

Шаблон отчёта по лабораторной работе №4

Дисциплина: архитектура компьютера

Кайнова Алина Андреевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	9
4.1	Создание программы Hello World!	9
4.2	Работа с транслятором NASM	10
4.3	Работа с расширенным синтаксисом командной строки NASM . .	11
4.4	Работа с компоновщиком LD	11
4.5	Запуск исполняемого файла	11
4.6	Выполнение заданий для самостоятельной работы	12
5	Выводы	15
	Список литературы	16

Список иллюстраций

4.1	Создание каталога	9
4.2	Перемещение между директориями	9
4.3	Создание файла	9
4.4	Открытие файла в текстовом редакторе	10
4.5	Заполнение файла	10
4.6	Компиляция текста программы	10
4.7	Компиляция текста программы	11
4.8	Передача файла на обработку компоновщику	11
4.9	Передача файла на обработку компоновщику	11
4.10	Запуск исполняемого файла	12
4.11	Создании копии файла	12
4.12	Изменение файла	12
4.13	Компиляция текста программы	13
4.14	Передача файла на обработку компоновщику	13
4.15	Запуск исполняемого файла	13
4.16	Копирование файлов в нужный каталог	13
4.17	Проверка копирования	13
4.18	Добавление файлов	14
4.19	Сохранение изменений	14
4.20	Отправка файлов	14

1 Цель работы

Освоить процедуру компиляции и сборки программ 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!

Создаю каталог для лабораторной работы №4

```
[aakayjnova@fedora ~]$ mkdir -p ~/work/arch-pc/lab04  
[aakayjnova@fedora ~]$ |
```

Рис. 4.1: Создание каталога

Перемещаюсь в каталог, в котором буду работать

```
[aakayjnova@fedora ~]$ mkdir -p ~/work/arch-pc/lab04  
[aakayjnova@fedora ~]$ cd ~/work/arch-pc/lab04  
[aakayjnova@fedora lab04]$ |
```

Рис. 4.2: Перемещение между директориями

Создаю пустой текстовый файл hello.asm

```
[aakayjnova@fedora lab04]$ touch hello.asm  
[aakayjnova@fedora lab04]$ |
```

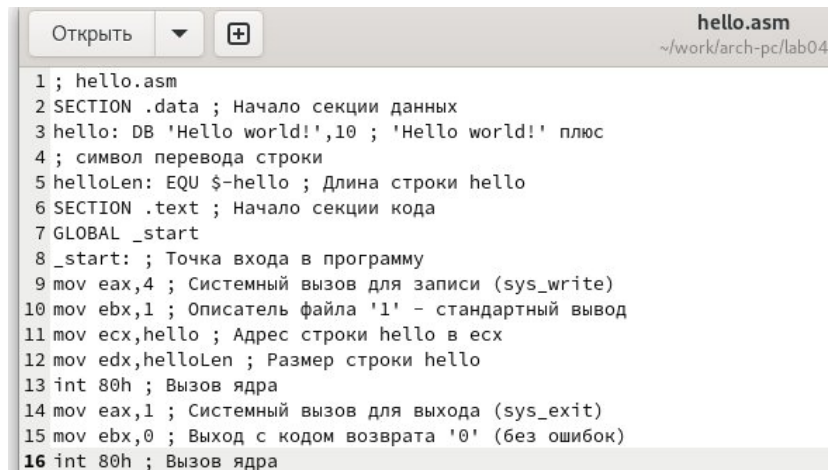
Рис. 4.3: Создание файла

Открываю созданный файл в текстовом редакторе gedit

```
[aakayjnova@fedora lab04]$ gedit hello.asm
```

Рис. 4.4: Открытие файла в текстовом редакторе

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



```
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. 4.5: Заполнение файла

4.2 Работа с транслятором NASM

Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM и проверяю правильность выполнения команды

```
[aakayjnova@fedora lab04]$ nasm -f elf hello.asm
[aakayjnova@fedora lab04]$ ls
hello.asm hello.o
```

Рис. 4.6: Компиляция текста программы

4.3 Работа с расширенным синтаксисом командной строки NASM

Ввожу команду для компиляции файла `hello.asm` в файл `obj.o` и проверяю правильность выполнения программы

```
[aakayjnova@fedora lab04]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[aakayjnova@fedora lab04]$ ls
hello.asm hello.o list.lst obj.o
```

Рис. 4.7: Компиляция текста программы

4.4 Работа с компоновщиком LD

Передаю объектный файл `hello.o` на обработку компоновщику LD, чтобы получить исполняемый файл `hello`, и проверяю правильность выполнения команды

