

Отчёт по лабораторной работе №4

Дисциплина: архитектура компьютеров

Карпачев Ярослав Олегович

Содержание

1	Цель работы	1
2	Задание	1
3	Теоретическое введение.....	1
4	Выполнение лабораторной работы.....	3
5	Выводы.....	6

1 Цель работы

Цель - 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 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); • 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

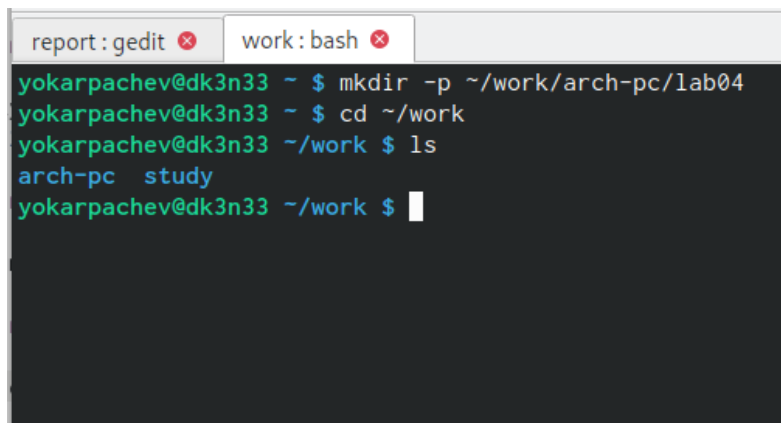
многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции. При выполнении 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 см. в [9; 11]

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня, таких как C/C++, Perl, Python и пр.

4 Выполнение лабораторной работы

1. Программа Hello world!

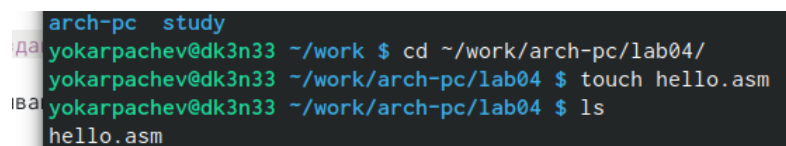
Создание каталога с помощью команды `mkdir` и проверка `ls`. (рис.1 [??]).



```
report: gedit x work: bash x
yokarpachev@dk3n33 ~ $ mkdir -p ~/work/arch-pc/lab04
yokarpachev@dk3n33 ~ $ cd ~/work
yokarpachev@dk3n33 ~/work $ ls
arch-pc  study
yokarpachev@dk3n33 ~/work $
```

рис.1 Создание каталога

Перехожу в каталог `~/work/arch-pc` и создаю текстовый файл с именем `hello.asm` (рис.2 [??]).



```
arch-pc study
да yokarpachev@dk3n33 ~/work $ cd ~/work/arch-pc/lab04/
ива yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ touch hello.asm
hello.asm
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ ls
hello.asm
```

рис.2 Создание файла

Открываю созданный файл с помощью vim и ввожу программу (рис.3 [??]).

```
SECTION .data
    hello: DB 'Hello world!', 10

    helloLen: EQU $-hello

SECTION .text
    GLOBAL _start

_start:
    mov eax,4
    mov ebx,1
    mov ecx,hello
    mov edx,helloLen
    int 80h

    mov eax,1
    mov ebx,0
    int 80h
```

рис.3 Введение программы

2. NASM

Превращаю текст программы в объектный код, используя команду `nasm -f elf hello.asm`. (рис.4 [??]).

```
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $
```

рис.4 Создание объектного файла

3. Расширенный синтаксис командной строки NASM

Ввожу команду `nasm -o obj.o -f elf -g -l list.lst hello.asm`. Данная команда компилирует файл в `obj.o`, кроме того, будет создан файл листинга `list.lst` (опция `-l`). (рис.5 [??]).

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

рис.5 Компиляция программы

4. Компоновщик

Передаю объектный файл на обработку компоновщику, получаю исполняемый файл (рис.6 [??]).

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

рис.6 Передача файла

Выполняю команду `ld -m elf_i386 obj.o -o main` (рис.7 [??]).

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

рис.7 Передача файла компоновщику

5. Запуск исполняемого файла

Запускаю созданный исполняемый файл. На экран выводится Hello word! (рис.8 [??]).

```
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ cd ~/work/arch-pc/lab04
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $
```

самостоятельной работы

Перехожу в каталог `~/work/arch-pc/lab04`, создаю копию файла `hello.asm` с именем `lab4.asm` с помощью команды `cp`. С помощью текстового редактора открываю файл `lab4.asm` и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис.9 [??]).

```
SECTION .data
hello: DB 'Karpachev Yaroslav', 10
helloLen: EQU $-hello

SECTION .text
GLOBAL _start
_start:
    mov eax,4
    mov ebx,1
    mov ecx,hello
    mov edx,helloLen
    int 80h

    mov eax,1
    mov ebx,0
    int 80h
```

рис.9 Копирование файла и введение программы

Создаю объектный файл, передаю на обработку компоновщику. Полученный исполняемый файл запускаю. На экран выводятся моё имя и моя фамилия.(рис.10 [??]).

```

yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ ./lab4
bash: ./lab4: команда не найдена
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ ./lab4
Karpachev Yaroslav
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $

```

рис.10 Компиляция, запуск и обработка исполняемого файла

Копирование файлов в репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/study_2023-2024_arh-pc/labs/lab04 (рис.11 [??]).

```

yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ cp hello.o ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab04
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ cp lab4.asm ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab04
yokarpachev@dk3n33 ~/work/arch-pc/lab04 $ cp lab4.asm ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab04
yokarpachev@dk3n33 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab04 $ rm hello.o
yokarpachev@dk3n33 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab04 $ ls
hello.asm lab4.asm presentation report
yokarpachev@dk3n33 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab04 $

```

рис.11 Копирование файлов

Добавляю файлы в коммит (рис.12 [??]).

```

yokarpachev@dk3n33 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab04 $ git pull
Уже актуально.
yokarpachev@dk3n33 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab04 $ git add .
yokarpachev@dk3n33 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab04 $ git commit -m "Add some files"
[master e1025c6] Add some files
3 files changed, 266 insertions(+)

```

рис.12 Добавление файлов

Отправка файлов на сервер с помощью git push (рис.13 [??]).

```

yokarpachev@dk3n33 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab04 $ git push
Перечисление объектов: 12, готово.
Подсчет объектов: 100% (12/12), готово.
При скатывании изменений используется до 4 потоков
Сканирование объектов: 100% (8/8), готово.
Запись объектов: 100% (8/8), 6.08 КБ | 3.04 МБ/с, готово.
Всего 8 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 3 local objects.
To github.com:YKarpachev/study_2023-2024_arh-pc.git
98dfc8a..e1025c6 master -> master
yokarpachev@dk3n33 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab04 $

```

рис.13 Отправка файлов

5 Выводы

Я освоил процедуры компиляции и сборки программ, написанных на ассемблере NASM.