

Отчёт по лабораторной работе 4

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

Власов Артем Сергеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Программа Hello world!	7
3.2	Транслятор NASM	8
3.3	Расширенный синтаксис командной строки NASM	8
3.4	Компоновщик LD	9
3.5	Запуск исполняемого файла	10
3.6	Задание для самостоятельной работы	10
4	Выводы	12

Список иллюстраций

3.1	Создаем каталоги с помощью команды <code>mkdir</code>	7
3.2	Переходим в каталог с помощью команды <code>cd</code>	7
3.3	Создаем текстовый файл <code>hello.asm</code>	7
3.4	Открываем файл и заполняем его по примеру	8
3.5	Используем команду <code>nasm</code>	8
3.6	Проверяем работу команды	8
3.7	Преобразуем файл <code>hello.asm</code> в <code>obj.o</code>	9
3.8	Проверяем создание файла командой <code>ls</code>	9
3.9	Используем команду <code>ld</code>	9
3.10	Используем команду <code>ls</code>	9
3.11	Используем команду <code>ld</code> , создавая файл <code>main</code>	9
3.12	Используем команду <code>ls</code>	9
3.13	Используем команду <code>./hello</code>	10
3.14	Используем команду <code>sr</code>	10
3.15	Открываем файл в текстовом редакторе	10
3.16	Редактируем файл для своего имени и фамилии	10
3.17	Прописываем команды для работы файла и запускаем программу	11
3.18	Копируем файлы в каталог с ЛР4	11
3.19	Загружаем файлы	11

Список таблиц

1 Цель работы

Освоить процедуры компиляции и сборки программ, познакомиться с языком ассемблера NASM.

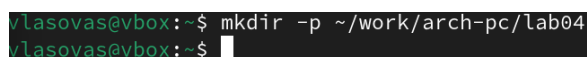
2 Задание

Написать 2 программы(Hello world, lab4(Имя Фамилия))

3 Выполнение лабораторной работы

3.1 Программа Hello world!

Создаем каталог для работы с программами на языке ассемблера NASM (рис. fig. 3.1).



```
vlasovas@vbox:~$ mkdir -p ~/work/arch-pc/lab04
vlasovas@vbox:~$
```

Рис. 3.1: Создаем каталоги с помощью команды mkdir

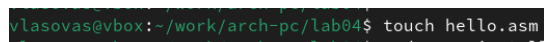
Переходим в созданный каталог (рис. fig. 3.2).



```
vlasovas@vbox:~$ cd ~/work/arch-pc/lab04
vlasovas@vbox:~/work/arch-pc/lab04$
```

Рис. 3.2: Переходим в каталог с помощью команды cd

Создаем текстовый файл (рис. fig. 3.3).



```
vlasovas@vbox:~/work/arch-pc/lab04$ touch hello.asm
```

Рис. 3.3: Создаем текстовый файл hello.asm

Открываем данный файл в текстовом редакторе (рис. fig. 3.4).

A screenshot of a text editor window titled 'hello.asm' with a path '~/.work/arch-pc/lab04'. The code is written in NASM assembly and includes comments in Russian. It defines a data section with a string 'Hello world!', calculates its length, and defines a text section with the main program logic. The program uses 'sys_write' to output the string and 'sys_exit' to terminate. It uses the 'int 80h' instruction for system calls.

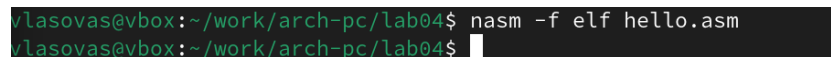
```
; hello.asm
SECTION .data      | ; Начало секции данных
    hello: DB 'Hello world!',10 ; 'Hello world!' плюс
              ; символ перевода строки
    helloLen: EQU $-hello ; Длина строки hello
SECTION .text      ; Начало секции кода
    GLOBAL _start
_start:            ; Точка входа в программу
    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. 3.4: Открываем файл и заполняем его по примеру

3.2 Транслятор NASM

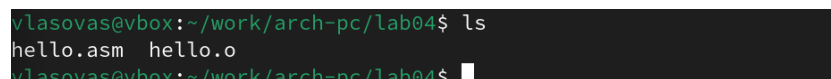
Преобразуем текст программы в объектный код (рис. fig. 3.5).

A terminal window showing the command 'nasm -f elf hello.asm' being executed successfully in a directory '~/.work/arch-pc/lab04'.

```
vasovas@vbox:~/work/arch-pc/lab04$ nasm -f elf hello.asm
vasovas@vbox:~/work/arch-pc/lab04$
```

Рис. 3.5: Используем команду nasm

Проверяем созданся ли объектный файл с помощью команды ls (рис. fig. 3.6).

A terminal window showing the command 'ls' being executed, which lists the files 'hello.asm' and 'hello.o' in the current directory '~/.work/arch-pc/lab04'.

```
vasovas@vbox:~/work/arch-pc/lab04$ ls
hello.asm hello.o
vasovas@vbox:~/work/arch-pc/lab04$
```

Рис. 3.6: Проверяем работу команды

3.3 Расширенный синтаксис командной строки NASM

Компилируем исходный файл (рис. fig. 3.7).


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

Рис. 3.7: Преобразуем файл hello.asm в obj.o

Проверяем, как сработала команда (рис. fig. 3.8).

