

Отчёта по лабораторной работе № 4

**Создание и процесс обработки программ на языке ассемблера
NASM**

Гущина Екатерина Антоновна

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
3 Задание для самостоятельной работы	10
4 Выводы	13

Список иллюстраций

2.1 Создание файла hello.asm	6
2.2 Редактирование файла hello.asm	7
2.3 Компиляция файла hello.asm	7
2.4 Расширенный синтаксис компиляции файла	8
2.5 Компактная компоновка объектного файла hello.o	8
2.6 Компактная компоновка объектного файла obj.o	8
2.7 Запуск исполняемого файла hello	9
3.1 Создание файла lab4.asm	10
3.2 Редактирование нового файла lab4.asm	11
3.3 Сборка исполняемого файла lab4	11
3.4 Копируем исходники в каталог репозитория	12
3.5 Загрузка на github	12

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

2.1 Программа Hello world!

Создадим каталог для работы с программами на языке ассемблера NASM, в нем создадим текстовый файл с именем hello.asm, откроем его редактором gedit:

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

Рисунок 2.1: Создание файла hello.asm

Введем в него нужный текст:

The screenshot shows a text editor window titled "hello.asm" with the file path "~/work/arch-pc/lab04". The code is written in NASM assembly language. It defines a data section with a string "Hello world!", calculates its length, and then defines a .text section containing the _start label which prints the string to standard output and exits. Russian comments are provided for each line.

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

Рисунок 2.2: Редактирование файла hello.asm

2.2 Транслятор NASM

Скомпилируем программу:

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

Рисунок 2.3: Компиляция файла hello.asm

Был создан объектный файл hello.o.

2.3 Расширенный синтаксис командной строки NASM

Скомпилируем 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).

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

Рисунок 2.4: Расширенный синтаксис компиляции файла

Был создан объектный файл obj.o и файл листинга list.lst.

2.4 Компоновщик LD

Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику:

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

Рисунок 2.5: Компоновка объектного файла hello.o

Исполняемый файл hello был создан.

Выполним следующую команду компоновки:

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

Рисунок 2.6: Компоновка объектного файла obj.o

Был создан исполняемый файл main из объектного файла obj.o.

2.5 Запуск исполняемого файла

Запустим на выполнение созданный исполняемый файл:

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

Рисунок 2.7: Запуск исполняемого файла hello

Программа вывела на экран строку «Hello world!»

3 Задание для самостоятельной работы

1. В каталоге ~/work/arch-pc/lab04 с помощью команды cp создадим копию файла hello.asm с именем lab4.asm:

```
katya@desktop:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
katya@desktop:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm list.lst main obj.o
katya@desktop:~/work/arch-pc/lab04$
```

Рисунок 3.1: Создание файла lab4.asm

2. С помощью текстового редактора gedit внесем изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с фамилией и именем:

The screenshot shows a text editor window titled "lab4.asm" with the file path "~/work/arch-pc/lab04". The code is written in NASM assembly language. It defines a string "Гущина Екатерина" and its length, then prints it to standard output and exits.

```
1 ; lab4.asm
2 section .data
3     name:      db 'Гущина Екатерина',10          ; Начало секции данных
4                                         ; фамилия и имя плюс
5                                         ; символ перевода строки
6                                         ; Длина строки name
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
```

Рисунок 3.2: Редактирование нового файла lab4.asm

3. Оттранслируем полученный текст программы lab4.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл:

```
katya@desktop:~/work/arch-pc/lab04$ gedit lab4.asm
katya@desktop:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
katya@desktop:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
katya@desktop:~/work/arch-pc/lab04$ ./lab4
Гущина Екатерина
katya@desktop:~/work/arch-pc/lab04$
```

Рисунок 3.3: Сборка исполняемого файла lab4

Получен исполняемый файл lab4, который выводит фамилию и имя.

4. Скопируем файлы hello.asm и lab4.asm в локальный репозиторий в каталог ~/work/study/2025-2026/«Архитектура компьютера»/arch-pc/labs/lab04:

```
katya@desktop:~/work/arch-pc/lab04$ cp *.asm ~/work/study/2025-2026/Архитектура\ компьютера/arch-pc/labs/lab04
katya@desktop:~/work/arch-pc/lab04$ cd ~/work/study/2025-2026/Архитектура\ компьютера/arch-pc/labs/lab04/
katya@desktop:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm  lab4.asm  presentation report
```

Рисунок 3.4: Копируем исходники в каталог репозитория

Загрузим эти файлы на github:

```
katya@desktop:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab04$ git add hello.asm lab4.asm
katya@desktop:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab04$ git commit -am "feat(main): add lab-4 source code files"
[master 41c16aa] feat(main): add lab-4 source code files
 4 files changed, 99 insertions(+), 35 deletions(-)
  create mode 100644 labs/lab04/hello.asm
  create mode 100644 labs/lab04/lab4.asm
  delete mode 100644 labs/lab04/report/image/solvay.jpg
katya@desktop:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab04$ git push
Перечисление объектов: 13, готово.
Подсчет объектов: 100% (13/13), готово.
При сжатии изменений используется до 12 потоков
Сжатие объектов: 100% (8/8), готово.
Запись объектов: 100% (8/8), 2.76 КиБ | 2.76 МиБ/с, готово.
Всего 8 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 3 local objects.
To github.com:eaguthina/study_2025-2026_arch-pc.git
 2815366..41c16aa master -> master
katya@desktop:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab04$ 
```

Рисунок 3.5: Загрузка на github

4 Выводы

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