

# **Лабораторная работа номер 4**

**Архитектура компьютера**

Титков Ярослав Максимович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>12</b>

## Список иллюстраций

3.1	Изменени 'Hello World' на моё имя . . . . .	11
3.2	Проверяю правильно ли скопировались файлы . . . . .	11
3.3	Запускаю код . . . . .	11

## **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. рекомендациями методического пособия и выданным вариантом.

### 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

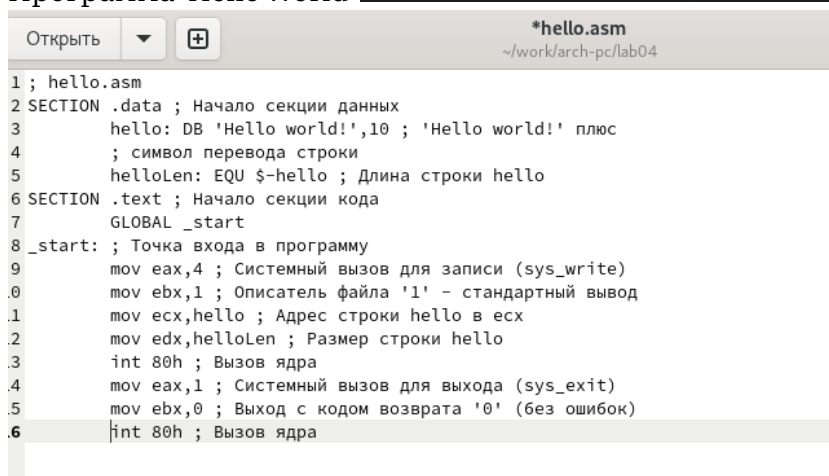
Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). ОЗУ — это быстродействующее энергозависимое запоминающее устройство, которое напрямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно работает в текущий 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. #  
Выполнение лабораторной работы

### 1. Программа 'Hello World'

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



The screenshot shows the gedit text editor with the file 'hello.asm' open. The code is as follows:

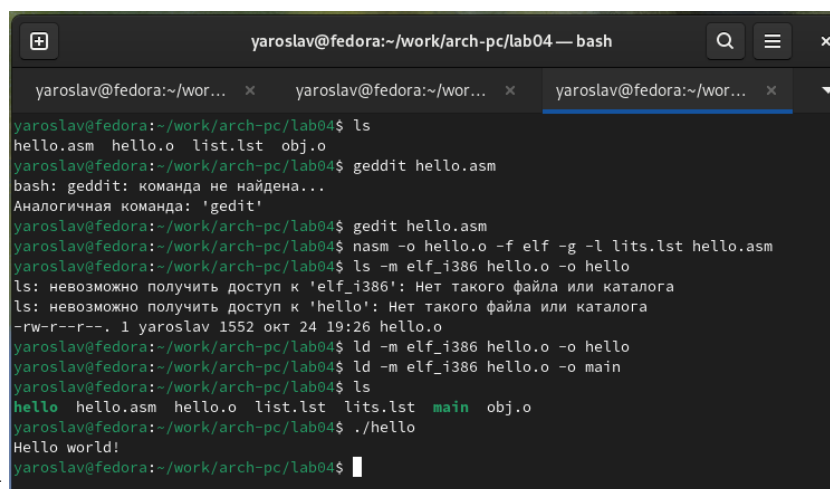
```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3     hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4           ; символ перевода строки
5     helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7     GLOBAL _start
8 _start: ; Точка входа в программу
9     mov 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    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
```

### 2. транслятор NASM и расширенный синтаксис командной строки NASM

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

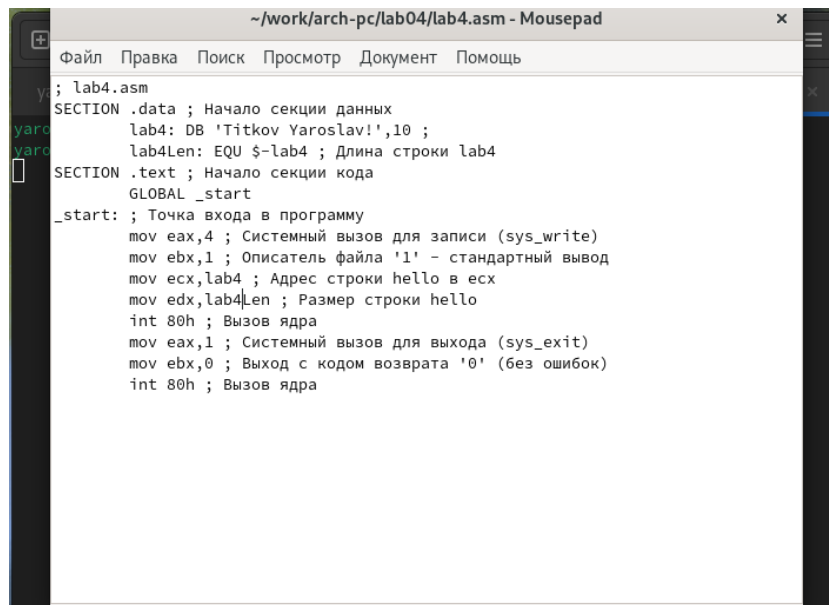
### 3. С помощью команды ls проверил, что исполняемый файл был создан, затем прописал все предложенные команды для корректности работы файлов и

