

Отчет по лабораторной работе №4

Язык ассемблера NASM

Лиджиева В.Д.

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	12
	Список литературы	13

Список иллюстраций

4.1	Создание каталога	8
4.2	Переход в каталог	8
4.3	Создание текстового файла и открытие файла	8
4.4	Ввод текста	9
4.5	Компиляция текста и проверка, что объектный файл был создан .	9
4.6	Создание файлов и проверка, что файлы были созданы.	9
4.7	Передача файла на компоновку и проверка, что исполняемый файл hello был создан	9
4.8	Зададим имя создаваемого исполняемого файла и запуск на выпол- нение	10
4.9	Создание копии файла с именем lab4.asm	10
4.10	Внесение изменения в текст программы	10
4.11	Оттранслирование, компоновка, запуск	11
4.12	Копирование файлов в локальный репозиторий	11
4.13	Загрузка файлов на гитхаб	11

Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . . .	7
-----	---	---

1 Цель работы

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

2 Задание

Программа `HelloWorld!` создать каталог для работы с программами на языке ассемблера `NASM`

перейти в созданный каталог

создать текстовый файл с именем `hello.asm`

открыть этот файл

ввести в него указанный текст

Транслятор `Nasm`

выполнить компиляцию в объектный код

Расширенный синтаксис

выполнить компиляцию исходного файла

Компановщик `LD`

передать объектный файл на обработку компановщику

Запустить исполняемый файл

Задания для самостоятельной работы

создать копию файла `hello.asm` с именем `lab4.asm`

изменить скопированный файл, чтобы выводилась строка с именем и фамилией

3 Теоретическое введение

Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux	
Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно про Unix см. в [1–4].

4 Выполнение лабораторной работы

1. Создайте каталог для работы с программами на языке ассемблера NASM:

```
vdlidzhieva@dk8n60 ~ $ mkdir -p ~/work/arch-pc/lab04
```

Рис. 4.1: Создание каталога

2. Перейдём в созданный каталог:

```
vdlidzhieva@dk8n60 ~ $ cd ~/work/arch-pc/lab04
```

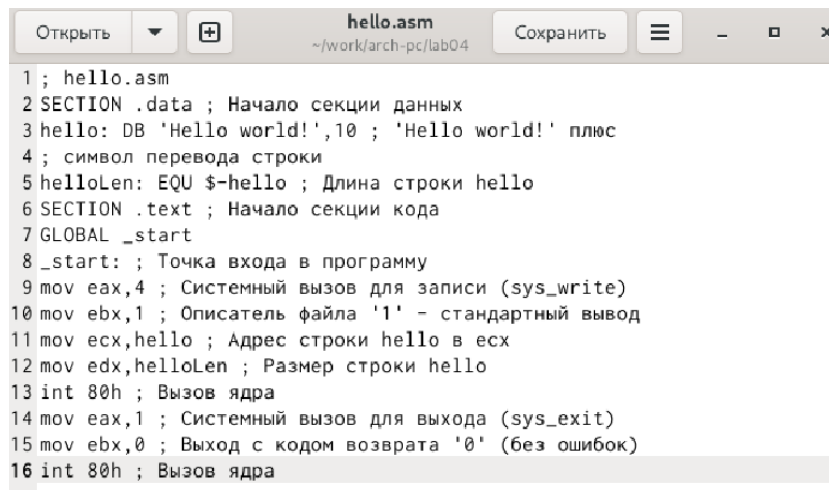
Рис. 4.2: Переход в каталог

3. Создадим текстовый файл с именем hello.asm и откроем этот файл с помощью текстового редактора:

```
vdlidzhieva@dk8n60 ~/work/arch-pc/lab04 $ touch hello.asm  
vdlidzhieva@dk8n60 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 4.3: Создание текстового файла и открытие файла

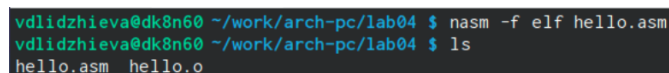
4. Введём в него текст:



```
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. 4.4: Ввод текста

5. Скомпилируем данный текст и проверим, что объектный файл был создан:

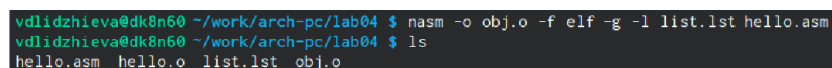


```
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
```

Рис. 4.5: Компиляция текста и проверка, что объектный файл был создан

6. Скомпилируем исходный файл hello.asm в obj.o и создадим файл листинга

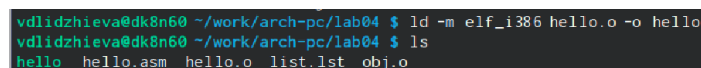
list.lst и проверим, что файлы были созданы.



```
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.6: Создание файлов и проверка, что файлы были созданы.

7. Передадим объектный файл на обработку компоновщику и проверим, что исполняемый файл hello был создан.



```
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.7: Передача файла на компоновку и проверка, что исполняемый файл hello был создан

8. Зададим имя создаваемого исполняемого файла, запустим на выполнение созданный исполняемый файл, находящийся в

текущем каталоге.

```
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ ./hello
Hello world!
```

Рис. 4.8: Зададим имя создаваемого исполняемого файла и запуск на выполнение

9. Создадим копию файла hello.asm с именем lab4.asm

```
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 4.9: Создание копии файла с именем lab4.asm

10. Внесём изменения в текст программы в файле lab4.asm

```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Lidzhieva Valeria',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. 4.10: Внесение изменения в текст программы

11. Оттранслируем полученный текст программы lab4.asm в объектный файл.

Выполним компоновку объектного файла и запустим получившийся исполняемый файл.

```

vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst lab4.asm
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ nasm -o lidzhieva.o -f elf -g -l list.lst lab4.asm
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ nasm -o lidzhieva.o -f elf -g -l list.lst lab4.asm
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 lidzhieva.o o lidzhieva
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ ./lidzhieva
lidzhieva Valerie

```

Рис. 4.11: Оттранслирование, компоновка, запуск

12. Скопировала файлы hello.asm и lab4.asm в локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/ спомощью утилиты cp и проверил наличие файлов с помощью утилиты ls

```

vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ cp hello.asm ~/work/arch-pc/lab04/report
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ cp lab4.asm ~/work/arch-pc/lab04/report
vdldzhieva@dk8n60 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  lidzhieva  lidzhieva.o  list.lst  main  obj.o  report

```

Рис. 4.12: Копирование файлов в локальный репозиторий

13. Загружаю файлы на Github

```

vdldzhieva@dk8n60 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/report $ git add .
vdldzhieva@dk8n60 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/report $ git commit -am
'feat(main):add files to lab 04'
[master 13d00b1] feat(main):add files to lab 04
2 files changed, 32 insertions(+)
create mode 100644 labs/lab04/report/hello.asm
create mode 100644 labs/lab04/report/lab4.asm
vdldzhieva@dk8n60 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/report $ git push
Перечисление объектов: 10, готово.
Подсчет объектов: 100% (10/10), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.04 КиБ | 1.04 МБ/с, готово.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:lidzhievav/tudy_2024-2025_arh-pc.git
   0cf75e6..13d00b1  master -> master

```

Рис. 4.13: Загрузка файлов на гитхаб

5 Выводы

В ходе выполнения работы, я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.
2. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.