**Государственное Бюджетное Профессиональное Образовательное Учреждение**

**Московской области**

**Воскресенский колледж**

Методические указания к самостоятельной работе

по учебной дисциплине ОП.03 «Архитектура аппаратных средств» основной образовательной программы по специальности СПО

09.02.07 «Информациоонные системы и программирование»

Воскресенск

Пояснительная записка

В методических указаниях рассмотрены научные предпосылки создания ЭВМ, а также технические предпосылки и практические потребности создания ЭВМ. Представлены поколения эволюции ЭВМ. Рассмотрены классы современных ЭВМ. Описано представление информации в вычислительных машинах. Рассмотрена функциональная и структурная организация ЭВМ, а именно структура и функционирование процессора, основные блоки ЭВМ, их назначение и функциональные характеристики.

Методическая разработка предназначена в первую очередь для студентов, обучающихся на очном отделении по специальности 09.02.07 «Программирование в компьютерных системах»

В результате изучения дисциплины студент должен знать:

* Базовые понятия и основные принципы построения архитектур вычислительных систем;
* Типы вычислительных систем и их архитектурные особенности;
* Организацию и принцип работы основных логических блоков компьютерных систем
* Алгебраическое представление двоичных чисел; прямой, обратный, дополнительный коды.
* Процессы обработки информации на всех уровнях компьютерных архитектур;
* Основные компоненты программного обеспечения компьютерных систем;
* Основные принципы управления ресурсами и организации доступа к этим ресурсам

В результате изучения дисциплины студент должен уметь:

* Получать информацию о параметрах компьютерной системы
* Составлять комбинации из логических элементов ЭВМ
* Подключать дополнительное оборудование и настраивать связь между элементами компьютерной системы;
* Производить инсталляцию и настройку программного обеспечения компьютерных систем

ЛЕКЦИИ ПО ДИСЦИПЛИНЕ «АРХИТЕКТУРА ЭВМ и систем»

# Лекция 1. Многоуровневая компьютерная организация

1. Языки, уровни и виртуальные машины

Человек пишет команды, которые в совокупности формируют язык, называемый условно Я1. Встроенные машинные команды тоже формируют язык, и мы будем называть его Я 0. Компьютер может выполнять только программы, написанные на его машинном языке Я 0. Компьютер не будет выполнять программы, на- писанные на языке Я 1, — ведь в конечном итоге компьютеру доступен только машинный язык Я 0.

Первый способ выполнения программы, написанной на языке Я 1, подразумевает замену каждой команды эквивалентным набором команд на языке Я 0. В этом случае компьютер выполняет новую программу, написанную на языке Я 0, вместо старой программы, написанной на Я 1. Эта технология называется **трансляцией**.

Второй способ означает создание программы на языке Я 0, получающей в качестве входных данных программы, написанные на языке Я 1. При этом каждая команда языка Я 1 обрабатывается поочередно, после чего сразу выполняется эквивалентный ей набор команд языка Я 0. Эта технология не требует составления новой программы на Я 0. Она называется **интерпретацией**, а программа, которая осуществляет интерпретацию, называется **интерпретатором**.

Между трансляцией и интерпретацией много общего. В обоих подходах компьютер в конечном итоге выполняет набор команд на языке Я 0, эквивалентных ко- мандам Я 1. Различие лишь в том, что при трансляции вся программа Я 1 переделывается в программу Я 0, программа Я 1 отбрасывается, а новая программа на Я 0 загружается в память компьютера и затем выполняется. При интерпретации каждая команда программы на Я 1 перекодируется в Я 0 и сразу же выполняется. В отличие от трансляции, здесь не создается новая программа на Я 0, а происходит последовательная перекодировка и выполнение команд.

Впрочем, чем мыслить категориями трансляции и интерпретации, гораздо проще представить себе существование гипотетического компьютера или **виртуальной машины**, для которой машинным языком является язык Я 1. Другими словами, можно писать программы для виртуальных машин так, как будто эти машины реально существуют.

Изобретение целого ряда языков, каждый из которых более удобен для человека, чем предыдущий, может продолжаться до тех пор, пока мы не дойдем до подходящего нам языка. Каждый такой язык использует своего предшественника как основу, поэтому мы можем рассматривать компьютер в виде ряда уровней, изображенных на рисунке 1. Язык, находящийся в самом низу иерархической структуры, — самый примитивный, а тот, что расположен на ее вершине — самый сложный.



Большинство современных компьютеров состоит из двух и более уровней.

Существуют компьютеры с шестью уровнями (рисунок 2). Уровень 0 — это аппаратное обеспечение машины. Электронные схемы на уровне 1 выполняют машинно-зависимые программы. Ради полноты нужно упомянуть о существовании еще одного уровня, который расположен ниже нулевого. Этот уровень не показан на рисунке 2, так как он попадает в сферу электронной техники и называется уровнем *физических устройств*.

