

Отчёт по лабораторной работе №4

Дисциплина: архитектура компьютера

Самойлова Софья Дмитриевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
4.1	Программа Hello world!	10
4.2	Транслятор NASM	11
4.3	Расширенный синтаксис командной строки NASM	12
4.4	Компоновщик LD	12
4.5	Запуск исполняемого файла	13
5	Выполнение самостоятельной работы	14
6	Выводы	16

Список иллюстраций

4.1	Создание каталога и файла hello.asm	10
4.2	Редактирование файла hello.asm	10
4.3	Работа транслятора	11
4.4	Расширенный вариант работы командной строки NASM	12
4.5	Работа транслятора	13
4.6	Работа транслятора	13
5.1	Копирование hello.asm	14
5.2	Редактирование программы	14
5.3	Трансляция программы	15
5.4	Компоновка и запуск программы	15

Список таблиц

1 Цель работы

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

2 Задание

1. Программа Hello world!
2. Транслятор NASM
3. Расширенный синтаксис командной строки NASM
4. Компоновщик 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

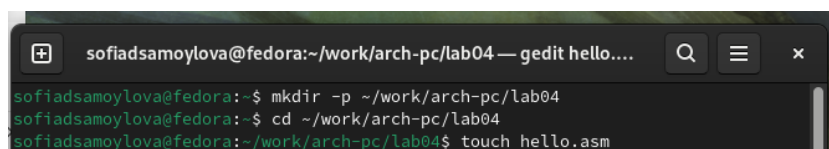
хранящихся в регистрах. Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита. В качестве примера приведем названия основных регистров общего назначения (именно 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 Программа Hello world!

Рассмотрим пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит приветственное сообщение Hello world! на экран. Создаём каталог для работы с программами на языке ассемблера NASM, переходим в созданный каталог и создаем текстовый файл с именем hello.asm (рис. 4.1).



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

Рис. 4.1: Создание каталога и файла hello.asm

Открываем этот файл с помощью любого текстового редактора, например, gedit и вводим в него код (рис. 4.2):



```
*hello.asm
~/work/arch-pc/lab04

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
```

Рис. 4.2: Редактирование файла hello.asm

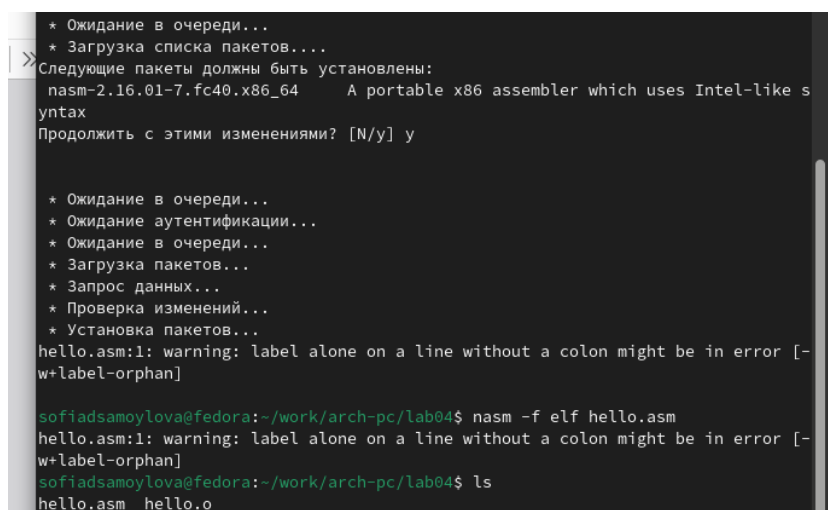
Важно: в отличие от многих современных высокоуровневых языков программирования, в ассемблерной программе каждая команда располагается на отдельной строке. Размещение нескольких команд на одной строке недопустимо. Синтаксис ассемблера NASM является чувствительным к регистру, т.е. есть разница между большими и малыми буквами.

4.2 Транслятор NASM

NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World» необходимо написать:

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

Если текст программы набран без ошибок, то транслятор преобразует текст программы из файла `hello.asm` в объектный код, который запишется в файл `hello.o`. Таким образом, имена всех файлов получаются из имени входного файла и расширения по умолчанию. При наличии ошибок объектный файл не создаётся, а после запуска транслятора появятся сообщения об ошибках или предупреждения. С помощью команды `ls` проверяем, что объектный файл был создан. Объектный файл был назван `hello.o` (рис. 4.3):



```
* Ожидание в очереди...
* Загрузка списка пакетов...
Следующие пакеты должны быть установлены:
nasm-2.16.01-7.fc40.x86_64      A portable x86 assembler which uses Intel-like s
yntax
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...
hello.asm:1: warning: label alone on a line without a colon might be in error [-
w+label-orphan]

sofiadsamoylova@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm
hello.asm:1: warning: label alone on a line without a colon might be in error [-
w+label-orphan]
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
```

Рис. 4.3: Работа транслятора

NASM не запускают без параметров. Ключ -f указывает транслятору, что требуется создать бинарные файлы в формате ELF. Следует отметить, что формат elf64 позволяет создавать исполняемый код, работающий под 64-битными версиями Linux. Для 32-битных версий ОС указываем в качестве формата просто elf. NASM всегда создаёт выходные файлы в текущем каталоге

