

Лабораторная работа №4

НКАбд-06-25

Хрисанова Ксения Олеговна

Содержание

1	Цель работы	1
2	Задание	1
3	Теоретическое введение	1
4	Выполнение лабораторной работы	1
5	Контрольные вопросы	Error! Bookmark not defined.
6	Выводы	Error! Bookmark not defined.
	Список литературы	Error! Bookmark not defined.

1 Цель работы

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

2 Задание

1. Создать каталог для работы с программами на языке ассемблера NASM.
2. Перейти в созданный каталог.
3. Создать текстовый файл **hello.asm** и ввести в него программу, выводящую сообщение *Hello world!* на экран.
4. Оттранслировать программу в объектный файл с помощью транслятора **NASM**.
5. Скомпоновать объектный файл с помощью компоновщика **LD** и получить исполняемый файл.
6. Запустить исполняемый файл и проверить корректность вывода сообщения.
7. Создать копию программы с именем **lab4.asm**, изменив текст выводимого сообщения на свои фамилию и имя.
8. Оттранслировать и скомпоновать изменённую программу.
9. Скопировать файлы **hello.asm** и **lab4.asm** в каталог лабораторных работ.
10. Загрузить файлы в свой репозиторий **GitHub**.

3 Теоретическое введение

Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства (рис. 4.1). 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 (системной) плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора (ЦП) 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 (половинки 16-битных регистров). Например, AH (high AX) — старшие 8 бит регистра AX, AL (low AX) — младшие 8 бит регистра AX. Таким образом можно отметить, что вы можете написать в своей программе, например, такие команды (mov – команда пересылки данных на языке ассемблера): `mov ax, 1` `mov eax, 1` Обе команды поместят в регистр AX число 1. Разница будет заключаться только в том, что вторая команда обнулит старшие разряды регистра EAX, то есть после выполнения второй команды в регистре EAX будет число 1. А первая команда оставит в старших разрядах регистра EAX старые данные. И если там были данные, отличные от нуля, то после выполнения первой команды в регистре EAX будет какое-то число, но не 1. А вот в регистре AX будет число 1. 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. В состав ЭВМ также входят периферийные устройства, которые можно разделить на: • устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных (жёсткие диски, твердотельные накопители, магнитные ленты); 32 Демидова А. В. Архитектура ЭВМ • 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. Программа состоит из машинных команд, которые указывают, какие операции и над какими данными (или операндами), в какой последовательности необходимо выполнить. Набор машинных команд определяется устройством конкретного процессора. 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. В самом общем виде 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. Данный алгоритм позволяет выполнить хранящуюся в ОЗУ программу. Кроме того, в зависимости от команды при её выполнении могут проходить не все этапы. Более подробно введение о теоретических основах архитектуры ЭВМ

4 Выполнение лабораторной работы

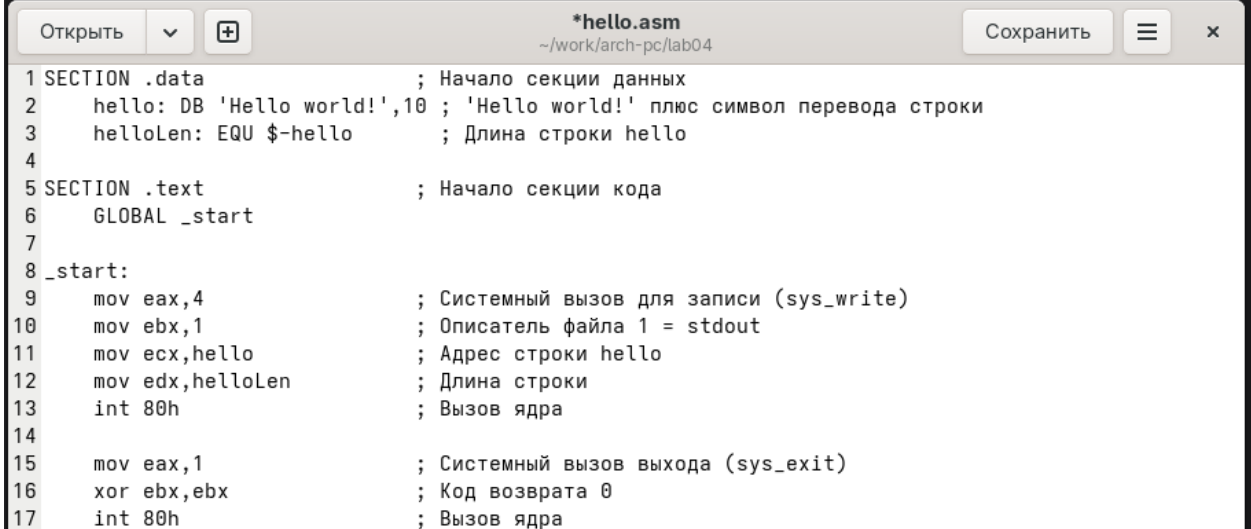
Сначала я создала каталог для лабораторной работы:(рис.1)

```
kseniahrisnova@fedora:~$ mkdir -p ~/work/arch-pc/lab04
kseniahrisnova@fedora:~$ cd ~/work/arch-pc/lab04
```