Уровень 2 мы будем называть уровнем **архитектуры набора команд**. Каждый производитель публикует руководство для компьютеров, которые он продает, под названием «Руководство по машинному языку X», «Принципы работы компьютера У» и т. п. Подобное руководство содержит информацию именно об этом уровне. Описываемый в нем набор машинных команд в действительности выполняется микропрограммой-интерпретатором или аппаратным обеспечением.

Если производитель поставляет два интерпретатора для одной машины, он должен издать два руководства по машинному языку, отдельно для каждого интерпретатора.

Следующий уровень обычно является **гибридным**. Большинство команд в его языке есть также и на уровне архитектуры набора команд (команды, имеющиеся на одном из уровней, вполне могут быть представлены и на других уровнях).

У этого уровня есть некоторые дополнительные особенности: новый набор команд, другая организация памяти, способность выполнять две и более программы одновременно и некоторые другие. При построении уровня 3 возможно больше вариантов, чем при построении уровней 1 и 2.

Новые средства, появившиеся на уровне 3, выполняются интерпретатором, который работает на втором уровне. Этот интерпретатор был когда-то назван *операционной системой*. Команды уровня 3, идентичные командам уровня 2, выполняются микропрограммой или аппаратным обеспечением, но не операционной системой. Другими словами, одна часть команд уровня 3 интерпретируется операционной системой, а другая часть — микропрограммой. Вот почему этот уровень считается гибридным.

Между уровнями 3 и 4 есть существенная разница. Нижние три уровня задуманы не для того, чтобы с ними работал обычный программист. Они изначально ориентированы на интерпретаторы и трансляторы, поддерживающие более высокие уровни. Эти трансляторы и интерпретаторы составляются так называемыми системными программистами, которые специализируются на разработке новых виртуальных машин. Уровни с четвертого и выше предназначены для прикладных программистов, решающих конкретные задачи.

Еще одно изменение, появившееся на уровне 4, — механизм поддержки более высоких уровней. Уровни 2 и 3 обычно интерпретируются, а уровни 4, 5 и выше обычно, хотя и не всегда, транслируются.

Другое различие между уровнями 1, 2, 3 и уровнями 4, 5 и выше — особенность языка. Машинные языки уровней 1, 2 и 3 — *цифровые*. Программы, написанные на этих языках, состоят из длинных рядов цифр, которые воспринимаются компьютерами, но малопонятны для людей.

Начиная с уровня 4, языки содержат слова и сокращения, понятные человеку. Уровень 4 представляет собой символическую форму одного из языков более низкого уровня. На этом уровне можно писать программы в приемлемой для человека форме. Эти программы сначала транслируются на язык уровня 1, 2 или 3, а затем интерпретируются соответствующей виртуальной или фактически существующей машиной. Программа, которая выполняет трансляцию, называется **ассемблером**.

Уровень 5 обычно состоит из языков, разработанных для прикладных программистов. Такие языки называются языками высокого уровня. Существуют сотни языков высокого уровня. Наиболее известные среди них — *С*, *C++, Java*, *LISP* и *Prolog*. Программы, написанные на этих языках, обычно транслируются на уровень 3 или 4. Трансляторы, которые обрабатывают эти программы, называются **компиляторами**. Иногда также имеет место интерпретация.

В некоторых случаях уровень 5 состоит из интерпретатора для конкретной прикладной области, например символической логики. Он предусматривает данные и операции для решения задач в этой области, выраженные при помощи специальной терминологии.

Таким образом, компьютер проектируется как иерархическая структура уровней, которые надстраиваются друг над другом. Каждый уровень представляет собой определенную абстракцию различных объектов и операций.

# Лекция 2. Архитектура компьютера

1. Общее представление архитектуры компьютера

**Архитектура компьютера** представляет собой совокупность средств, приемов, правил, абстракций и характеристик, которые порождают конкретную реализацию и которые можно использовать как классификационные признаки отличия вычислительных систем.

К *средствам* можно отнести систему из центральной части (одного или нескольких процессоров, выполняющих преобразование информации, и оперативной памяти), средств связи (каналов) и периферийных устройств (устройств ввода-вывода, отображения и запоминания информации).

Примером *приемов* может служить применение специальных таблиц для определения адреса расположения информации в памяти. В простом случае таблицы содержат имена данных и адреса их расположения в памяти. В более сложном случае таблицы содержат служебную информацию (дескрипторы), с помощью которой устанавливаются необходимые адреса памяти.

К *правилам* относятся алгоритмы взаимодействия процессора с терминальными устройствами, алгоритмы выполнения команд, методы вычисления физических адресов памяти и др.

