

Отчет по лабораторной работе №4

Дисциплина: архитектура компьютера

Бизев Никита Владимирович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
4.1	Создание программы Hello world!	10
4.2	Работа с транслятором NASM	12
4.3	Работа с расширенным синтаксисом командной строки NASM . .	13
4.4	Работа с компоновщиком LD	15
4.5	Запуск исполняемого файла	18
5	Выводы	19
6	Список литературы	20

Список иллюстраций

4.1	Перемещение между директориями	10
4.2	Создание пустого файла	11
4.3	Открытие файла в текстовом редакторе	11
4.4	Заполнение файла	12
4.5	Компиляция текста программы	12
4.6	Проверка	13
4.7	Компиляция текста программы	14
4.8	Проверка	14
4.9	Передача объектного файла на обработку компоновщику	15
4.10	Проверка	16
4.11	Передача объектного файла на обработку компоновщику	17
4.12	Проверка	17
4.13	Запуск исполняемого файла	18

Список таблиц

1 Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

1. Создание программы Hello world!
2. Работа с транслятором NASM
3. Работа с расширенным синтаксисом командной строки NASM
4. Работа с компоновщиком LD
5. Запуск исполняемого файла
6. Выполнение заданий для самостоятельной работы.

3 Теоретическое введение

Основными функциональными элементами любой ЭВМ являются центральный процессор, память и периферийные устройства. Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих дорожек на материнской плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора входят следующие устройства: - арифметико-логическое устройство (АЛУ) — выполняет логические и арифметические действия, необходимые для обработки информации, хранящейся в памяти; - устройство управления (УУ) — обеспечивает управление и контроль всех устройств компьютера; - регистры — сверхбыстрая оперативная память небольшого объёма, входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные регистры. Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. Большинство команд в программах написанных на ассемблере используют регистры в качестве операндов. Практически все команды представляют собой преобразование данных хранящихся в регистрах процессора, это например пересылка данных между регистрами или между регистрами и памятью, преобразование

(арифметические или логические операции) данных хранящихся в регистрах. Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита. В качестве примера приведем названия основных регистров общего назначения (именно эти регистры чаще всего используются при написании программ): - RAX, RCX, RDX, RBX, RSI, RDI — 64-битные - EAX, ECX, EDX, EBX, ESI, EDI — 32-битные - AX, CX, DX, BX, SI, DI — 16-битные - AH, AL, CH, CL, DH, DL, BH, BL — 8-битные

Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). ОЗУ — это быстродействующее энергозависимое запоминающее устройство, которое напрямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно работает в текущий момент. ОЗУ состоит из одинаковых пронумерованных ячеек памяти. Номер ячейки памяти — это адрес хранящихся в ней данных. Периферийные устройства в составе ЭВМ: - устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных. - устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой.

В основе вычислительного процесса ЭВМ лежит принцип программного управления. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы.

Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции. При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. Он заключается в следующем: 1. формирование адреса в памяти очередной команды; 2.

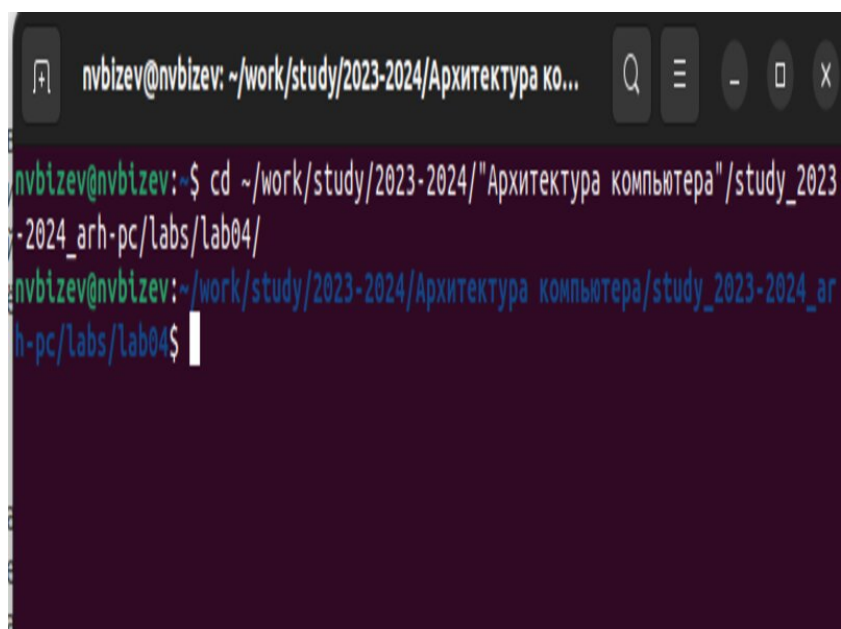
считывание кода команды из памяти и её дешифрация; 3. выполнение команды; 4. переход к следующей команде.

