

Отчёт по лабораторной работе №4

дисциплина: Архитектура вычислительных систем

Мосолов Александр Денисович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Программа Hello world!	6
2.2	Транслятор NASM	7
2.3	Расширенный синтаксис командной строки NASM	7
2.4	Компоновщик LD	8
2.5	Запуск исполняемого файла	8
2.6	Задание для самостоятельной работы	8
3	Выводы	11

Список иллюстраций

2.1	Создаем каталог для работы	6
2.2	Asm-файл	6
2.3	Открываем файл	6
2.4	Вводим текст	7
2.5	Компилируем текст	7
2.6	Присваиваем имя скомпилированному файлу и создаем файл ли- стинга	7
2.7	Передаем файл hello.o на обработку	8
2.8	Передаем файл obj.o на обработку	8
2.9	Запускаем исполняемый файл	8
2.10	Копируем файл hello.asm	8
2.11	Вносим изменения в текст программы	9
2.12	Транслируем текст	9
2.13	Переносим файлы	9
2.14	Загрузка файлов на Github	10

Список таблиц

1 Цель работы

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

2 Выполнение 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.

Создаем каталог для работы с программами на языке ассемблера *NASM* (рис. [2.1]):

```
admosolov@admosolov-VirtualBox:~$ mkdir -p ~/work/arch-pc/lab04
```

Рис. 2.1: Создаем каталог для работы

Переходим в каталог *lab4* и создаем в нем файл с именем *hello.asm*.

```
admosolov@admosolov-VirtualBox:~$ cd ~/work/arch-pc/lab04
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ touch hello.asm
```

Рис. 2.2: Asm-файл

Открываем этот файл с помощью текстового редактора *gedit*.

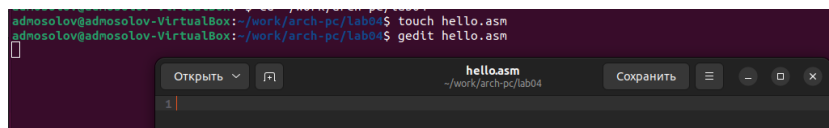
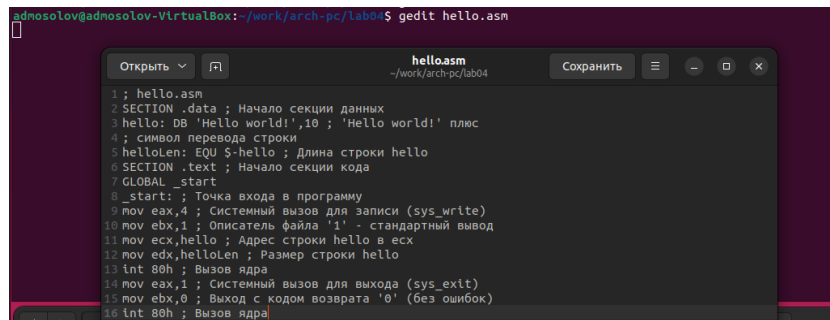


Рис. 2.3: Открываем файл

Вводим в него текст для последующей компиляции.

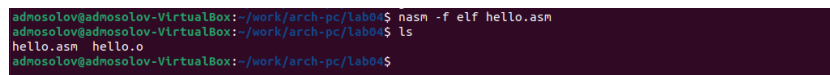


```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ gedit hello.asm
hello.asm
~/work/arch-pc/lab04
Сохранить
1; hello.asm
2SECTION .data ; Начало секции данных
3hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4; символ перевода строки
5helloLen: EQU $-hello ; Длина строки hello
6SECTION .text ; Начало секции кода
7GLOBAL _start
8_start: ; Точка входа в программу
9mov eax,4 ; Системный вызов для записи (sys_write)
10mov ebx,1 ; Описатель файла '1' - стандартный вывод
11mov ecx,hello ; Адрес строки hello в ecx
12mov edx,helloLen ; Размер строки hello
13int 80h ; Вызов ядра
14mov eax,1 ; Системный вызов для выхода (sys_exit)
15mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16int 80h ; Вызов ядра
```

Рис. 2.4: Вводим текст

2.2 Транслятор NASM

NASM превращает текст программы в объектный код. Компилируем приведённый выше текст программы, проверяем наличие нового файла *hello.o*.



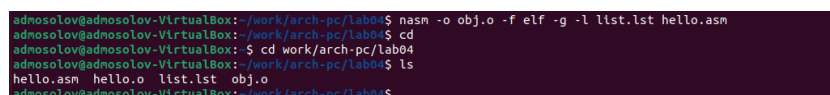
```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ nasm -f elf hello.asm
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$
```

Рис. 2.5: Компилируем текст

2.3 Расширенный синтаксис командной строки NASM

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*).

Проверяем существование созданных файлов с помощью команды *ls*.



```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ cd
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$
```

Рис. 2.6: Присваиваем имя скомпилированному файлу и создаем файл листинга

2.4 Компоновщик LD

Объектный файл необходимо передать на обработку компоновщику.

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

Рис. 2.7: Передаем файл hello.o на обработку

Для тренировки отправляем на обработку компоновщику файл *obj.o* и называем его *main*.