Под *абстракциями* понимаются уровни архитектуры, определяющие принципы организации вычислительной системы и функции процессора. В качестве примера можно выделить следующие уровни абстракции архитектуры вычислительной системы:

* архитектура системы - разделяет функции по обработке данных, выполняемые системой и внешней средой (пользователями), при этом связь системы с внешней средой производится через два набора интерфейсов - языки программирования и системные программы;
* архитектура аппаратно-программной границы - устанавливает границу между аппаратным обеспечением (электронными логическими схемами и микропрограммами) и операционной системой;
* архитектура микропрограммного управления - задает границу выполнения операций с помощью логических электронных схем и с помощью микропрограмм (программ, записанных в постоянной памяти);
* архитектура программного обеспечения - устанавливает уровни разделения используемых языков, модулей и приемы их построения;
* архитектура процессора - устанавливает организацию процессора и интерфейса между процессором и каналами связи;
* архитектура каналов связи - определяет взаимодействие процессоров с периферийными устройствами;
* мультипроцессорная архитектура - устанавливает разделение функций между процессорами и их взаимосвязь.

***Аппаратная архитектура*** может быть разделена на структурную, схемотехническую и конструкторскую архитектуру. Каждое из этих направлений имеет свою отличительную совокупность показателей. *Структурные* показатели позволяют установить наличие функциональных модулей (блоков) и их взаимосвязь в компьютере. *Схемотехнические* показатели могут касаться характеристик используемого микропроцессорного комплекса. *Конструкторские* показатели связаны с вы явлением особенностей принятых конструкторских решений. Последние могут оказаться решающими для персональных компьютеров (ПК) при реализации требования расширения их возможностей, т.е. построения ПК с открытой архитектурой.

***Программная архитектура*** рассматривает архитектурные показатели компьютера с точки зрения программиста. При этом рассматриваются структура данных, система сигналов и команд, способы адресации, программнодоступные средства, методы управления и основные режимы работы.

Архитектура и характеристики аппаратных средств определяют характеристики всего компьютера как программно-аппаратного комплекса обработки информации. Рассмотрим коротко историю развития архитектуры аппаратных средств и их классификацию.

Наиболее обобщенный способ классификации архитектур аппаратных средств компьютера базируется на понятиях потока команд и потока данных в вычислительной структуре. Соответственно этому подходу можно определить четыре класса структур аппаратных средств компьютеров (рисунок 3).

1. *S1SD - архитектура с одинарным потоком команд и одинарным потоком данных.* Управления осуществляет одинарная последовательность команд, любая из которых обеспечивает выполнения одной операции со своими данными и дальше передает управления следующей команде. В компьютерах этого типа команды выполняются только последовательно во времени на одном процессорном элементе.
2. *MISD - архитектура с множественным потоком команд и одинарным потоком данных,* которая получила также название конвейера обработки данных. Она составляет цепочку последовательно соединенных процессоров (микропроцессоров), которые управляются параллельным потоком команд. На вход конвейера из памяти подается одинарный поток данных, которые проходят последовательно через все процессоры, любой из которых делает обработку данных под управлением своего потока команд и передает ре- зультаты следующему по цепочке процессору, который использует их как входные данные.
3. *SIMD - архитектура с одинарным потоком команд и множественным потоком данных.* Процессор таких машин имеет матричную структуру, в узлах которой включенное большое количество сравнительно простых быстродействующих процессорных элементов, которые могут иметь собственную или общую память данных. Одинарный поток команд вырабатывает одно общее устройство управления. При этом все процессорные элементы выполняют одновременно одну и ту же команду, но над разными операндами, которые доставляются из памяти множественным потоком.
4. *MIMD - архитектура с множественными потоками команд и данных.* К таким структурам относятся многопроцессорные и многомашинные вычислительные системы. Они могут отличаться принципом управления (централизованное или распределенное), организацией памяти (общей, распределенная или комбинированная) и структурой связей между компьютерами или процессорами. Гибкость *MIMD* структур разрешает организовать совместную работу компьютеров, которые входят в них, или процессоров за распараллеленной программой при решении одной сложной задачи, или раздельную работу всех компьютеров при одновременном решении великого множества задач с помощью независимых программ. Низшее приводится более подробное описание некоторых более всего распространенных структур компьютеров.



Рисунок 3 – *а -* SISD (однопроцессорная), *б -* MISD (конвейерная);

*в* - SIMD (векторная); *г* - MIMD (матричная)

*Основные* функции определяют назначение компьютера: обработка и хранение информации, обмен информацией с внешними объектами. *Дополнительные* функции повышают эффективность выполнения основных функций: обеспечивают эффективные режимы ее работы, диалог с пользователем, высокую надежность и др. Названные функции компьютера реализуются с помощью его компонентов: аппаратных и программных средств.

**Элементы конструкции ПК.** Конструктивно ПК выполнены в виде центрального системного блока, к которому через разъемы подключаются внешние устройства: дополнительные устройства памяти, клавиатура, дисплей, принтер, мышь и др.

**Системный блок** обычно включает в себя системную плату, блок питания, накопители на дисках, разъемы для дополнительных устройств и *платы расширения* с контроллерами - адаптерами внешних устройств.

