# Лекция № 1

Т. Ф. Хирьянов

## История создания вычислительных машин

### Первые вычислительные машины

Первая вычислительная машина была создана Чарльзом Бэббиджем в начале 19 века. Она была механической и способствовала автоматизации вычислений путём аппроксимации функций многочленами и вычисления конечных разностей. Однако технические возможности того времени не позволили добиться большой мощности вычислений.

Следующим человеком, оставившим след в истории вычислительной техники, был Конрад Цузи. Его вычислительные машины (Z1, Z2, ... Z22) были релейными и работали в двоичной системе счисления.

Затем над созданием вычислительных машин уже трудились группы ученых. Среди самых известных – это Эниак (США), МЭСМ и БЭСМ (СССР). Которые уже являлись электронными вычислительными машинами.

### Поколения ЭВМ

Основным направлением совершенствования ЭВМ был поиск наиболее компактных и эффективных устройств, выполняющих преобразование входного сигнала.

1. *Ламповые ЭВМ*
2. *Транзисторные ЭВМ*
3. *ЭВМ на интегральных схемах*

   Интегральные схемы представляют собой элементы, на которых уже реализованны те или иные логические операции.

   **Пример.**

   Допустим, что на вход интегральной схеме подаются два сигнала. На выходе получается сигнал, с напряжением, увеличившимся на конкретное значение. Для логической операции «И» на выходе появится цифровой сигнал, только если на вход были одновременно были поданы 2 сигнала. Для логического «ИЛИ» будет достаточно появление всего 1 сигнала.

4. *Сверхбольшие интегральные схемы*

   Эти схемы представляют собой микропроцессор или микроконтроллер. Однако у последних имеются отличия. Так, микропроцессор работает на внешних командах, а микроконтроллер – на уже заложенной в него программе поведения. Иначе можно сказать, что микропроцессор работает по архитектуре фон Неймана, а микроконтроллер – по гарвардской архитектуре.

Рассмотрим теперь данные архитектуры на примере принципов построения одной из них.

# Принципы фон Неймана

1. *Принцип двоичности*

   Все вычисления выполняются в двоичной системе счисления.

2. *Принцип программного управления*

   Поведением компьютера управляет специальная программа.

3. *Принцип однородности памяти*

   Компьютеру необходимо иметь центральное процессорное устройство (CPU) и оперативное запоминающее устройство (RAM). Согласно фон Нейману, данные и команды следует хранить в одном месте, причем программы должны быть записаны в виде двоичных команд. Преимущество такой концепции состоит в том, что программа может быть загружена в оперативную память и начать выполняться. В гарвардской архитектуре существует память с запрещенной записью, тем самым отсутствует возможность загрузить вирусную программу.

4. *Принцип адресуемости памяти*

   Это означает, что у каждой ячейки памяти есть свой номер, по которому можно получить доступ к ней.

5. *Принцип последовательного программного управления*

   Команды, записанные в программе, выполняются последовательно. Впрочем линейные программы не всегда удобны. Например, чтобы не прописывать подряд повторяющиеся команды, разумнее организовать их в цикл. Однако такой подход может привести к зацикливанию программы.

6. *Принцип условного перехода*

   Для устранения этой трудности необходим условный переход, в котором условие и последующие команды обеспечивают выход из цикла.

   **Пример.**

   Так, например, пусть условие заключается в сравнении с переменной некоторого числа. Если переменная больше заданного числа, то программа выполняется по сценарию A. При этом переменная уменьшается на единицу и в какой-то момент станет меньше или равна заданной величины. Тогда выполнится сценарий B.

   Замечательно то, что циклов и условных переходов достаточно для реализации любого алгоритма.

# Устройство компьютера

## Организация команд

Рассмотрим центральное процессорное устройство (ЦПУ). Процессор представляет собой арифметико-логическое устройство, в котором реализованы сумматоры, мультипликаторы и другие логические операторы. Также ЦПУ содержит устройство управления (УУ), которое распознает язык машинных команд.

**Пример.**

Эти команды, как уже было сказано, двоичные (точнее шестнадцатиричные из-за схожести этих систем счисления и более компактной записи). Для удобства восприятия этот код переводится на язык ассемблера, в котором названия команд имеют понятный для человека вид.

Для работы процессору необходима специальная локализованная память, которая называется регистром. Регистры имеющие такие имена, как AX, BX, CX, DX, EX, являются арифметическими, т.е. в них записываются числа. Есть регистр IP (не имеющий ничего общего с сетью Интернет), в котором хранится адрес текущей исполняемой программы.

Доступ к этой части памяти осуществляется через специальный провод. При этом за один такт работы компьютера через него можно передать всего 1 бит информации. Поэтому для ускорения процесса используется 8 проводов, которые вместе называются шиной данных.

Также необходимо управлять процессами записи в память, считывания из нее, а также доступа к другим устройствам. Этот доступ осуществляется посредством обращения к микроконтроллеру. Для управлениями процессами считывания и записи данных необходима шина управления.