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64.

4 Выполнение лабораторной работы

4.1 Создание программы Hello world!

С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать (рис. [4.1]).



```
nvbizev@nvbizev: ~/work/study/2023-2024/Архитектура ко...  
nvbizev@nvbizev:~$ cd ~/work/study/2023-2024/"Архитектура компьютера"/study_2023-2024_arh-pc/labs/lab04/  
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab04$
```

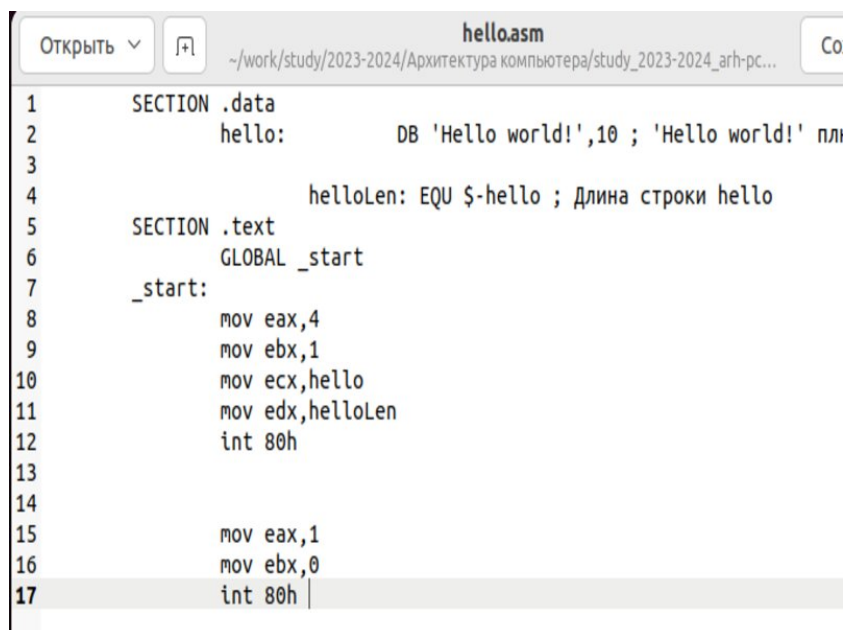
Рис. 4.1: Перемещение между директориями

Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch` (рис. [4.2]).

```
h-pc/labs/lab04$ touch hello.asm
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar
h-pc/labs/lab04$
```

Рис. 4.2: Создание пустого файла

Открываю созданный файл в текстовом редакторе. (рис. [4.3]).



The screenshot shows a text editor window titled 'hello.asm'. The file path is '~/.work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc...'. The code is as follows:

```
1 SECTION .data
2     hello:      DB 'Hello world!',10 ; 'Hello world!' пл
3
4     helloLen: EQU $-hello ; Длина строки hello
5 SECTION .text
6     GLOBAL _start
7     _start:
8         mov eax,4
9         mov ebx,1
10        mov ecx,hello
11        mov edx,helloLen
12        int 80h
13
14
15        mov eax,1
16        mov ebx,0
17        int 80h |
```

Рис. 4.3: Открытие файла в текстовом редакторе

Заполняю файл, вставляя в него программу для вывода “Hello word!” (рис.

[4.4]).

Заполнение файла

Рис. 4.4: Заполнение файла

4.2 Работа с транслятором NASM

Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду `nasm -f elf hello.asm`, ключ `-f` указывает транслятору `nasm`, что требуется создать бинарный файл в формате ELF (рис. [4.5]). Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “`hello.o`”.