**Структура компьютера** - это некоторая модель, устанавливающая состав, порядок и принципы взаимодействия входящих в нее компонентов.

2. Типовые архитектуры ПК

* 1. Архитектура неймановского компьютера

Исторически неймановская архитектура компьютера является первой классической вычислительной архитектурой. Основные принципы организации неймановских компьютеров:

* использования двоичной системы исчисления для представления информации;
* программы и данные записываются в двоичном коде с использованием форматов одинаковой длины, которая разрешает сохранять их в общих запоминающих устройствах и над командами программы делать операции как над числами;
* управления вычислительным процессом осуществляется централизованно путем последовательного выполнения команд. Каждая команда руководит выполнением одной операции и передает управления следующей команде;
* память компьютера имеет линейную организацию, то есть она состоит из последовательно пронумерованных ячеек;
* для управления используется язык машинных команд, которым отвечают элементарные операции, которые последовательно выполняются над элементарными операндами.

Перечисленные принципы обеспечивают построение алгоритмически универсальных компьютеров простой архитектуры (рисунок 4) с минимальными аппаратурными затратами. Такую архитектуру имели компьютеры первых генераций.



Рисунок 4 – Архитектура неймановского компьютера

* 1. Архитектура постнеймановских компьютеров

В настоящее время различают архитектуру больших универсальных компьютеров-мейнфреймов (наиболее типичными представителями являются компьютеры серий IBM 360/370 и их «потомков» ES9000) и архитектуру мини-, микро- и персональных компьютеров.

Особенностью универсальных компьютеров является *параллельная и асинхронная работа процессора и специализированных процессоров ввода-вывода, каналов ввода-вывода.* Каналы ввода-вывода полностью управляют всеми периферийными устройствами. Взаимодействие периферийных устройств с каналами и каналов с процессором обеспечивается системой прерывания. Если при выполнении программы возникает необходимость в работе периферийного устройства, то процессор инициализирует канал на выполнение данной операции, после чего продолжает выполнять основную программу. О завершении своей работы канал сообщает процессору прерыванием. Такая архитектура наиболее эффективная в понимании быстродействия, но требует больших аппаратных затрат (каналы ввода-вывода по своей архитектуре более сложные, чем процессор), сложного управления и имеют более низкую архитектурную надежность.

Универсальные компьютеры типа IBM 360/370 используются в режиме мультипрограммной обработки информации для многих пользователей и имеют широкий набор периферийных устройств. Типовая архитектура изображена на рисунке 5.

Процессор имеет арифметико-логическое устройство (АЛУ), устройство центрального управления (ЦУ), устройство управления памятью (УУП) и устройство контроля и диагностики (УКиД). *Арифметико-логическое устройство* (АЛУ) выполняет арифметические и логические операции над двоичными и двоично- десятичными числами. *Устройство центрального управления* (ЦУ) обеспечивает микропрограммное управление всего процессора, обработку прерываний и отсчет времени. *Устройство управления памятью* (УУП) обеспечивает связь процессора и каналов ввода-вывода с оперативным запоминающим устройством (ОЗУ), решения конфликтов при обращении к памяти и буферизацию информации, которая передается. *Устройство контроля и диагностики* (УКиД) обеспечивает текущий контроль функционирования компьютера при инициализации системы.

*Мультиплексный канал* является специализированным процессором ввода- вывода и обеспечивает ввод/вывод информации из медленнодействующих *периферийных устройств* (ПУ). Он работает в мультиплексном режиме, то есть после чтения/записи одного байта информации из одного периферийного устройства возможный обмен байтом информации с другим более приоритетным устройством (если канал получи запрос на обмен);

*Селекторные каналы* также являются специализированными процессорами ввода-вывода, но они предназначены для работы с периферийными быстродействующими устройствами, например, с устройствами внешней памяти, накопителями на дисках. Селекторный канал работает в селекторном режиме, то есть, если начался обмен информацией с одним устройством, то он не может быть прерван другим, даже более приоритетным, устройством.

Все периферийные устройства подключаются к каналу через свои *устройства управления* (УУ), что обеспечивают стандартное подключение разнотипных устройств к каналам.

Отличительной особенностью мини-, микро- и персональных компьютеров является простота и надежность в управлении. Поэтому все эти компьютеры имеют магистральную архитектуру, при которой процессор связан со всеми другими блоками компьютера (блоки ОЗУ, ПЗУ, периферийные устройства) путем единого интерфейса типа *общей шины* (рисунок 6)*.* При такой архитектуре в данный момент времени возможный обмен информацией только между двумя блоками, один из которых является *задатчиком* (обычно процессор или контроллер прямого доступа к памяти) и управляет процессом передачи информации по обшей шине, которая состоит из трех подшин - *шины адреса (А), шины данных (Д)* и *шины управления (У).* Общее управление системным интерфейсом выполняет контроллер шины.