затем запустил



```
yaroslav@fedora:~/work/arch-pc/lab04 — bash
yaroslav@fedora:~/work/arch-pc/lab04$ ls
hello.asm hello.o list.lst obj.o
yaroslav@fedora:~/work/arch-pc/lab04$ gedit hello.asm
bash: gedit: команда не найдена...
Аналогичная команда: 'gedit'
yaroslav@fedora:~/work/arch-pc/lab04$ gedit hello.asm
yaroslav@fedora:~/work/arch-pc/lab04$ nasm -o hello.o -f elf -g -l lits.lst hello.asm
yaroslav@fedora:~/work/arch-pc/lab04$ ls -m elf_i386 hello.o -o hello
ls: невозможно получить доступ к 'elf_i386': Нет такого файла или каталога
ls: невозможно получить доступ к 'hello': Нет такого файла или каталога
-rw-r--r--. 1 yaroslav 1552 окт 24 19:26 hello.o
yaroslav@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
yaroslav@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o main
yaroslav@fedora:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst lits.lst main obj.o
yaroslav@fedora:~/work/arch-pc/lab04$ ./hello
Hello world!
yaroslav@fedora:~/work/arch-pc/lab04$
```

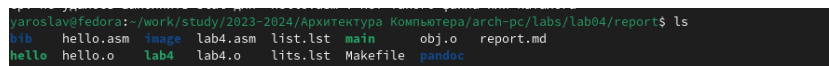
Задания для самостоятельной работы: 1. В каталоге ~/work/arch-pc/lab04 с помощью команды `cp` создайте копию файла `hello.asm` с именем `lab4.asm` 2. С помощью любого текстового редактора внесите изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с вашими фамилией и именем. 3. Оттранслируйте полученный текст программы `lab4.asm` в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл. 4. Скопируйте файлы `hello.asm` и `lab4.asm` в Ваш локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/. Загрузите файлы на Github без кода



```
~/.work/arch-pc/lab04/lab4.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

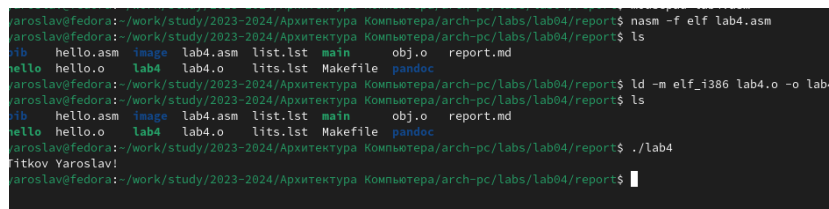
; lab4.asm
SECTION .data ; Начало секции данных
lab4: DB 'Titkov Yaroslav!',10 ;
lab4Len: EQU $-lab4 ; Длина строки lab4
SECTION .text ; Начало секции кода
GLOBAL _start
_start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,lab4 ; Адрес строки hello в ecx
mov edx,lab4Len ; 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. 3.1: Изменени ‘Hello World’ на моё имя



```
yaroslav@fedora: ~/work/study/2023-2024/Архитектура Компьютера/arch-pc/labs/lab04/report$ ls
b1b  hello.asm  image  lab4.asm  list.lst  main      obj.o  report.md
hello hello.o      lab4    lab4.o    lits.lst  Makefile  pandoc
```

Рис. 3.2: Проверяю правильно ли скопировались файлы



```
yaroslav@fedora: ~/work/study/2023-2024/Архитектура Компьютера/arch-pc/labs/lab04/report$ nasm -f elf lab4.asm
yaroslav@fedora: ~/work/study/2023-2024/Архитектура Компьютера/arch-pc/labs/lab04/report$ ls
b1b  hello.asm  image  lab4.asm  list.lst  main      obj.o  report.md
hello hello.o      lab4    lab4.o    lits.lst  Makefile  pandoc
yaroslav@fedora: ~/work/study/2023-2024/Архитектура Компьютера/arch-pc/labs/lab04/report$ ld -m elf_i386 lab4.o -o lab4
yaroslav@fedora: ~/work/study/2023-2024/Архитектура Компьютера/arch-pc/labs/lab04/report$ ls
b1b  hello.asm  image  lab4.asm  list.lst  main      obj.o  report.md
hello hello.o      lab4    lab4.o    lits.lst  Makefile  pandoc
yaroslav@fedora: ~/work/study/2023-2024/Архитектура Компьютера/arch-pc/labs/lab04/report$ ./lab4
Titkov Yaroslav!
yaroslav@fedora: ~/work/study/2023-2024/Архитектура Компьютера/arch-pc/labs/lab04/report$
```

Рис. 3.3: Запускаю код

## **4 Выводы**

Цель данной лабораторной работы заключалась в освоении процедур компиляции и сборки программ, написанных на ассемблере NASM.