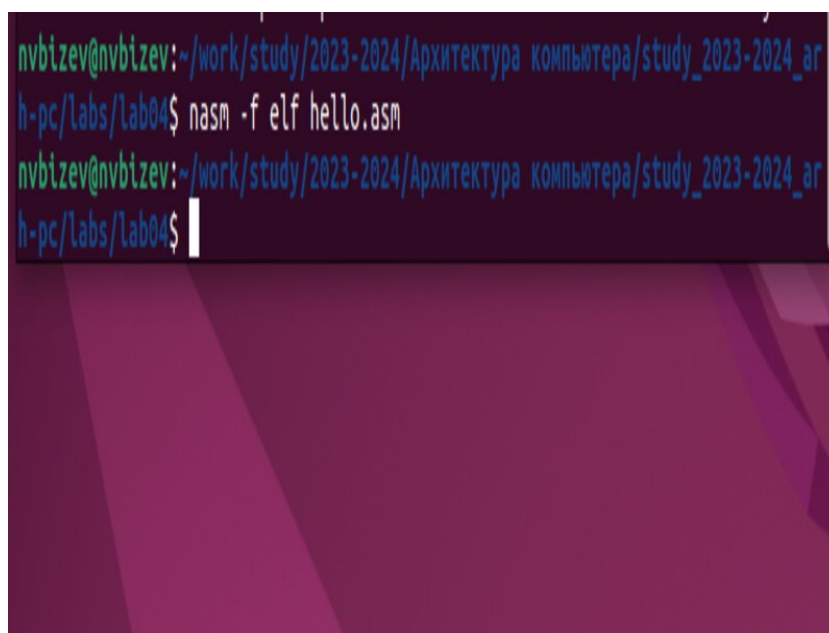
A screenshot of a terminal window with a dark background and light-colored text. The prompt is `nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar`. The user enters the command `h-pc/labs/lab04$ nasm -f elf hello.asm`. The prompt changes to `nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar` and the user enters `h-pc/labs/lab04$` followed by a cursor.

Рис. 4.5: Компиляция текста программы

Проверяю выполнение команды. (рис. [4.6])

```
h-pc/labs/lab04$ ls
hello.asm hello.o presentation report
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar
h-pc/labs/lab04$
```

Рис. 4.6: Проверка

4.3 Работа с расширенным синтаксисом командной строки NASM

Ввожу команду, которая скомпилирует файл `hello.asm` в файл `obj.o`, при этом в файл будут включены символы для отладки (ключ `-g`), также с помощью ключа `-l` будет создан файл листинга `list.lst` (рис. [4.7]). Далее проверяю с помощью утилиты `ls` правильность выполнения команды.

```
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar  
h-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm  
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar  
h-pc/labs/lab04$
```

Рис. 4.7: Компиляция текста программы

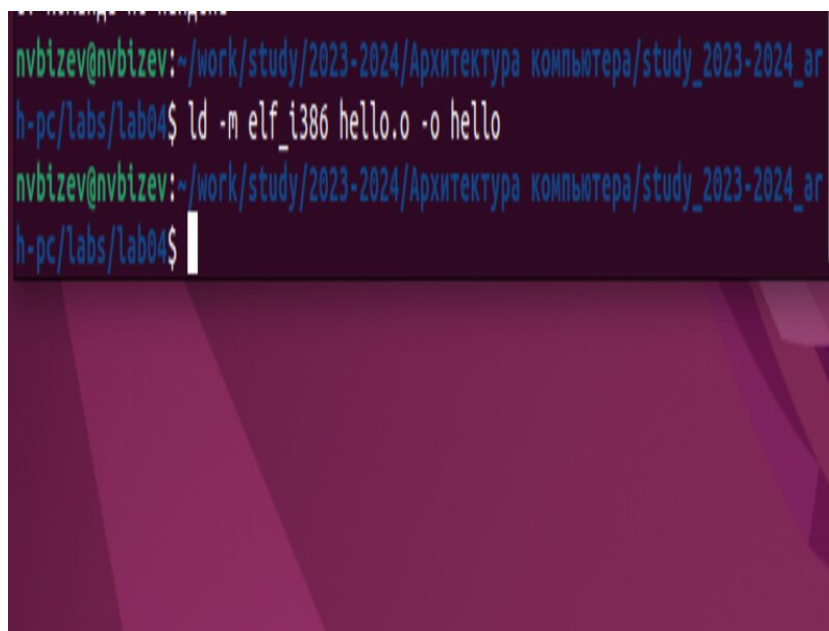
Проверяю выполнение команды. (рис. [4.8])

```
h-pc/labs/lab04$ ls  
hello.asm hello.o list.lst obj.o presentation report  
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar  
h-pc/labs/lab04$
```