Архитектура персональных компьютеров (ПК) берет начало от магистральной архитектуры, но в процессе своего развития архитектура ПК стала базироваться на *системы шин,* которая включает: *локальную шину процессора* (L-local bus), *шину оперативной памяти* (М - Memory bus), *системную шину* (S - System bus), которая связывает работу всех модулей компьютера в единое целое, и *внешнюю (периферийную) шину (X -* external bus), связанную с периферийными модулями.

# Лекция 3. Системный интерфейс и архитектура системной платы

* + 1. Материнская плата

*Материнская плата (Motherboard)* — основной компонент каждого ПК. Называется *главной (Mainboard),* или *системной,* платой. Это самостоятельный элемент, который управляет внутренними связями и взаимодействует с внешними устройствами. Материнская плата является основным элементом внутри ПК, влияющим на производительность компьютера в целом.

Современные платы исполняются на основе *чипсетов* (*Chipset*) - наборов из нескольких БИС, реализующих все необходимые функции связи основных компонентов - процессора, памяти и шин расширения. Чипсет определяет возможности применения различных типов процессоров, основной и кэш-памяти и ряд других характеристик системы, определяющих возможности ее модернизации. Его тип существенно влияет и на производительность - при одинаковых установленных компонентах производительность компьютеров, собранных на разных системных платах -читай, чипсетах, - может отличаться на 30%.

Современные чипсеты обеспечивают совместимость устанавливаемых на системную плату модулей и позволяют выполнять автоматическую идентификацию типов установленных компонент, а в некоторых случаях и быстродействие (на- пример ОЗУ).

На рисунке 8 представлена структура типовой материнской платы:



* процессор, установленный в специальный разъем и охлаждаемый радиатором с вентилятором;
* микросхемы кэш-памяти второго уровня (внешней). В современных процессорах эти микросхемы устанавливаются на плату картриджа центрального процес сора;
* слоты для установки модулей оперативной памяти;
* слоты для установки карт расширения. Как правило, на материнских платах имеются разъемы для карт стандарта ISA и PCI. Современные модели материн- ских плат оборудованы дополнительно слотом AGP. Наличие слотов и возмож- ность установки в них любых карт расширения (видеоадаптера, звуковой карты, модема, карты АЦП и других) определяет открытую архитектуру ПК;
* микросхема перепрограммируемой памяти, в которой хранятся программы BIOS, программы тестирования ПК, загрузки операционной системы, драйверы устройств, начальные установки;
* разъемы для подключения накопителей HDD, FDD**.**

Все компоненты материнской платы связаны между собой системой проводников (линий), по которым происходит обмен информацией. Эту совокупность линий называют информационной шиной, или просто шиной *(Bus).*

Взаимодействие между компонентами и устройствами ПК, подключенными к разным шинам, осуществляется с помощью так называемых **мостов**, реализованных на одной из микросхем *Chipset*. Например, на рисунке структуры типовой системной платы мост для соединения шины ISA и PCI реализован в микросхеме 82371АВ.

Размеры материнской платы, а также отверстия внутри платы, которые соединяют ее с дном корпуса, стандартизованы.

Поскольку быстродействие различных компонентов (процессора, памяти, адаптеров для шин ISA, EISA, VLB, PCI) существенно различается, в компьютерах на процессорах класса 486 и старше применяется деление опорной частоты для синхронизации шин ввода/вывода и внутреннее умножение частоты в процессорах. Различают следующие частоты:

* *Host Bus Clock -* частота системной шины (внешняя частота шины процессора). Эта частота является опорной для всех других;
* *CPU Clock,* или *Core Speed -* внутренняя частота процессора, на которой работает его вычислительное ядро. Современные технологии позволили существенно повысить предельные частоты интегральных компонентов, в связи с чем широко применяется внутреннее умножение частоты на 1,5, 2,2,5, 3,3,5, 4 и некоторые другие значения;
* *PCI Bus Clock -* частота шины PCI, которая должна составлять 25-33,3 МГц (спецификация PCI-2.1 допускает частоту до 66,6 МГц). Она обеспечивается делением Host Bus Clock на 2 (реже на 3);
* *VLB Bus Clock -* частота шины VLB, определяемая аналогично PCI Bus Clock. Платы с шиной VLB обычно имеют джампер, переключаемый в зависимости от того, превышает ли системная частота значение 33,3 МГц;
* *ISA Bus Clock,* или *A TCLK -* частота шины ISA, которая должна быть близка к 8 МГц. Она обычно задается в BIOS Setup через коэффициент деления системной частоты.
	+ 1. Система шин

*Шиной (Bus)* называется вся совокупность линий (проводников на материнской плате), по которым обмениваются информацией компоненты и устройства ПК. Шина предназначена для обмена информацией между двумя и более устройствами. Шина, связывающая только два устройства, называется **портом**. На рисунке 9 дана система шины.



Рисунок 9 – Система шин

Шина имеет места для подключения внешних устройств — слоты, которые в результате становятся частью шины и могут обмениваться информацией со всеми другими подключенными к ней устройствами.