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

Рис. 2.8: Передаем файл obj.o на обработку

2.5 Запуск исполняемого файла

Запустить на выполнение созданный исполняемый файл, находящийся в текущем каталоге, можно, набрав в командной строке *./hello*.

```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ ./hello
Hello world!
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$
```

Рис. 2.9: Запускаем исполняемый файл

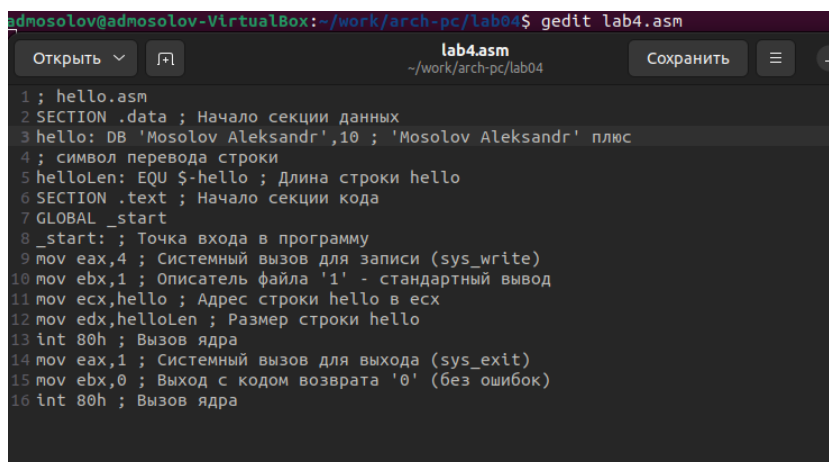
2.6 Задание для самостоятельной работы

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

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

Рис. 2.10: Копируем файл hello.asm

С помощью текстового редактора вносим изменения в текст программы в файле *lab4.asm* так, чтобы вместо *Hello world!* на экран выводилась строка с фамилией и именем.

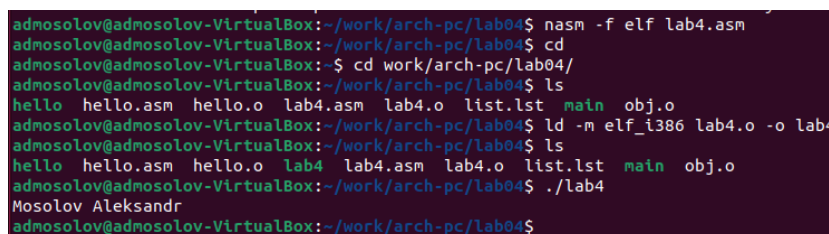


```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ gedit lab4.asm
lab4.asm
~/work/arch-pc/lab04
Сохранить

1; hello.asm
2SECTION .data ; Начало секции данных
3hello: DB 'Mosolov Aleksandr',10 ; 'Mosolov Aleksandr' плюс
4; символ перевода строки
5helloLen: EQU $-hello ; Длина строки hello
6SECTION .text ; Начало секции кода
7GLOBAL _start
8_start: ; Точка входа в программу
9mov eax,4 ; Системный вызов для записи (sys_write)
10mov ebx,1 ; Описатель файла '1' - стандартный вывод
11mov ecx,hello ; Адрес строки hello в ecx
12mov edx,helloLen ; Размер строки hello
13int 80h ; Вызов ядра
14mov eax,1 ; Системный вызов для выхода (sys_exit)
15mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16int 80h ; Вызов ядра
```

Рис. 2.11: Вносим изменения в текст программы

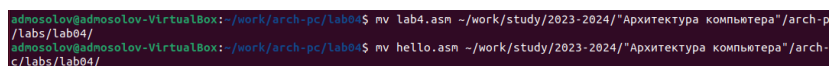
Транслируем полученный текст программы *lab4.asm* в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл.



```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ cd
admosolov@admosolov-VirtualBox:~$ cd work/arch-pc/lab04/
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ ls
hello hello.o lab4.asm lab4.o list.lst main obj.o
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ ls
hello hello.o lab4 lab4.asm lab4.o list.lst main obj.o
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ ./lab4
Mosolov Aleksandr
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$
```

Рис. 2.12: Транслируем текст

Переносим файлы *hello.asm* и *lab4.asm* в локальный репозиторий в каталог `~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/`.



```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ mv lab4.asm ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab04$ mv hello.asm ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/
```

Рис. 2.13: Переносим файлы

Загружаем файлы на *Github*.

```
admosolov@admosolov-VirtualBox: /work/arch-pc/lab04$ cd ~/work/study/2023-2024/Архитектура компьютера/arch-pc/
admosolov@admosolov-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git pull
Уже актуально.
admosolov@admosolov-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git add .
git commit -am 'feat(nain): add files lab-4'
git push
[master 033476c] feat(nain): add files lab-4
2 files changed, 32 insertions(+)
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab4.asm
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 3 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 955 байтов | 955.00 КиБ/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:almos05/study_2023-2024_arh-pc.git
   badba90..033476c  master -> master
admosolov@admosolov-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc$
```

Рис. 2.14: Загрузка файлов на Github

3 Выводы

В ходе выполнения лабораторной работы были освоены процедуры компиляции и сборки программ, написанных на ассемблере *NASM*.