

# Отчёт по лабораторной работе

## Дисциплина: архитектура компьютера

Шакиров Индус Равилевич

### Содержание

#### 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.

С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать

Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch`

Открываю созданный файл в текстовом редакторе `mouepad`

Заполняю файл, вставляя в него программу для вывода "Hello word!"

## 4 Выполнение лабораторной работы

С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать

```
ishakirov@dk8n78 ~ $ mkdir -p ~/work/arch-pc/lab04
ishakirov@dk8n78 ~ $
```

*Перемещение между директориями*

Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch`

```
ishakirov@dk8n78 ~ $ cd ~/work/arch-pc/lab04
ishakirov@dk8n78 ~/work/arch-pc/lab04 $
```

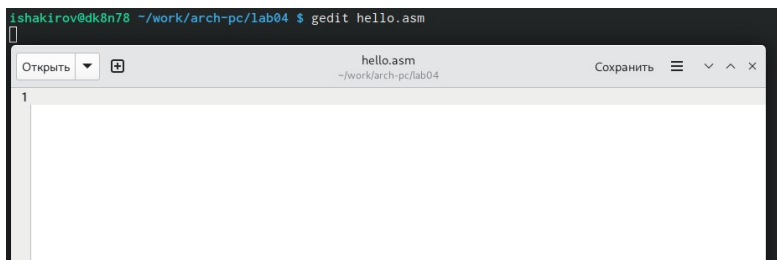
*Создание пустого файла*

Открываю созданный файл в текстовом редакторе `mousepad`

```
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ touch hello.asm
ishakirov@dk8n78 ~/work/arch-pc/lab04 $
```

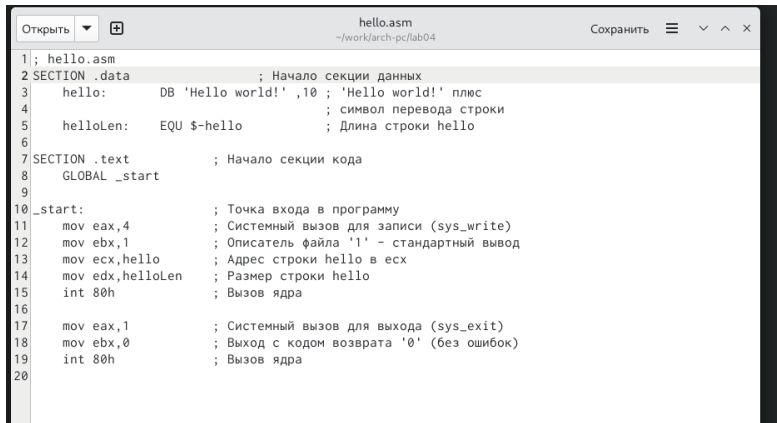
*Открытие файла в текстовом редакторе*

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



*Заполнение файла*

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`. Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “`hello.0`”.



```
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
6
7SECTION .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 0x08                 ; 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 0x08                 ; Вызов ядра
20
```

### Компиляция текста программы

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

```
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
ishakirov@dk8n78 ~/work/arch-pc/lab04 $
```

### Компиляция текста программы

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

```
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
ishakirov@dk8n78 ~/work/arch-pc/lab04 $
```

### Передача объектного файла на обработку компоновщику

Выполняю следующую команду. Исполняемый файл будет иметь имя таип, т.к. после ключа -o было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя bj.o

```
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
ishakirov@dk8n78 ~/work/arch-pc/lab04 $
```

### Передача объектного файла на обработку компоновщику

Запускаю на выполнение созданный исполняемый файл hello

```
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
```

### Запуск исполняемого файла

С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm

```
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ ./hello
Hello world!
```

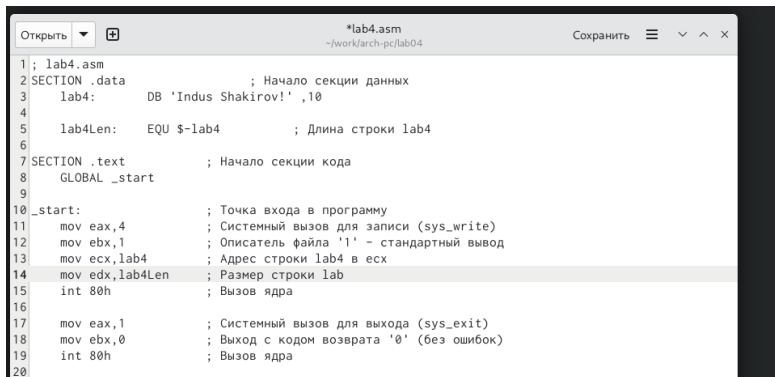
### Создание копии файла

Открываю файл и вношу изменения в программу, чтобы она выводила мои имя и фамилию.

```
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
ishakirov@dk8n78 ~/work/arch-pc/lab04 $
```

### Изменение программы

Компилирую текст программы в объектный файл.



### Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4.

```
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o
ishakirov@dk8n78 ~/work/arch-pc/lab04 $
```

### Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4.

```
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o
ishakirov@dk8n78 ~/work/arch-pc/lab04 $
```

### Запуск исполняемого файла

```
ishakirov@dk8n78 ~/work/arch-pc/lab04 $ ./lab4
Indus Shakirov!
ishakirov@dk8n78 ~/work/arch-pc/lab04 $
```

# Выводы

Здесь кратко описываются итоги проделанной работы.

## Список литературы

1.<https://esystem.rudn.ru/mod/page/view.php?id=1030505>