Шины в ПК различаются по своему функциональному назначению:

* *системная шина* (или шина CPU) используется микросхемами *Cipsct* для пересылки данных к CPU и обратно;
* *шина кэш-памяти* предназначена для обмена данными между CPU и кэш- памятью;
* *шина памяти* используется для обмена данными между оперативной памятью RAM и CPU;
* *шины ввода/вывода* данных подразделяются на стандартные и локальные.

Локальная шина ввода/вывода — это скоростная шина, предназначенная для обмена данными между быстродействующими периферийными устройствами (видеоадаптерами, сетевыми картами, картами сканера и др.) и системной шиной под управлением CPU. В настоящее время в качестве локальной шины используется шина PCI. Для ускорения ввода/вывода видеоданных и повышения производительности ПК при обработке трехмерных изображении корпорацией Intel была разработана шина **AGP** *(Accelerated Graphics Port}.*

Стандартная тина ввода/вывода используется для подключения к перечисленным выше шинам более медленных устройств (например, мыши, клавиатуры, модемов, старых звуковых карт). До недавнего времени в качестве этой шины использовалась шина стандарта **ISA**. В настоящее время — шина **USB**.

Шина имеет собственную архитектуру, позволяющую реализовать важнейшие се свойства — возможность параллельного подключения практически неограниченного числа внешних устройств и обеспечение обмена данными между ними. Архитектура любой шины имеет следующие компоненты:

* линии для обмена данными (шина данных);
* линии для адресации данных (шина адреса);
* линии управления данными (шина управления);
* контроллер шины.

**Контроллер шины** осуществляет управление процессом обмена данными и служебными сигналами и обычно выполняется в виде отдельной микросхемы либо в виде совместимого набора микросхем — Chipset.

**Шина данных** обеспечивает обмен данными между *CPU*, картами расширения, установленными в слоты, и памятью *RAM*. Чем выше разрядность шины, тем больше данных может быть передано за один такт и тем выше производительность ПК**.** Компьютеры с процессором 80286 имели 16-разрядную шину данных, с CPU 80386 и 80486 — 32-разрядную, а компьютеры с CPU семейства Pentium — 64-разрядную шину данных.

**Шина адреса** служит для указания адреса к какому-либо устройству ПК, с которым CPU производит обмен данными. Каждый компонент ПК, каждый регистр ввода/вывода и ячейка RAM (оперативная память) имеют свой адрес и входят в общее адресное пространство PC. По шине адреса передается идентификационный код (адрес) отправителя и (или) получателя данных.

**Шина управления** передает ряд служебных сигналов: записи/считывания, готовности к приему/передаче данных, подтверждения приема данных, аппаратного прерывания, управления и других, чтобы обеспечить передачу данных.

* + 1. Центральный процессор

**Центральный процессор** – это устройство компьютера, которое выполняет операции по обработке данных и управляет периферийными устройствами.

Современные процессоры выполняются в виде микропроцессоров.

Физически **микропроцессор** представляет собой **интегральную схему** — тонкую пластинку кристаллического кремния прямоугольной формы площадью всего несколько квадратных миллиметров, на которой размещены схемы, реализующие все функции процессора. Кристалл-пластинка обычно помещается в пластмассовый или керамический плоский корпус и соединяется золотыми проводками с металлическими штырьками, чтобы его можно было присоединить к системной плате компьютера.

Подробнее рисунок 11:

из основной памяти в основную память

БУР

АЛУ

БРП

БСОП

синхронное напряжение

прерывания

УУ

АЛУ – арифметико-логическое устройство

УУ – устройство управления

БУР – блок управления регистров

БСОП – блок связи с основной памятью

БРП – блок регистровой памяти

ОП – основная память (ПЗУ или ОЗУ, L2 кэш)

Рисунок 11 – Структура ЦП

* *устройство управления* (УУ) - формирует и подает во все блоки компьютера в нужные моменты времени определенные сигналы управления (управляющие импульсы), обусловленные спецификой выполняемой операции и результатами предыдущих операций; формирует адреса ячеек памяти, используемых выполняемой операцией, и передает эти адреса в соответствующие блоки компьютера; опорную последовательность импульсов устройство управления получает от генератора тактовых импульсов;
* *арифметико-логическое устройство* **(АЛУ)** - предназначено для выполнения всех арифметических и логических операций над числовой и символьной информацией;
* БУР предназначен для временного хранения обрабатываемой информации и представляет собой местную сверхбыстродействующую память (СОЗУ)*. Микропроцессорная память* **(МПП)** - служит для кратковременного хранения, записи и выдачи информации, непосредственно используемой в вычислениях в ближайшие такты работы компьютера. МПП строится на регистрах и используется для обеспечения высокого быстродействия компьютера, ибо основная память (ОП) не всегда обеспечивает скорость записи, поиска и считывания информации, необходимую для эффективной работы быстродействующего микропроцессора;
* *интерфейсная система микропроцессора* (БСОП) - реализует сопряжение и связь с другими устройствами ПК; включает в себя внутренний интерфейс МП, буферные запоминающие регистры и схемы управления портами ввода- вывода (ПВВ) и системной шиной. *Интерфейс* (interface) - система средств сопряжения и связи устройств компьютера, обеспечивающая их эффективное взаимодействие. *Порт ввода-вывода (I/O ≈ Input/Output port)* - аппаратура сопряжения, позволяющая подключить к микропроцессору другое устройство ПК.
* **Генератор тактовых импульсов**. Он генерирует последовательность электрических импульсов; частота генерируемых импульсов определяет тактовую частоту машины. Промежуток времени между соседними импульсами определяет время одного такта работы машины или просто *такт работы компьютера*.

