

Отчет по лабораторной работе №4

Дисциплина: Архитектура компьютеров

Малинина Анастасия Игоревна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
4.1	Создание программы Hello world!	10
4.2	Запуск исполняемого файла.	12
4.3	Задание для самостоятельной работы	12
5	Выводы	14

Список иллюстраций

4.1	создание каталога	10
4.2	создание и открытие текстового файла	10
4.3	ввод текста	11
4.4	ввод команды	11
4.5	проверка	11
4.6	компиляция файла	11
4.7	обработка	12
4.8	ввод команды	12
4.9	запуск файла	12
4.10	ввод команды	12
4.11	изменения в программе	12
4.12	запуск файла	13
4.13	копирование файлов	13
4.14	загрузка	13

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.

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:

1. арифметико-логическое устройство (АЛУ) — выполняет логические и арифметические действия, необходимые для обработки информации, хранящейся в памяти;
2. устройство управления (УУ) — обеспечивает управление и контроль всех устройств компьютера;
3. 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:

1. устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных.
2. 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:

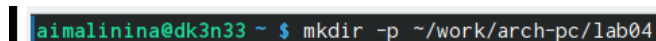
1. формирование адреса в памяти очередной команды;
2. считывание кода команды из памяти и её дешифрация;
3. выполнение команды;
4. переход к следующей команде.

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!

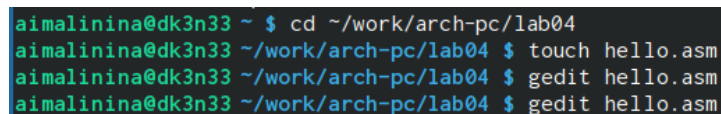
Создаю каталог для работы с программами на языке ассемблера NASM:(рис. 4.1).



```
aimalinina@dk3n33 ~ $ mkdir -p ~/work/arch-pc/lab04
```

Рис. 4.1: создание каталога

Перехожу в созданный каталог. Создаю текстовый файл с именем hello.asm и открываю этот файл с помощью любого текстового редактора, например, gedit. (рис. 4.2).



```
aimalinina@dk3n33 ~ $ cd ~/work/arch-pc/lab04
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ touch hello.asm
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ gedit hello.asm
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 4.2: создание и открытие текстового файла

Ввожу в него следующий текст:(рис. 4.3).

```

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
7 SECTION .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 80h; 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 80h; Вызов ядра

```

Рис. 4.3: ввод текста

NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World» необходимо написать:(рис. 4.4).

```

aimalinina@dk3n33 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm

```

Рис. 4.4: ввод команды

С помощью команды ls проверяю, что объектный файл был создан. (рис. 4.5).

```

aimalinina@dk3n33 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o list.lst main obj.o

```

Рис. 4.5: проверка

Выполняю следующую команду, которая скомпилирует исходный файл hello.asm в obj.o: (рис. 4.6).

```

aimalinina@dk3n33 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o list.lst main obj.o

```

Рис. 4.6: компиляция файла

Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику:(рис. 4.7).

```
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
```

Рис. 4.7: обработка

Выполняю следующую команду:(рис. 4.8).

```
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
```

Рис. 4.8: ввод команды

4.2 Запуск исполняемого файла.

Запустить на выполнение созданный исполняемый файл, находящийся в текущем каталоге, можно, набрав в командной строке:(рис. 4.9).

```
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ ./hello
Hello world!
```

Рис. 4.9: запуск файла

4.3 Задание для самостоятельной работы

В каталоге ~/work/arch-pc/lab04 с помощью команды cp создаю копию файла hello.asm с именем lab4.asm :(рис. 4.10).

```
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
```

Рис. 4.10: ввод команды

С помощью текстового редактора вношу изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с фило:(рис. 4.11).

```
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ gedit lab4.asm
```

Рис. 4.11: изменения в программе

Оттранслирую полученный текст программы lab4.asm в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл:(рис. 4.12).

```
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ gedit lab4.asm
(gedit:9142): GVFS-RemoteVolumeMonitor-WARNING **: 15:34:00.186: remote volume monitor with dbus name org.gtk.vfs.GoaVolumeMonitor is not supported
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ ./lab4
Malinina Anastasia
```

Рис. 4.12: запуск файла

Копирую файлы hello.asm и lab4.asm в локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/.(рис. 4.13).

```
aimalinina@dk3n33 ~/work/arch-pc/lab04 $ cp lab4.asm ~/work/study/2024-2025/“Архитектура компьютера”/arch-pc/labs/lab04/
```

Рис. 4.13: копирование файлов

Загружаю файлы на Github.:(рис. 4.14).

```
aimalinina@dk3n33 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04 $ git add .
aimalinina@dk3n33 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04 $ git commit -am 'feat(main): add files lab-4'
git: «commitgit» не является командой git. Смотрите «git --help».
aimalinina@dk3n33 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04 $ git commit -am 'feat(main): add files lab-4'
[master c46370a] feat(main): add files lab-4
4 files changed, 40 insertions(+), 1 deletion(-)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
aimalinina@dk3n33 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04 $ git push
Перечисление объектов: 17, готово.
Подсчет объектов: 100% (17/17), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (10/10), готово.
Запись объектов: 100% (10/10), 5.93 КБ | 1.19 МБ/с, готово.
Total 10 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 5 local objects.
To github.com:seokiri/name-study_2024-20245arh-pc.git
07bc8cc...c46370a master -> master
```

Рис. 4.14: загрузка

5 Выводы

При выполнении данной лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.