Рис. 4.8: Проверка

4.4 Работа с компоновщиком LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (рис. [4.9]). Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды.



```
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar  
h-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello  
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar  
h-pc/labs/lab04$
```

Рис. 4.9: Передача объектного файла на обработку компоновщику

Проверяю выполнение команды. (рис. [4.10])

```
h-pc/labs/lab04$ ls
hello hello.asm hello.o list.lst obj.o presentation report
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar
h-pc/labs/lab04$
```

Рис. 4.10: Проверка

Выполняю следующую команду (рис. [4.11]). Исполняемый файл будет иметь имя `main`, т.к. после ключа `-o` было задано значение `main`. Объектный файл, из которого собран этот исполняемый файл, имеет имя `obj.o`


```
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar  
h-pc/labs/lab04$ ld -m elf_i386 obj.o -o main  
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar  
h-pc/labs/lab04$
```

Рис. 4.11: Передача объектного файла на обработку компоновщику

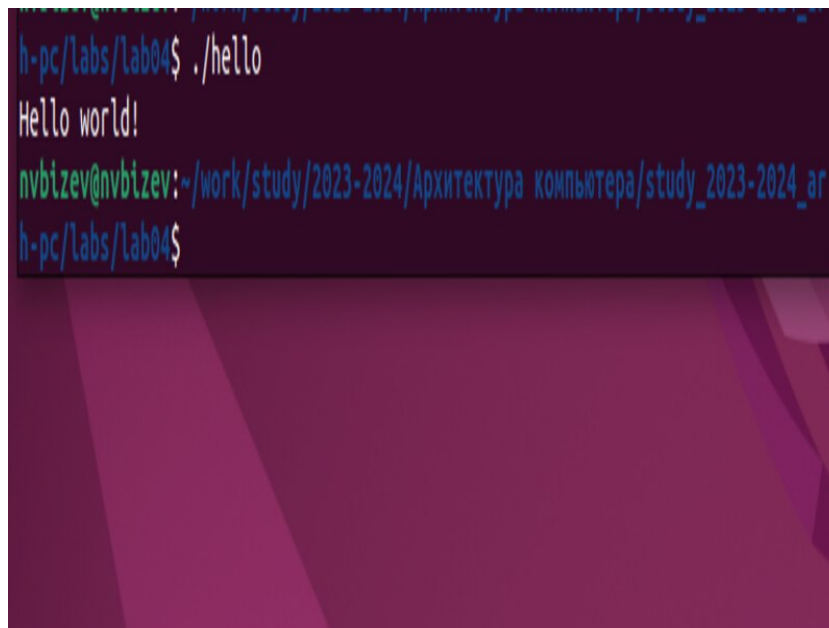
Проверяю выполнение команды. (рис. [4.12])

```
h-pc/labs/lab04$ ls  
hello hello.asm hello.o list.lst main obj.o presentation report  
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar  
h-pc/labs/lab04$
```

Рис. 4.12: Проверка

4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. [4.13]).

A terminal window with a dark purple background. The prompt is 'h-pc/labs/lab04\$'. The user enters './hello'. The output is 'Hello world!'. The prompt changes to 'nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar'. The user enters 'h-pc/labs/lab04\$'.

```
h-pc/labs/lab04$ ./hello
Hello world!
nvbizev@nvbizev:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_ar
h-pc/labs/lab04$
```

Рис. 4.13: Запуск исполняемого файла

5 Выводы

При выполнении данной лабораторной работы я освоил процедуры компиляции и сборки программ, написанных на ассемблере NASM.

6 Список литературы

1. https://esystem.rudn.ru/pluginfile.php/1584628/mod_resource/content/1/%D0%9B%D0%B0