К основным характеристикам процессора относятся:

*Быстродействие* (вычислительная мощность) – это среднее число операций процессора в секунду.

*Тактовая частота* в МГц. Тактовая частота равна количеству тактов в секунду. Такт - это промежуток времени между началом подачи текущего импульса ГТЧ и началом подачи следующего. Тактовая частота отражает уровень промышленной технологии, по которой изготавливался данный процессор. Она также характеризирует и компьютер, поэтому по названию модели микропроцессора можно составить достаточно полное представление о том, к какому классу принадлежит компьютер. Поэтому часто компьютерам дают имена

микропроцессоров, входящих в их состав. Частота генератора тактовых импульсов является одной из основных характеристик компьютера и во многом определяет скорость его работы, ибо каждая операция в компьютере выполняется за определенное количество тактов

*Разрядность процессора* - это максимальное количество бит информации, которые могут обрабатываться и передаваться процессором одновременно. Разрядность процессора определяется разрядностью регистров, в которые помещаются обрабатываемые данные.

Термин «**ядро микропроцессора**» (*processor core*) не имеет чёткого определения и в зависимости от контекста употребления может обозначать:

* часть микропроцессора, содержащую основные функциональные блоки.
* набор параметров, характеризующий микропроцессор.
* кристалл микропроцессора, чаще всего, открытый.
* часть процессора, осуществляющая выполнение одного потока команд.

Многоядерные процессоры имеют несколько ядер и поэтому способны осуществлять независимое параллельное выполнение нескольких потоков команд одновременно.

**Ядро** - это как бы версия (вариант) процессора. Процессоры с разными ядрами, это можно сказать разные процессоры. Разные ядра отличаются по размеру кэш-памяти, частоте шины, технологии изготовления и т. п. Чем новее ядро, тем лучше процессор разгоняется. В качестве примера можно привести Pentium 4, который имеет (на данный момент) два ядра Willamette и Northwood. Первое ядро производилось по 0.18 мкм технологии и работало исключительно на 400Mhz шине. Самые младшие модели имели частоту 1.3Ghz, максимальные частоты для ядра находились немного выше 2Ghz. Своими разгонными качествами эти процессоры особо не славились. Позже был выпущен Northwood. Он уже был выполнен по 0.13мкм технологии и поддерживал шину в 400 и 533Mhz, а также имел увеличенный объём кэш-памяти. Переход на новое ядро позволил значительно увеличить производительность и максимальную частоту. Младшие процессоры Northwood с частотой 1.6Ghz прекрасно разгоняются. Из данного примера можно делать для себя вывод, что это разные процессоры.

В рамках одной и той же архитектуры различные процессоры могут достаточно сильно отличаться друг от друга. И различия эти воплощаются в разнообразных процессорных ядрах, обладающих определенным набором строго обусловленных характеристик. Чаще всего эти отличия воплощаются в различных частотах системной шины, размерах кэша второго уровня, поддержке тех или иных новых систем команд или технологических процессах, по которым изготавливаются процессоры. Нередко смена ядра в одном и том же семействе процессоров влечет за собой замену процессорного разъема, из чего вытекают вопросы дальнейшей совместимости материнских плат. Однако в процессе совершенствования ядра, производителям приходится вносить в него незначительные изменения. Такие изменения называются **ревизиями ядра** и, чаще всего, обозначаются цифробуквенными комбинациями. Однако в новых ревизиях одного и того же ядра могут встречаться достаточно заметные нововведения. Так, компания Intel ввела поддержку 64-битной архитектуры EM64T в отдельные процессоры семейства Pentium 4 именно в процессе изменения ревизии.

# Лекция 4. Память

**Основная память (ОП).** Она предназначена для хранения и оперативного обмена информацией с прочими блоками компьютера. ОП содержит два вида запоминающих устройств: постоянное запоминающее устройство (ПЗУ) и оперативное запоминающее устройство (ОЗУ).

