

Отчёт по лабораторной работе №4

Дисциплина: Архитектура компьютеров

Мухина Ксения Николаевна

Содержание

| | | |
|----------|---|-----------|
| 1 | Цель работы | 5 |
| 2 | Выполнение лабораторной работы | 6 |
| 3 | Задания для самостоятельной работы | 10 |
| 4 | Выводы | 12 |
| | Список литературы | 13 |

Список иллюстраций

| | | |
|-----|--|----|
| 2.1 | Рис. 1. Создание каталога и файла для будущей программы. . . . | 6 |
| 2.2 | Рис. 2. Код hello. | 7 |
| 2.3 | Рис. 3. Компиляция текста Hello world! | 7 |
| 2.4 | Рис. 4. Компиляция текста в файл obj.o и создание листинга list.lst. . . . | 8 |
| 2.5 | Рис. 5. Создание исполняемого файла hello. | 8 |
| 2.6 | Рис. 6. Создание исполняемого файла с заданным именем. | 9 |
| 2.7 | Рис. 7. Запуск исполняемого файла hello. | 9 |
| 3.1 | Рис. 8. Код lab4. | 10 |
| 3.2 | Рис. 9. Запуск программы lab4. | 11 |
| 3.3 | Рис. 10. Загрузка файлов на GitHub. | 11 |

Список таблиц

1 Цель работы

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

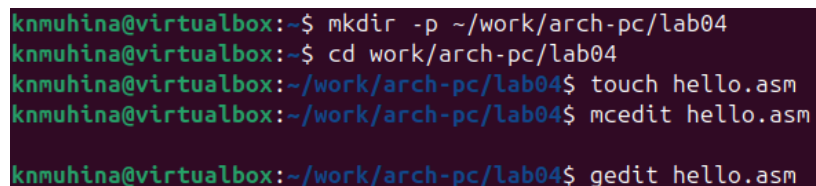
2 Выполнение лабораторной работы

Далее описываемая работа была выполнена на виртуальной машине Oracle VirtualBox с ОС Linux Ubuntu.

1. Программа Hello world!

Рассмотрим пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит на экран сообщение „Hello world!“.

Для этого создадим каталог для работы с будущей программой и перейдём в него, создадим текстовый файл „hello.asm“ и отредактируем его при помощи gedit.

A screenshot of a terminal window with a dark background and light-colored text. It shows a series of commands being executed in a shell. The prompt is 'knmuhina@virtualbox:~\$'. The commands are: 'mkdir -p ~/work/arch-pc/lab04', 'cd work/arch-pc/lab04', 'touch hello.asm', 'mcedit hello.asm', and 'gedit hello.asm'. The output of the first command is shown as 'knmuhina@virtualbox:~\$ mkdir -p ~/work/arch-pc/lab04' followed by a new line.

```
knmuhina@virtualbox:~$ mkdir -p ~/work/arch-pc/lab04
knmuhina@virtualbox:~$ cd work/arch-pc/lab04
knmuhina@virtualbox:~/work/arch-pc/lab04$ touch hello.asm
knmuhina@virtualbox:~/work/arch-pc/lab04$ mcedit hello.asm
knmuhina@virtualbox:~/work/arch-pc/lab04$ gedit hello.asm
```

Рисунок 2.1: Рис. 1. Создание каталога и файла для будущей программы.

Наш код будет выглядеть так.

```

1; hello.asm
2SECTION .data ; Начало секции данных
3    hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4    ; символ перевода строки
5    helloLen: EQU $-hello ; Длина строки hello
6SECTION .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

```

Рисунок 2.2: Рис. 2. Код hello.

Данный код взят из указаний по выполнению лабораторной работы; комментарии к коду указаны после символа „;“. Обратим внимание, что каждая команда располагается строго на отдельной строке. Также синтаксис ассемблера чувствителен к регистру, т.е. есть разница между большими и малыми буквами.

2. Транслятор NASM

NASM превращает текст программы в объектный код. Для компиляции кода программы используем команду `nasm`. После этого при помощи `ls` убедимся в успешности компиляции.

```

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

```

Рисунок 2.3: Рис. 3. Компиляция текста Hello world!

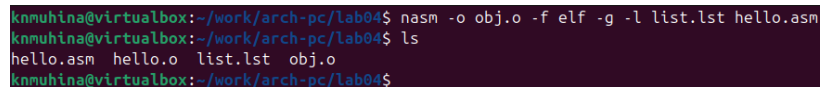
Текст был записан в виде объектного кода в созданный объектный файл под названием „hello.o“. Отметим, что ключ `-f` указывает транслятору, что требуется создать бинарные файлы в формате ELF. NASM всегда создаёт выходные файлы в текущем каталоге: в данном случае это наш каталог lab04.

