Swift-SF315-52G:~/work/study/2023-2024/Архитектура компьютера
a/arch-pc/labs/lab04$ ls
hello      hello.o    list.lst  obj.o      report
hello.asm  lab4.asm  main      presentation
```

Рис. 5.1: Рисунок6

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.



```
Открыть  lab4.asm  Сохранить
~/work/study/2023-20...

1 ; hello.asm
2 SECTION .data ; Начало секции данных
3     hello: DB 'Sobko Alexander',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. 5.2: Рисунок7

3. Оттранслируйте полученный текст программы lab4.asm в объектный файл.

Выполните компоновку объектного файла и запустите получившийся исполняемый файл

```
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютера
a/arch-pc/labs/lab04$ nasm -f elf lab4.asm
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютера
a/arch-pc/labs/lab04$ ls
hello      hello.o    lab4.o     main      presentation
hello.asm  lab4.asm  list.lst  obj.o     report
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютера
a/arch-pc/labs/lab04$ ld -m elf_i386 lab4.o -o lab4
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютера
a/arch-pc/labs/lab04$ ls
hello      hello.o    lab4.asm  list.lst  obj.o     report
hello.asm  lab4      lab4.o    main      presentation
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютера
a/arch-pc/labs/lab04$ ./lab4
Sobko Alexander
alexander@alexander-Swift-SF315-S2G:~/work/study/2023-2024/Архитектура компьютера
a/arch-pc/labs/lab04$
```

Рис. 5.3: Рисунок8

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.

6 Выводы

Узнали основные этапы получения исполняемого файла и сделали его.

Список литературы