```
vlasovas@vbox:~/work/arch-pc/lab04$ ls  
hello.asm hello.o list.lst obj.o  
vlasovas@vbox:~/work/arch-pc/lab04$
```

Рис. 3.8: Проверяем создание файла командой ls

3.4 Компоновщик LD

Передаем объектный файл на обработку компоновщику (рис. fig. 3.9).

```
vlasovas@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello  
vlasovas@vbox:~/work/arch-pc/lab04$
```

Рис. 3.9: Используем команду ld

Проверяем создался ли исполняемый файл hello (рис. fig. 3.10).

```
vlasovas@vbox:~/work/arch-pc/lab04$ ls  
hello hello.asm hello.o list.lst obj.o  
vlasovas@vbox:~/work/arch-pc/lab04$
```

Рис. 3.10: Используем команду ls

Передаем объектный файл на обработку компоновщику (рис. fig. 3.11).

```
vlasovas@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main  
vlasovas@vbox:~/work/arch-pc/lab04$
```

Рис. 3.11: Используем команду ld, создавая файл main

Проверяем создался ли исполняемый файл hello (рис. fig. 3.12).

```
vlasovas@vbox:~/work/arch-pc/lab04$ ls  
hello hello.asm hello.o list.lst main obj.o  
vlasovas@vbox:~/work/arch-pc/lab04$
```

Рис. 3.12: Используем команду ls

3.5 Запуск исполняемого файла

Запускаем на выполнение созданный исполняемый файл (рис. fig. 3.13).

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

Рис. 3.13: Используем команду ./hello

3.6 Задание для самостоятельной работы

Создаем копию файла hello.asm (рис. fig. 3.14).

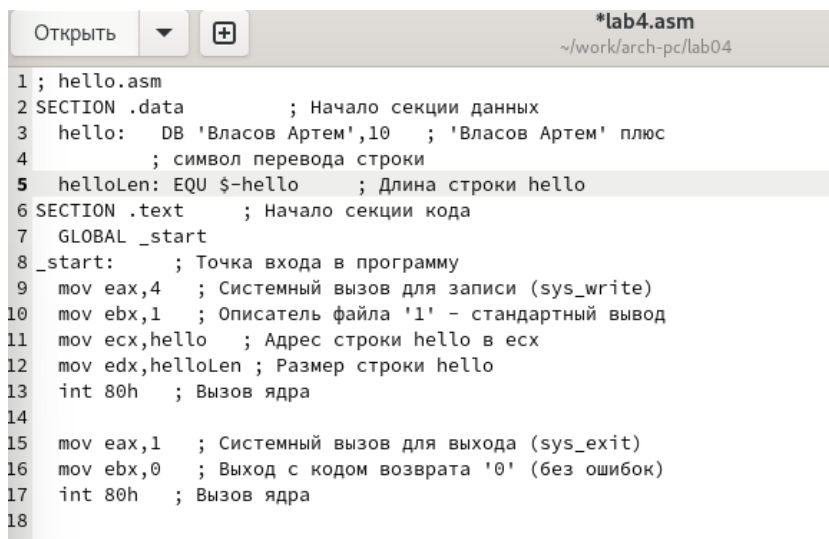
```
vlasovas@vbox:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
vlasovas@vbox:~/work/arch-pc/lab04$
```

Рис. 3.14: Используем команду cp

Открываем файл и редактируем его (рис. fig. 3.15).

```
vlasovas@vbox:~/work/arch-pc/lab04$ gedit lab4.asm
```

Рис. 3.15: Открываем файл в текстовом редакторе



```
Открыть ▼ + *lab4.asm
~/work/arch-pc/lab04

1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Власов Артем',10 ; 'Власов Артем' 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
15 mov eax,1 ; Системный вызов для выхода (sys_exit)
16 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
17 int 80h ; Вызов ядра
18
```

Рис. 3.16: Редактируем файл для своего имени и фамилии

Прописываем те же команды, что и с первой программой (рис. fig. 3.17).

```

vlasovas@vbox:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
vlasovas@vbox:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst lab4.asm
vlasovas@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o hello
vlasovas@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
vlasovas@vbox:~/work/arch-pc/lab04$ ./hello
Власов Артем
vlasovas@vbox:~/work/arch-pc/lab04$

```

Рис. 3.17: Прописываем команды для работы файла и запускаем программу

Копируем файлы в локальный репозиторий (рис. fig. 3.18).

```

vlasovas@vbox:~/work/arch-pc/lab04$ cp hello.asm ~/work/study/2024-2025/Архитектура\
компьютера/arch-pc/labs/lab04/
vlasovas@vbox:~/work/arch-pc/lab04$ cp lab4.asm ~/work/study/2024-2025/Архитектура\ к
мпьютера/arch-pc/labs/lab04/
vlasovas@vbox:~/work/arch-pc/lab04$

```

Рис. 3.18: Копируем файлы в каталог с ЛР4

Переходим в каталог лабораторных работ и загружаем файлы на Github (рис. fig. 3.19).

```

vlasovas@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git add .
vlasovas@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git commit -am '
feat(main): add files lab-4'
[master 05229f5] feat(main): add files lab-4
2 files changed, 36 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
vlasovas@vbox:~/work/study/2024-2025/Архитектура компьютера/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), 984 байта | 984.00 КиБ/с, готово.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:vlasovas52/study_2024-2025_arch-pc.git
  042b351..05229f5  master -> master
vlasovas@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$

```

Рис. 3.19: Загружаем файлы

4 Выводы

Мы познакомились с языком ассемблера NASM и создали две работающих программы.