3. Расширенный синтаксис командной строки NASM

Полный синтаксис команды `nasm` выглядит следующим образом:

`nasm [-@ косвенный_файл_настроек] [-o объектный_файл] [-f формат_объектного_файла] [-l листинг] [параметры...] [-] исходный_файл`

Выполним следующую команду и сразу же проверим её результат при помощи `ls`:



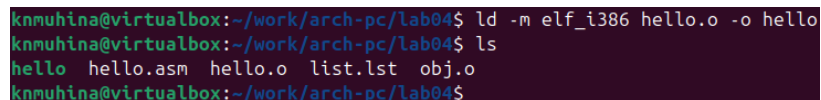
```
knmuhina@virtualbox:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
knmuhina@virtualbox:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
knmuhina@virtualbox:~/work/arch-pc/lab04$
```

Рисунок 2.4: Рис. 4. Компиляция текста в файл `obj.o` и создание листинга `list.lst`.

Данная команда скомпилировала `hello.asm` в `obj.o` (`-o` позволила задать имя объектному файлу), при этом формат выходного файла будет `elf`, в который включены символы для отладки благодаря опции `-g`. Помимо этого был создан файл листинга `list.lst` при помощи опции `-l`.

4. Компоновщик LD

Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику (команда `ld`).



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

Рисунок 2.5: Рис. 5. Создание исполняемого файла `hello`.

Файл был успешно обработан, и в результате была получена программа `hello`.

Компоновщик не предполагает по умолчанию расширений для файлов, но принято использовать следующие:

- `o`: для объектных файлов;

- без расширения: для исполняемых файлов;
- tar: для файлов схемы программы;
- lib: для библиотек.

Так же, как и в команде `nasm`, в команде `ld` при помощи ключа `-o` можно задать имя создаваемому файлу.

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

Рисунок 2.6: Рис. 6. Создание исполняемого файла с заданным именем.

В результате была собрана программа `main` из объектного файла `obj.o`.

4.1. Запуск исполняемого файла

Запустим созданный исполняемый файл `hello`.

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

Рисунок 2.7: Рис. 7. Запуск исполняемого файла `hello`.

Терминал вывел текст „Hello world!“, что показывает успешное исполнение.

3 Задания для самостоятельной работы

Создадим копию файла hello.asm с именем lab4.asm и в соответствии с заданием внесём в него изменения с помощью gedit.

Теперь наш код будет выглядеть так:

```
1 ; lab4.asm
2 SECTION .data ; Начало секции данных
3     lab4: DB 'Мухина Ксения',10
4     lab4Len: EQU $-lab4
5 SECTION .text ; Начало секции кода
6     GLOBAL _start
7 _start: ; Точка входа в программу
8     mov eax,4
9     mov ebx,1
10    mov ecx,lab4
11    mov edx,lab4Len
12    int 80h
13    mov eax,1
14    mov ebx,0
15    int 80h
```

Рисунок 3.1: Рис. 8. Код lab4.

Оттранслируем полученный текст в объектный файл, выполним компоновку файла и запустим получившийся исполняемый файл.

```

kmmuhina@virtualbox:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
kmmuhina@virtualbox:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
kmmuhina@virtualbox:~/work/arch-pc/lab04$ ./lab4
Мухина Ксения
kmmuhina@virtualbox:~/work/arch-pc/lab04$

```

Рисунок 3.2: Рис. 9. Запуск программы lab4.

Загрузим файлы hello.asm и lab4.asm на GitHub.

```

kmmuhina@virtualbox:~/work/arch-pc/lab04$ cp -p hello.asm lab4.asm ~/work/study/2025-2026/
"Архитектура компьютеров"/arch-pc/labs/lab04/
kmmuhina@virtualbox:~/work/arch-pc/lab04$ cd ~/work/study/2025-2026/"Архитектура компьютер
ов"/arch-pc/
kmmuhina@virtualbox:~/work/study/2025-2026/Архитектура компьютеров/arch-pc$ git add .
kmmuhina@virtualbox:~/work/study/2025-2026/Архитектура компьютеров/arch-pc$ git commit -am
'feat(main): add files lab-4'
[master 62c92c9] feat(main): add files lab-4
 2 files changed, 32 insertions(+)
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab4.asm
kmmuhina@virtualbox:~/work/study/2025-2026/Архитектура компьютеров/arch-pc$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.14 КиБ | 582.00 КиБ/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:kmmuhina/study_2025-2026_arch-pc.git
 fc4ad83..62c92c9 master -> master
kmmuhina@virtualbox:~/work/study/2025-2026/Архитектура компьютеров/arch-pc$

```

Рисунок 3.3: Рис. 10. Загрузка файлов на GitHub.

4 Выводы

В ходе выполнения лабораторной работы мы освоили процедуры компиляции и сборки программ, написанных на ассемблере NASM.

Список литературы

1. Файл «Лабораторная работа №4. Создание и процесс обработки программ на языке ассемблера NASM.pdf»