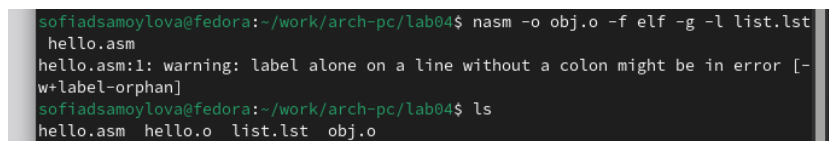
4.3 Расширенный синтаксис командной строки NASM

Полный вариант командной строки nasm выглядит следующим образом:

```
nasm [-@ косвенный_файл_настроек] [-o объектный_файл] [-f формат_объектного_файла] [-l листинг] [параметры...] [--] исходный_файл
```

Выполняем следующую команду и с помощью команды ls проверяем, что файлы были созданы.(рис. 4.4):

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



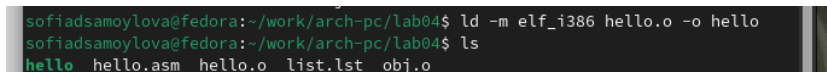
```
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
hello.asm:1: warning: label alone on a line without a colon might be in error [-w+label-orphan]
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.4: Расширенный вариант работы командной строки NASM

Данная команда скомпилирует исходный файл hello.asm в obj.o (опция -o позволяет задать имя объектного файла, в данном случае obj.o), при этом формат выходного файла будет elf, и в него будут включены символы для отладки (опция -g), кроме того, будет создан файл листинга list.lst (опция -l).

4.4 Компоновщик LD

Чтобы получить исполняемую программу, объектный файл передаем на обработку компоновщику и с помощью команды ls проверяем, что исполняемый файл hello был создан (рис. 4.5):

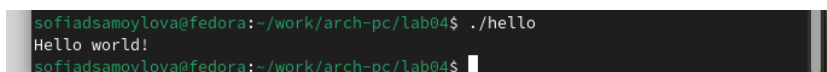


```
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.5: Работа транслятора

4.5 Запуск исполняемого файла

Запустим на выполнение созданный исполняемый файл, находящийся в текущем каталоге, набрав в командной строке: (рис. 4.6): `./hello`

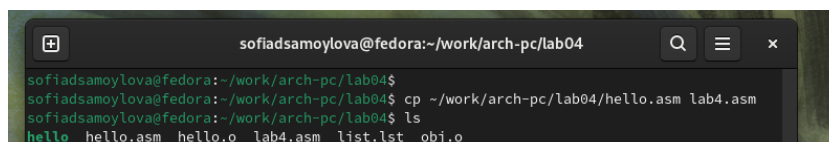


```
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ ./hello
Hello world!
sofiadsamoylova@fedora:~/work/arch-pc/lab04$
```

Рис. 4.6: Работа транслятора

5 Выполнение самостоятельной работы

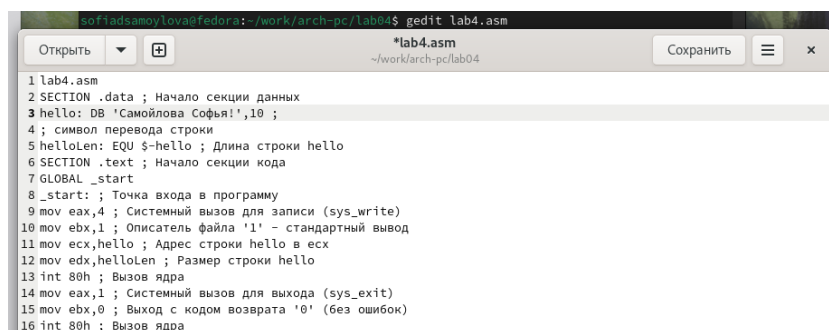
В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создаю копию файла `hello.asm` с именем `lab4.asm` (рис. 5.1):



```
sofiadsamoylova@fedora:~/work/arch-pc/lab04
sofiadsamoylova@fedora:~/work/arch-pc/lab04$
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ cp ~/work/arch-pc/lab04/hello.asm lab4.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  obj.o
```

Рис. 5.1: Копирование `hello.asm`

Вношу изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с моими фамилией и именем (рис. 5.2):



```
*lab4.asm
~/work/arch-pc/lab04
Сохранить

1 lab4.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Самойлова Софья!',10 ;
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
```

Рис. 5.2: Редактирование программы

Транслирую полученный текст программы `lab4.asm` в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл.(рис. 5.3):

```
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
lab4.asm:1: warning: label alone on a line without a colon might be in error [-w+label-orphan]
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ ls
hello.asm hello.o lab4.asm lab4.o list.lst obj.o
sofiadsamoylova@fedora:~/work/arch-pc/lab04$
```

Рис. 5.3: Трансляция программы

(рис. 5.4):

```
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ ls
hello.asm hello.o lab4.asm lab4.o list.lst obj.o
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst lab4.asm
lab4.asm:1: warning: label alone on a line without a colon might be in error [-w+label-orphan]
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ ls
hello.asm hello.o lab4.asm lab4.o list.lst obj.o
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ ls
hello.asm hello.o lab4 lab4.asm lab4.o list.lst obj.o
sofiadsamoylova@fedora:~/work/arch-pc/lab04$ ./lab4
Самойлова Софья!
sofiadsamoylova@fedora:~/work/arch-pc/lab04$
```

Рис. 5.4: Компоновка и запуск программы

Копирую файлы hello.asm и lab4.asm в свой локальный репозиторий в каталог ~/work/study/2024-2025/“Архитектура компьютера”/arch-pc/labs/lab04/. Загружаю файлы на Github

6 Выводы

Я освоила процедуры оформления отчетов с помощью легковесного языка разметки Markdown