рис.1 – создание каталога

Далее создала текстовый файл программы, открыла файл для редактирования и ввела текст программы: (рис.2)

```
kseniahrisnova@fedora:~/work/arch-pc/lab04$ touch hello.asm
kseniahrisnova@fedora:~/work/arch-pc/lab04$ gedit hello.asm
```



```
1 SECTION .data                ; Начало секции данных
2     hello: DB 'Hello world!',10 ; 'Hello world!' плюс символ перевода строки
3     helloLen: EQU $-hello      ; Длина строки hello
4
5 SECTION .text                ; Начало секции кода
6     GLOBAL _start
7
8 _start:
9     mov eax,4                ; Системный вызов для записи (sys_write)
10    mov ebx,1                ; Описатель файла 1 = stdout
11    mov ecx,hello             ; Адрес строки hello
12    mov edx,helloLen          ; Длина строки
13    int 80h                  ; Вызов ядра
14
15    mov eax,1                ; Системный вызов выхода (sys_exit)
16    xor ebx,ebx              ; Код возврата 0
17    int 80h                  ; Вызов ядра
```

Рис.2 – создание и открытие файла для редактирования

Следующим шагом я выполнила трансляцию программы с помощью команды:

```
nasm -f elf hello.asm
```

Данная команда предназначена для трансляции (компиляции) исходного текста программы на языке ассемблера NASM в объектный файл. После выполнения я проверила содержимое каталога с помощью команды `ls`, убедившись, что объектный файл `hello.o` был успешно создан.(рис.3.1)

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

Рис. 3.1 – Трансляция программы и проверка создания объектного файла.

Далее я использовала расширенный вариант команды NASM:

```
nasm -o obj.o -f elf -g -l list.lst hello.asm
```

Эта команда выполняет трансляцию программы с созданием объектного файла `obj.o`, файла листинга `list.lst` и добавлением отладочной информации (параметр `-g`). Проверка содержимого каталога показала наличие всех файлов: `hello.asm`, `hello.o`, `list.lst`, `obj.o`.(рис.3.2)

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

Рис. 3.2 – Расширенная трансляция программы.

Затем я выполнила компоновку объектных файлов с помощью команд:

```
ld -m elf_i386 hello.o -o hello
```

```
ld -m elf_i386 obj.o -o main
```

Команды ld предназначены для сборки исполняемых файлов из объектных. Параметр -m elf_i386 указывает, что используется 32-битная архитектура, а -o задаёт имя выходного исполняемого файла. В результате были созданы два исполняемых файла: hello и main.(рис.3.3)

```
kseniahrisanoval@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
kseniahrisanoval@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
kseniahrisanoval@fedora:~/work/arch-pc/lab04$ ./hello
Hello world!
kseniahrisanoval@fedora:~/work/arch-pc/lab04$
```

Рис. 3.3 – Компоновка объектных файлов.

Далее я выполнила создание и запуск программы, выводящей на экран строку с моими фамилией и именем.

1. В каталоге ~/work/arch-pc/lab04 я создала копию исходного файла программы **hello.asm** с помощью команды: `cp hello.asm lab4.asm`(рис. 4.1)

```
kseniahrisanoval@fedora:~/work/arch-pc/lab04$ cd ~/work/arch-pc/lab04
kseniahrisanoval@fedora:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
```

Рис.4.1 – создание копии файла программы

2. Внесла изменения в текст программы, заменив строку вывода *Hello world!* на *Хрисанова Ксения!*.(рис.4.2)

```
kseniahrisanoval@fedora:~/work/arch-pc/lab04$ sed -i 's/Hello world!/Хрисанова Ксения!/g' lab4.asm
```

Рис.4.2 – изменение текста

3. Выполнила трансляцию изменённого файла в объектный с помощью транслятора NASM, затем выполнила компоновку объекта в исполняемый файл:
После этого запустила программу командой
В результате на экране появилось сообщение: Хрисанова Ксения! (рис.4.3)

```
kseniahrisanoval@fedora:~/work/arch-pc/lab04$ nasm -f elf lab4.asm -o lab4.o
kseniahrisanoval@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
kseniahrisanoval@fedora:~/work/arch-pc/lab04$ ./lab4
Хрисанова Ксения!
```

Рис.4.3 – компоновка объекта и запуск файла

На завершающем этапе я перешла в каталог ~/work/arch-pc/lab04, где были созданы и собраны файлы программы.

После проверки содержимого каталога я убедилась, что присутствуют все необходимые файлы.

Затем с помощью команды

```
cp hello.asm lab4.asm ~/work/study_2025-2026_arh-pc/labs/lab04/
```

файлы **hello.asm** и **lab4.asm** были скопированы в каталог лабораторных работ репозитория.

```
kseniahrisanoval@fedora:~/work/arch-pc/lab04$ cp hello.asm lab4.asm ~/work/study_2025-2026_arh-pc/labs/lab04/
kseniahrisanoval@fedora:~/work/arch-pc/lab04$ ls ~/work/study_2025-2026_arh-pc/labs/lab04/
hello.asm lab4.asm
```

5 Вывод

В ходе выполнения лабораторной работы была изучена структура и принципы написания программ на языке ассемблера **NASM**.

6 Список литературы

Лабораторная работа №4 (Архитектура ОС).