

Отчёт по лабораторной работе №5

Уткина Алина Дмитриевна

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	8
2.4	Компоновщик LD	8
2.5	Запуск исполняемого файла	9
2.6	Задания для самостоятельной работы	9
3	Выводы	12

Список иллюстраций

2.1	Создание файла hello.asm	6
2.2	Текст файла hello.asm	6
2.3	Компиляция программы	7
2.4	Компиляция программы с дополнительными опциями	8
2.5	Процесс создания ассемблерной программы	8
2.6	Передача файла на обработку компоновщику	9
2.7	Применение команды ld	9
2.8	Запуск исполняемого файла	9
2.9	Копирование файла с новым именем	10
2.10	Внесение изменений в объектный файл	10
2.11	Трансляция, компоновка и запуск программы	10
2.12	Копирование файлов и их загрузка на Github	11

Список таблиц

1 Цель работы

Целью данной работы является 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.

Для этого переходим в каталог для работы с программами на языке ассемблера NASM, создаем там текстовый файл с именем hello.asm и открываем его с помощью текстового редактора gedit (рис. 2.1).

```
[adutkina@fedora report]$ cd ~/work/study/2022-2023/Архитектура\ компьютера/arch-  
pc/labs/lab05/  
[adutkina@fedora lab05]$ touch hello.asm  
[adutkina@fedora lab05]$ gedit hello.asm
```

Рис. 2.1: Создание файла hello.asm

Далее в файл вводим следующий текст (рис. 2.2):

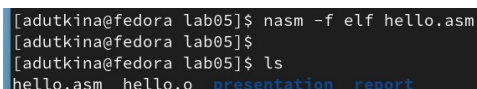
```
1 ; hello.asm  
2 SECTION .data  
3     hello:          DB  'ello world!',10      ; Начало секции данных  
4                                     ; 'Hello world!' плюс  
5     helloLen:       EQU $-hello              ; символ перевода строки  
6                                     ; Длина строки hello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

Рис. 2.2: Текст файла hello.asm

В отличие от многих современных высокоуровневых языков программирования, в ассемблерной программе каждая команда располагается на отдельной строке. Размещение нескольких команд на одной строке недопустимо. Синтаксис ассемблера NASM является чувствительным к регистру, т.е. есть разница между большими и малыми буквами.

2.2 Транслятор NASM

NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World» необходимо написать определенную команду (рис. 2.3):



```
[adutkina@fedora lab05]$ nasm -f elf hello.asm
[adutkina@fedora lab05]$
[adutkina@fedora lab05]$ ls
hello.asm hello.o presentation report
```

Рис. 2.3: Компиляция программы

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. NASM не запускают без параметров. Ключ `-f` указывает транслятору, что требуется создать бинарные файлы в формате ELF. Следует отметить, что формат `elf64` позволяет создавать исполняемый код, работающий под 64-битными версиями Linux. Для 32-битных версий ОС указываем в качестве формата просто `elf`. NASM всегда создаёт выходные файлы в текущем каталоге.

2.3 Расширенный синтаксис командной строки NASM

Полный вариант командной строки `nasm` выглядит следующим образом: `nasm [-@ косвенный_файл_настроек] [-o объектный_файл] [-f формат_объектного_файла] [-l листинг] [параметры...] [-] исходный_файл`

Выполним следующую команду (рис. 2.4):

```
[adutkina@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[adutkina@fedora lab05]$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
```

Рис. 2.4: Компиляция программы с дополнительными опциями

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`).

2.4 Компоновщик LD

Как видно из схемы (рис. 2.5), чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику (рис. 2.6):



Рис. 2.5: Процесс создания ассемблерной программы


```
[adutkina@fedora lab05]$ ld -m elf_i386 hello.o -o hello
[adutkina@fedora lab05]$ ls
hello hello.asm hello.o list.lst obj.o presentation report
```

Рис. 2.6: Передача файла на обработку компоновщику

Компоновщик `ld` не предполагает по умолчанию расширений для файлов, но принято использовать следующие расширения: `*.o` – для объектных файлов; `*` без расширения – для исполняемых файлов; `*.tar` – для файлов схемы программы; `*.lib` – для библиотек

Ключ `-o` с последующим значением задаёт в данном случае имя создаваемого исполняемого файла. Выполним следующую команду (рис. 2.7)