```
[aakayjnova@fedora lab04]$ ld -m elf_i386 hello.o -o hello
[aakayjnova@fedora lab04]$ ls
hello hello.asm hello.o list.lst obj.o
```

Рис. 4.8: Передача файла на обработку компоновщику

Ввожу необходимую команду и проверяю правильность её выполнения

```
[aakayjnova@fedora lab04]$ ld -m elf_i386 obj.o -o main
[aakayjnova@fedora lab04]$ ls
hello hello.asm hello.o list.lst main obj.o
```

Рис. 4.9: Передача файла на обработку компоновщику

4.5 Запуск исполняемого файла

Запускаю на выполнение созданный файл `hello`

```
[aakayjnova@fedora lab04]$ ./hello
Hello world!
[aakayjnova@fedora lab04]$ |
```

Рис. 4.10: Запуск исполняемого файла

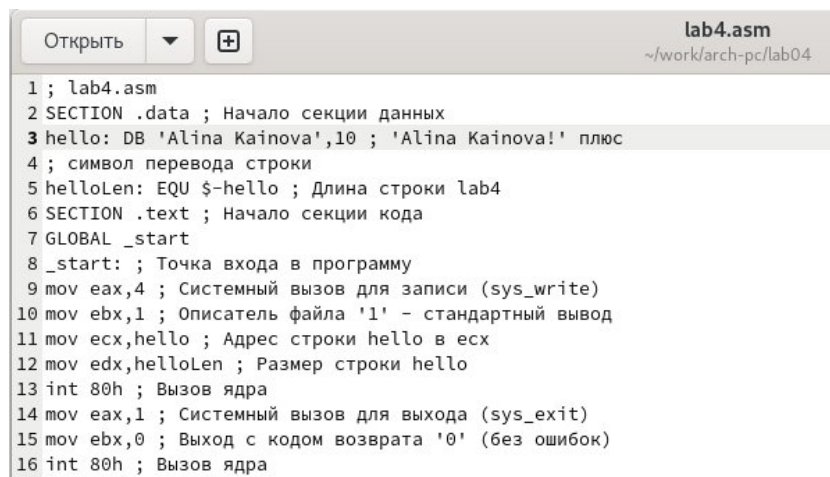
4.6 Выполнение заданий для самостоятельной работы

Копирую файл `hello.asm` с новым именем `lab4.asm` и проверяю правильность копирования

```
[aakayjnova@fedora lab04]$ cp hello.asm lab4.asm
[aakayjnova@fedora lab04]$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
```

Рис. 4.11: Создании копии файла

С помощью текстового редактора `gedit` открываю файл `lab4.asm` и вношу изменения, чтобы программа выводила мои имя и фамилию



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

1 ; lab4.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Alina Kainova',10 ; 'Alina Kainova!' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки lab4
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. 4.12: Изменение файла

Компилирую файл `lab4.asm` в объектный файл `lab4.o` и проверяю правильность компиляции

```
[aakayjnova@fedora lab04]$ nasm -f elf lab4.asm
[aakayjnova@fedora lab04]$ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o
```

Рис. 4.13: Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD

```
[aakayjnova@fedora lab04]$ ld -m elf_i386 lab4.o -o lab4
[aakayjnova@fedora lab04]$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
```

Рис. 4.14: Передача файла на обработку компоновщику

Запускаю исполняемый файл lab4

```
[aakayjnova@fedora lab04]$ ./lab4
Alina Kainova
[aakayjnova@fedora lab04]$ |
```

Рис. 4.15: Запуск исполняемого файла

Копирую файлы hello.asm и lab4.asm в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04

```
[aakayjnova@fedora lab04]$ cp hello.asm lab4.asm ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04
```

Рис. 4.16: Копирование файлов в нужный каталог

Проверяю правильность копирования файлов

```
[aakayjnova@fedora lab04]$ ls ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04
hello.asm lab4.asm presentation report
[aakayjnova@fedora lab04]$ |
```

Рис. 4.17: Проверка копирования

Добавляю изменения на GitHub

```
[aakayjnova@fedora arch-pc]$ git add .
```

Рис. 4.18: Добавление файлов

Сохраняю добавленные изменения

```
[aakayjnova@fedora arch-pc]$ git commit -m "Add files for lab04"  
[main 5ff30e5] Add files for lab04  
10 files changed, 50 insertions(+)
```

Рис. 4.19: Сохранение изменений

Отправляю файлы на сервер

```
[aakayjnova@fedora arch-pc]$ git push  
Перечисление объектов: 9, готово.  
Подсчет объектов: 100% (9/9), готово.
```

Рис. 4.20: Отправка файлов

5 Выводы

В ходе данной лабораторной работы мы научились компилировать программы и выполнять сборку программ, написанных на ассемблере NASM.

Список литературы

- [illegible]