

Лабораторная работа №4

Архитектура вычислительных систем

Заболотная Кристина Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	14
	Список литературы	15

Список иллюстраций

4.1	41.png	9
4.2	42.png	9
4.3	44.png	10
4.4	45.png	11
4.5	43.png	12
4.6	46.png	12
4.7	47.png	12
4.8	48.png	13

Список таблиц

1 Цель работы

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

2 Задание

1. В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создадим копию файла `hello.asm` с именем `lab4.asm`.
2. С помощью любого текстового редактора внесим изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с моим фамилией и именем.
3. Оттранслируем полученный текст программы `lab5.asm` в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл.
4. Скопируем файлы `hello.asm` и `lab5.asm` в свой локальный репозиторий в каталог `~/work/study/2022-2023/“Архитектура компьютера”/archpc/labs/lab05/`. Загрузим файлы на Github.

3 Теоретическое введение

Язык ассемблера (assembly language, сокращённо asm) — машинноориентированный 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. Мнемокод — непосредственно мнемоника инструкции процессору, которая является обязательной частью команды. Операндами могут быть числа, данные, адреса регистров или адреса оперативной памяти. Метка — это идентификатор, с которым ассемблер ассоциирует некоторое число, чаще всего адрес в памяти. Архитектура ЭВМ. В процессе создания ассемблерной программы можно выделить четыре шага: 1. Набор текста программы в текстовом редакторе и сохранение её в отдельном файле. Каждый файл имеет свой тип (или расширение), который определяет назначение файла. Файлы с исходным текстом программ на языке ассемблера имеют тип asm. 2. Трансляция — преобразование с помощью транслятора, например nasm, текста программы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную дополнительную информацию, созданную транслятором. Тип объектного файла — o, файла листинга — lst. 3. Компоновка или линковка — этап обработки объектного кода компоновщиком (ld), который принимает на вход объектные файлы и собирает по ним исполняемый файл. Исполняемый файл обычно не имеет расширения. Кроме того, можно получить файл карты загрузки программы в ОЗУ, имеющий расширение map. 4. Запуск программы. Конечной целью является

работоспособный исполняемый файл. Ошибки на предыдущих этапах могут привести к некорректной работе программы, поэтому может присутствовать этап отладки программы при помощи специальной программы — отладчика. При нахождении ошибки необходимо провести коррекцию программы, начиная с первого шага.

4 Выполнение лабораторной работы

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 4.1)

1. Создадим каталог для работы с программами на языке ассемблера NASM, перейдем в созданный каталог. Ссылка на иллюстрацию (рис. 4.1)

```
kazabolotnaya@dk2n26 ~ $ mkdir ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab04
```

Рис. 4.1: 41.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 4.2)

2. Создадим текстовый файл с именем hello.asm. Ссылка на иллюстрацию (рис. 4.2)

```
kazabolotnaya@dk2n26 ~ $ cd ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab04
kazabolotnaya@dk2n26 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04 $ nasm -f elf hello.asm
```

Рис. 4.2: 42.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 4.3)

3. Откроем этот файл с помощью текстового редактора gedit. Ссылка на иллюстрацию (рис. 4.3)

```

; hello.asm
SECTION .data                                ; Начало секции данных
    hello:    DB 'Hello world!',10 ; 'Hello world!' плюс
                                           ; символ перевода строки
    helloLen: EQU $-hello                ; Длина строки hello

SECTION .text                                ; Начало секции кода
    GLOBAL _start

_start:                                       ; Точка входа в программу
    mov eax,4                               ; Системный вызов для записи (sys_write)
    mov ebx,1                               ; Описатель файла '1' - стандартный вывод
    mov ecx,hello                           ; Адрес строки hello в ecx
    mov edx,helloLen                       ; Размер строки hello
    int 80h                                ; Вызов ядра

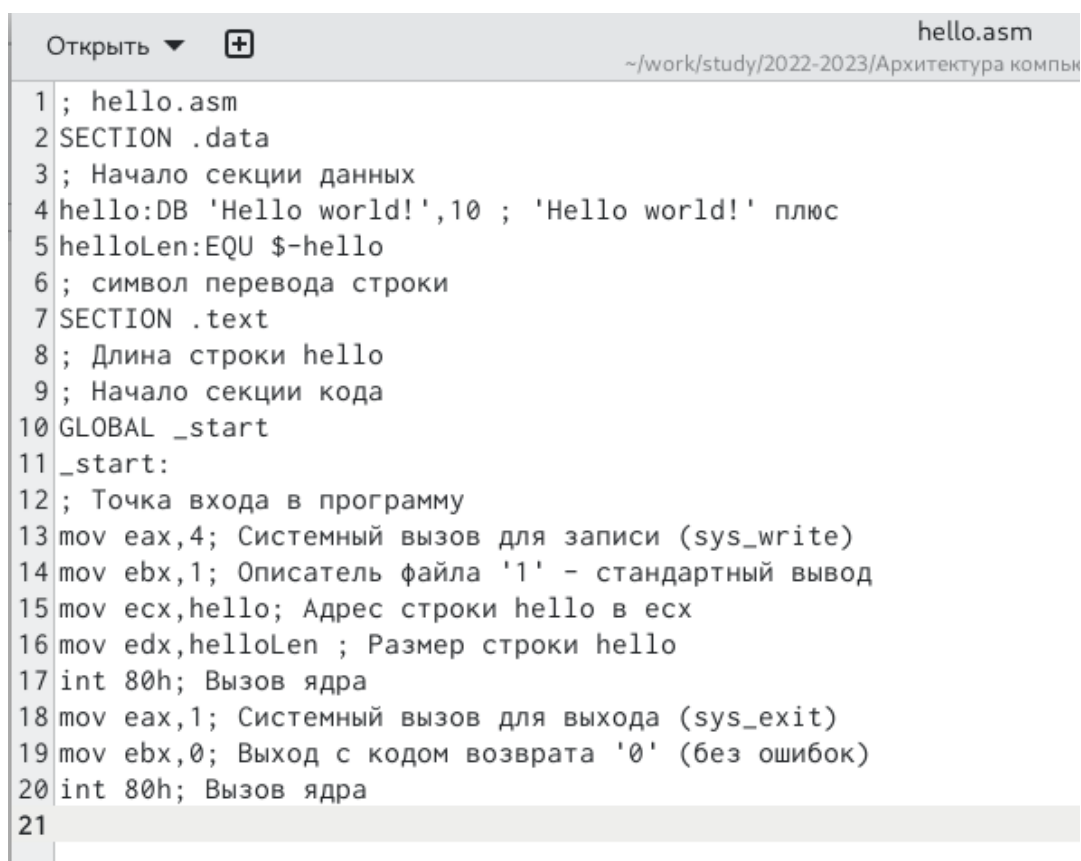
    mov eax,1                               ; Системный вызов для выхода (sys_exit)
    mov ebx,0                               ; Выход с кодом возврата '0' (без ошибок)
    int 80h                                ; Вызов ядра