Для записи информации в память компьютера нужно передать сначала адрес ячейки памяти, а потом уже саму информацию. Чтобы сделать это одновременно используется дополнительная шина адресов.

Эти три шины вместе называют магистралью.

Далее допустим, что процессор занят вычислениями, например, раскладывает число на множители. В некоторый момент пользователь вызывает некую команду, например нажимает CTRL-C. Для того чтобы ее обработать устройство вызывает аппаратное прерывание посредством обращения к контроллеру прерываний.

Во время прерывания регистры процессора перезаписываются, т.к. вместо их старых значений необходимы данные для обработки другого процесса. Для того чтобы по окончании прерывания вернуть регистрам первоначальные значения, перед обработкой команд прерывания, значения регистров складываются в специальный сегмент памяти — стек.

Для этого существует специальный регистр stack-pointer (SP), который указывает на адрес этого участка. Первым записывается значение IP, содержащее адрес текущей программы: в противном случае будет невозможно вернуться к ее выполнению. Затем последовательно загружаются значения других регистров.

После этого процессору нужно определить, в каком участке памяти располагаются команды обработчика прерывания. Для этого он обращается к началу оперативной памяти, в которой размещена таблица прерываний, содержащей адреса всех таких обработчиков, и по номеру прерывания получает соответствующий адрес.

## Файловая система

Жесткий диск представляет собой стопку из нескольких дисков, каждый из которых разбит на кольца и сектора. При этом адрес ячейки памяти состоит из номера поверхности диска (head), номера кольца (cylinder) и сектора (sector). Однако для удобства программирования и хранения файлов данные должны быть удобно структурированы, т.е. должна существовать файловая система. Ее организацией и расшифровкой значения адреса по имени файла, например,

```
/home/student/1.txt
```

занимается операционная система. Более того возникла необходимость в создании разделов жесткого диска. Для доступа к ним в памяти есть специальная таблица разделов. Для работы с памятью такого виртуального диска, вызывается программное прерывание. При этом в регистры записывается задача: в AX – адрес начала имени файла, в BX – права доступа к этому файлу, и т.п. После этого вызывается стандартная процедура , которая и осуществляет работу с жестким диском. Эти

стандартные обработчики прерываний записываются в оперативную память при загрузке BIOS и операционной системы. Это позволяет программному обеспечению, вызывая команду INT 0x21, запустить операционную систему, которая определяет задачу по значениям регистров и выполняет ее.

### Назначение операционной системы

Изучив машинный язык или ассемблер для соответствующей архитектуры процессора можно написать ПО (software), оперирующее, например, непосредственно с жестким диском и ОЗУ (hardware). Однако это весьма трудоемкий процесс для программиста.

Операционная система занимает промежуточное место между программами и машинными командами. Таким образом, программа (application) может работать с регистрами и оперативной памятью. Команды же взаимодействующие с компьютером выполняются только ОС.

Начиная с 386 компьютера архитектуры Intel, появился защищенный режим работы компьютера. В нем процессор с помощью контроллера памяти определяет, обладает ли тот сегмент памяти, из которого берутся команды, привилегиями операционной системы. Если это так, программа может выполнять запрещенные для приложения ассемблерные команды и, ей доступна прямая физическая адресация памяти.

Приложение хранится в виртуальной памяти. Существует специальный контроллер, обычно встроенный в процессор, который содержит таблицу соответствий ее сегментов частям физической памяти компьютера.

**Пример.**

При записи приложением некоторого числа по адресу \$7531 контроллер памяти определяет, что данный адрес соответствует, например, $\$751B$.

Этой таблицей соответствий управляет операционная система, тем самым распределяя ресурсы компьютера.

### Виртуальная машина

Работая на одном компьютере можно использовать разные операционные системы. Для этого необходимо, чтобы в главной операционной системе в качестве приложения работала виртуальная машина – программа, эмулирующая работу компьютера и запускающая другую ОС.

### Компиляция и интерпретация в высокоуровневых и низкоуровневых языках программирования

Писать ассемблерный код достаточно трудоемко, поэтому этот язык используют только для оптимизации частей программы. Приложения пишут на более удобных языках программирования. После написания кода программы в итоге нужно получить бинарный исполняемый файл, который специфичен для каждой архитектуры. Этот перевод в некоторое количество этапов осуществляется специальными программами – компиляторами.

Чтобы отложить этот процесс и тем самым позволить программе запускаться на любой архитектуре, необходим интерпретатор. Python является одним из интерпретируемых языков программирования.

## 4 свободы программного обеспечения

Ричардом Столманом, основателем проекта GNU, были сформулированы следующие свободы программного обеспечения:

1. *Свобода использования*

   Это значит, что ПО может бесплатно использоваться кем угодно, когда, как и для чего угодно.

2. *Свобода видеть исходный код программы* (open source)

3. *Свобода копирования и распространения*

   Встречаются программы, включающие такую свободу, однако в них не предусмотрен доступ к исходному коду (shareware).

4. *Свобода соучастия в разработке*