```
[adutkina@fedora lab05]$ ld -m elf_i386 obj.o -o main
[adutkina@fedora lab05]$ ls
hello hello.asm hello.o list.lst main obj.o presentation report
```

Рис. 2.7: Применение команды `ld`

Здесь исполняемый файл с именем `main`, а объектный файл, из которого собран исполняемый файл – `obj.o`

2.5 Запуск исполняемого файла

Запустить на выполнение созданный исполняемый файл, находящийся в текущем каталоге, можно, набрав в командной строке `./hello` (рис. 2.8)

```
[adutkina@fedora lab05]$ ./hello
Hello world!
```

Рис. 2.8: Запуск исполняемого файла

2.6 Задания для самостоятельной работы

1. В каталоге `~/.../lab05` с помощью команды `cp` создадим копию файла `hello.asm` с именем `lab5.asm` (рис. 2.9)

```
[adutkina@fedora lab05]$ cp hello.asm lab5.asm
[adutkina@fedora lab05]$ ls
hello    hello.o    list.lst  obj.o      report
hello.asm lab5.asm  main      presentation
```

Рис. 2.9: Копирование файла с новым именем

2. С помощью любого текстового редактора внесем изменения в текст программы в файле lab5.asm так, чтобы вместо Hello world! на экран выводилась строка с фамилией и именем (рис. 2.10).

```
1 ; lab05.asm
2 SECTION .data                ; Начало секции данных
3     name:                    DB 'Utkina Alina',10      ;
4                               ; символ перевода строки
5     nameLen:                 EQU $-name                ; Длина строки name
6
7 SECTION .text                ; Начало секции кода
8     GLOBAL _start
9
10    _start:                  ; Точка входа в программу
11        mov eax,4            ; Системный вызов для записи (sys_write)
12        mov ebx,1            ; Описатель файла '1' - стандартный вывод
13        mov ecx,name         ; Адрес строки name в ecx
14        mov edx,nameLen      ; Размер строки name
15        int 80h              ; Вызов ядра
16
17        mov eax,1            ; Системный вызов для выхода (sys_exit)
18        mov ebx,0            ; Выход с кодом возврата '0' (без ошибок)
19        int 80h              ; Вызов ядра
```

Рис. 2.10: Внесение изменений в объектный файл

3. Оттранслируем полученный текст программы lab5.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл (рис. 2.11).

```
10 [adutkina@fedora lab05]$ nasm -f elf lab5.asm
11
12 [adutkina@fedora lab05]$ ls
13 hello    hello.o    lab5.o    main      presentation
14 hello.asm lab5.asm  list.lst  obj.o      report
15
16 [adutkina@fedora lab05]$ ld -m elf_i386 lab5.o -o lab5
17
18 [adutkina@fedora lab05]$ ls
19 hello    hello.o    lab5.asm  list.lst  obj.o      report
20 hello.asm lab5     lab5.o    main      presentation
21
22 [adutkina@fedora lab05]$ ./lab5
23 Utkina Alina
```

Рис. 2.11: Трансляция, компоновка и запуск программы

4. Скопируем файлы hello.asm и lab5.asm в локальный репозиторий в каталог ~/work/study/2022-2023/“Архитектура компьютера”/archpc/labs/lab05/. Загрузим файлы на Github (рис. 2.12).

```

[adutkina@fedora lab05]$ cp hello.asm lab5.asm ~/work/study/2022-2023/Архитек
тура\ компьютера/arch-pc/labs/lab05/
[adutkina@fedora lab05]$ cd ~/work/study/2022-2023/Архитектура\ компьютера/ar
ch-pc/labs/lab05/
[adutkina@fedora lab05]$ git add .
[adutkina@fedora lab05]$ git commit -am 'feat(main): add files lab-5'
[master 436896c] feat(main): add files lab-5
4 files changed, 200 insertions(+), 119 deletions(-)
delete mode 100644 labs/lab03/report/report.docx
create mode 100644 labs/lab05/hello.asm
create mode 100644 labs/lab05/lab5.asm
rewrite labs/lab05/report/report.md (71%)
[adutkina@fedora lab05]$ git push
Перечисление объектов: 17, готово.
Пересчет объектов: 100% (17/17) - готово

```

Рис. 2.12: Копирование файлов и их загрузка на Github

3 Выводы

В ходе лабораторной работы были изучены теоретические сведения о языке ассемблера и освоены процедуры компиляции и сборки программ, написанных на ассемблере NASM.