```

Рис. 4.3: 44.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 4.4)

4. Введём в него следующий текст. Ссылка на иллюстрацию (рис. 4.4)



The screenshot shows a text editor window titled 'hello.asm' with a menu bar containing 'Открыть' (Open) and a plus icon. The address bar shows the path '~/.work/study/2022-2023/Архитектура компь'. The code is as follows:

```
1 ; hello.asm
2 SECTION .data
3 ; Начало секции данных
4 hello:DB 'Hello world!',10 ; 'Hello world!' плюс
5 helloLen:EQU $-hello
6 ; символ перевода строки
7 SECTION .text
8 ; Длина строки hello
9 ; Начало секции кода
10 GLOBAL _start
11 _start:
12 ; Точка входа в программу
13 mov eax,4; Системный вызов для записи (sys_write)
14 mov ebx,1; Описатель файла '1' - стандартный вывод
15 mov ecx,hello; Адрес строки hello в ecx
16 mov edx,helloLen ; Размер строки hello
17 int 80h; Вызов ядра
18 mov eax,1; Системный вызов для выхода (sys_exit)
19 mov ebx,0; Выход с кодом возврата '0' (без ошибок)
20 int 80h; Вызов ядра
21
```

Рис. 4.4: 45.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 4.5)

5. Выполним команду, которая скомпилирует исходный файл hello.asm в obj.o, при этом формат выходного файла будет elf, и в него будут включены символы для отладки, кроме того, будет создан файл листинга list.lst. С помощью команды ls проверим, что файлы были созданы. Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику. С помощью команды ls проверим, что исполняемый файл hello был создан. Ссылка на иллюстрацию (рис. 4.5)

```

kazabolotnaya@dk2n26 ~ $ mkdir ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab04
mkdir: невозможно создать каталог «/afs/.dk.sci.pfu.edu.ru/home/k/a/kazabolotnaya/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04»: Файл существует
kazabolotnaya@dk2n26 ~ $ cd ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab04

kazabolotnaya@dk2n26 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04 $ nasm -f elf hello.asm
kazabolotnaya@dk2n26 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04 $ gedit hello.asm
kazabolotnaya@dk2n26 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
kazabolotnaya@dk2n26 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello hello.asm hello.o list.lst main obj.o presentation report
kazabolotnaya@dk2n26 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04 $ ld -m elf_i386 hello.o -o hello
kazabolotnaya@dk2n26 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello hello.asm hello.o list.lst main obj.o presentation report
kazabolotnaya@dk2n26 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04 $ ld -m elf_i386 obj.o -o main
kazabolotnaya@dk2n26 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04 $ ld --help
Использование ld [параметры] файл...
Параметры:
  -a КЛЮЧЕВОЕ СЛОВО                Управление общей библиотекой для совместимости с HP/UX
  -A АРХИТЕКТУРА, --architecture АРХИТЕКТУРА  Задать архитектуру
  -b ЦЕЛЬ, --format ЦЕЛЬ

```

Рис. 4.5: 43.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 4.6)

6. Запустим исполняемый файл набрав в командной строке: ./hello. Ссылка на иллюстрацию (рис. 4.6)

```

kazabolotnaya@dk2n26 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04 $ ./hello
Hello world!

```

Рис. 4.6: 46.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 4.7)

7. В каталоге ~/work/arch-pc/lab04 с помощью команды cp создадим копию файла hello.asm с именем lab4.asm. Ссылка на иллюстрацию (рис. 4.7)

```

kazabolotnaya@dk8n64 ~ $ cd ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab04
kazabolotnaya@dk8n64 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04 $ cp hello.asm lab4.asm

```

Рис. 4.7: 47.png

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 4.8)

8. С помощью любого текстового редактора внесем изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с фамилией и именем. Оттранслируем полученный текст программы lab4.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл. Ссылка на иллюстрацию (рис. 4.8)

```
kazabolotnaya@dk8n64 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/
lab04 $ gedit lab4.asm
kazabolotnaya@dk8n64 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/
lab04 $ nasm -f elf -g -l list.lst lab4.asm
kazabolotnaya@dk8n64 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/
lab04 $ ld -m elf_i386 lab4.o -o lab4
kazabolotnaya@dk8n64 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/
lab04 $ ./lab4
Заболотная Кристина
kazabolotnaya@dk8n64 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/
lab04 $ █
```

Рис. 4.8: 48.png

5 Выводы

В ходе выполнения данной лабораторной работы была освоена процедура компиляции и сборки программ, написанных на ассемблере NASM.

Список литературы