

Лабораторная работа №4

Создание и процесс обработки программ на языке ассемблера NASM

Колонтырский Илья Русланович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	10

Список иллюстраций

2.1	Создание файла и папки	6
2.2	Открытие файла	6
2.3	Вставка кода	6
2.4	Компиляция файла	7
2.5	Компиляция файла с аргументами	7
2.6	Сборка файла	7
2.7	Сборка другого файла	7
2.8	Запуск программы	8
2.9	Создание копии	8
2.10	Редактирование файла	8
2.11	Сборка и проверка	8
2.12	Копирование файлов и их загрузка	9

Список таблиц

1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM

2 Выполнение лабораторной работы

Создадим папку для хранения файлов и создадим там файл в формате .asm (рис. 2.1)

```
irkolontyrskiy@irkolontyrskiy:~$ mkdir -p ~/work/arch-pc/lab04
irkolontyrskiy@irkolontyrskiy:~$ cd ~/work/arch-pc/lab04
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab04$ touch hello.asm
```

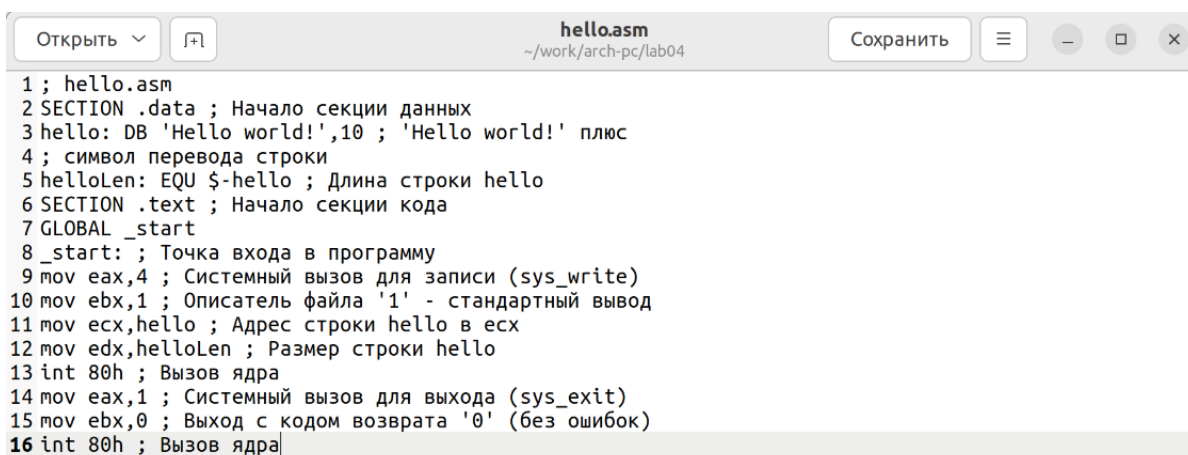
Рис. 2.1: Создание файла и папки

Откроем файл для редактирования (рис. 2.2)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab04$ gedit hello.asm
```

Рис. 2.2: Открытие файла

Вставим в файл требуемый код (рис. 2.3)



```
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
```

Рис. 2.3: Вставка кода

Скомпилируем код в объектный файл с помощью `nasm` и убедимся в том, что он был создан (рис. 2.4)

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

Рис. 2.4: Компиляция файла

Объектный файл имеет такое же имя, как и у файла `.asm`, только с расширением `.obj`. Теперь скомпилируем код с большим количеством аргументов (рис. 2.5)

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

Рис. 2.5: Компиляция файла с аргументами

Соберём исполняемый файл с помощью `ld` и проверим успешность операции (рис. 2.6)

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

Рис. 2.6: Сборка файла

Соберём другой файл и проверим успешность его сборки (рис. 2.7)

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

Рис. 2.7: Сборка другого файла

Здесь исполняемый файл будет иметь имя `main`, и будет собираться из файла `obj.o`. Запустим один из собранных файлов (рис. 2.8)

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

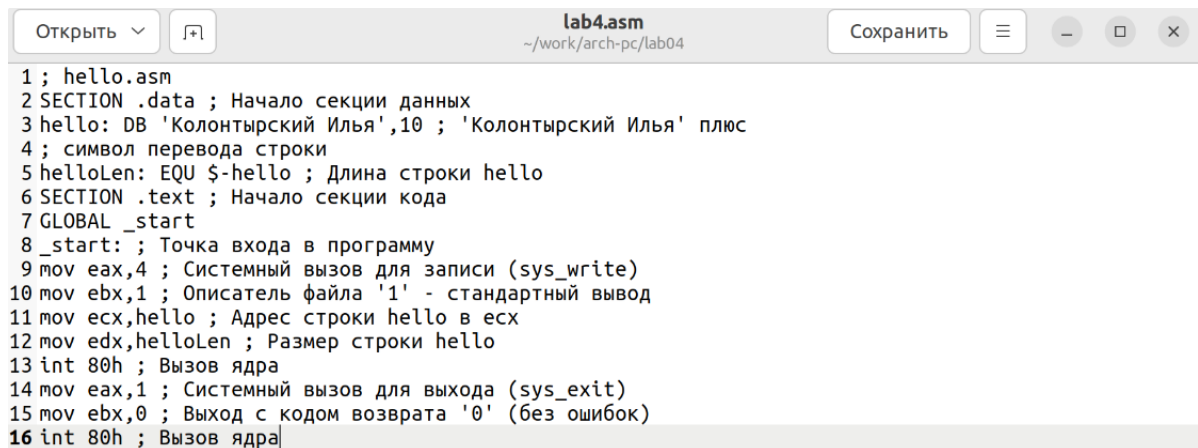
Рис. 2.8: Запуск программы

Создадим копию файла с исходным кодом (рис. 2.9)

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

Рис. 2.9: Создание копии

Изменим код так, чтобы программа выводила фамилию и имя (рис. 2.10)



```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Колонтырский Илья',10 ; 'Колонтырский Илья' 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. 2.10: Редактирование файла

Соберём исполняемый файл и проверим его работу (рис. 2.11)

```
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab04$ ./lab4
Колонтырский Илья
```

Рис. 2.11: Сборка и проверка

Скопируем .asm файлы в рабочую папку и загрузим их на гитхаб (рис. 2.12)


```

irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab04$ cp hello.asm ~/work/study/20
23-2024/Архитектура\ компьютера/arch-pc/labs/lab04/
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab04$ cp lab4.asm ~/work/study/202
3-2024/Архитектура\ компьютера/arch-pc/labs/lab04/
irkolontyrskiy@irkolontyrskiy:~/work/arch-pc/lab04$ cd ~/work/study/2023-2024/Ар
хитектура\ компьютера/arch-pc/
irkolontyrskiy@irkolontyrskiy:~/work/study/2023-2024/Архитектура компьютера/arch
-pc$ git add .
irkolontyrskiy@irkolontyrskiy:~/work/study/2023-2024/Архитектура компьютера/arch
-pc$ git commit -am 'feat(main): files for lab4'
[master bef6e28] feat(main): files for lab4
 2 files changed, 32 insertions(+)
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab4.asm
irkolontyrskiy@irkolontyrskiy:~/work/study/2023-2024/Архитектура компьютера/arch
-pc$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 983 байта | 983.00 КиБ/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно использов
ано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:NaomizChill/study_2023-2024_arh-pc.git
 33e07e5..bef6e28 master -> master

```

Рис. 2.12: Копирование файлов и их загрузка

3 Выводы

Были получены базовые навыки работы с ассемблером и сборкой исполняемых файлов из него