* + - 1. Оперативная память

**Оперативная память,** или *оперативное запоминающее устройство (ОЗУ),* предназначено для *приема, хранения* и *выдачи* информации и представляет собой самую быстродействующую запоминающую систему компьютера. Оперативная намять обозначается RAM *(Random Access Memory—* намять с Произвольным доступом). Процессор имеет возможность выполнять программы только после того, как они загружены в оперативную рабочую память, т.е. и память, доступную для программ пользователя. CPU имеет непосредственный доступ к данным, находящимся в оперативной памяти, а к внешней памяти (на гибких или жестких дисках) - через буфер, являющийся также разновидностью оперативной памяти. Работа программ, загруженных с внешнего носителя, возможна только после того, как она будет скопирована в RAM.

Однако оперативная намять имеет существенный недостаток, заключающийся в том, что она временная, т.е. при отключении питания оперативная память полностью очищается. При этом данные, не записанные на внешний носитель, будут утеряны. Основная задача RAM — предоставлять необходимую информацию к виде двоичных колов по запросам CPU, т.е. данные в любой момент должны быть доступны для обработки.

Существует два типа ОЗУ: статическое и динамическое. Статическое ОЗУ (Static RAM, SRAM) конструируется с использованием D-триггеров. Информация в ОЗУ сохраняется на протяжении всего времени, пока к нему подается питание: секунды, минуты, часы и даже дни. Статическое ОЗУ работает очень быстро. Обычно время доступа составляет несколько наносекунд. По этой причине статическое ОЗУ часто используется в качестве кэш-памяти второго уровня.

В динамическом ОЗУ (Dynamic RAM, DRAM), напротив, триггеры не используются. Динамическое ОЗУ представляет собой массив ячеек, каждая из которых содержит транзистор и крошечный конденсатор. Конденсаторы могут быть заряженными и разряженными, что позволяет хранить нули и единицы. Поскольку электрический заряд имеет тенденцию исчезать, каждый бит в динамическом ОЗУ должен обновляться (перезаряжаться) каждые несколько миллисекунд, чтобы предотвратить утечку данных. Поскольку об обновлении должна заботиться внешняя логика, динамическое ОЗУ требует более сложного сопряжения, чем статическое, хотя этот недостаток компенсируется большим объемом.

Поскольку динамическому ОЗУ нужен только 1 транзистор и 1 конденсатор на бит (статическому ОЗУ требуется в лучшем случае 6 транзисторов на бит), динамическое ОЗУ имеет очень высокую плотность записи (много битов на одну микросхему). По этой причине основная память почти всегда строится на основе динамических ОЗУ. Однако динамические ОЗУ работают очень медленно (время доступа занимает десятки наносекунд). Таким образом, сочетание кэш-памяти на основе статического ОЗУ и основной памяти на основе динамического ОЗУ соединяет в себе преимущества обоих устройств.

Конструктивно оперативная память выполняется и виде модулей микросхем, что позволяет дополнять объем оперативной памяти, которая используется не только в ПК, но и в самых разных периферийных устройствах — от видеокарт до лазерных принтеров. Микросхемы оперативной памяти и этом случае могут принадлежать к разным модификациям, но все они относятся к типу динамической оперативной памяти (DRAM).

**Синхронные динамические ОЗУ** (Synchronous DRAM, **SDRAM**). Синхронное динамическое ОЗУ управляется одним синхронизирующим сигналом. Данное устройство представляет собой гибрид статического и динамического ОЗУ. Основное преимущество синхронного динамического ОЗУ состоит в том, что оно исключает зависимость микросхемы памяти от управляющих сигналов. ЦП сообщает памяти, сколько циклов следует выполнить, а затем запускает эти циклы. Каждый цикл на выходе дает 4, 8 или 16 бит в зависимости от количества выходных строк. Устранение зависимости от управляющих сигналов приводит к увеличению скорости передачи данных между ЦП и памятью.

Следующим этапом в развитии памяти SDRAM стала память **DDR** (Double Data Rate - передача данных с двойной скоростью).

**DDR SDRAM** (от англ. Double Data Rate Synchronous Dynamic Random Access Memory - синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных) - тип компьютерной памяти, используемой в вычислительной технике в качестве оперативной и видеопамяти. При использовании DDR SDRAM достигается удвоенная скорость работы, нежели в SDRAM, за счёт считывания команд и данных не только по фронту, как в SDRAM, но и по спаду тактового сигнала. За счёт этого удваивается скорость передачи данных без увеличения частоты тактового сигнала шины памяти. Таким образом, при работе DDR на частоте 100 МГц мы получим эффективную частоту 200МГц (при сравнении с аналогом SDR SDRAM).

* + - 1. Постоянная память

**ПЗУ (постоянное запоминающее устройство), или ROM (Read-Only Memory — постоянная память)***. ПЗУ* служит для хранения неизменяемой (постоянной) программной и справочной информации, позволяет оперативно только считывать хранящуюся в нем информацию. Содержание памяти специальным образом «зашивается» в устройстве при его изготовлении для постоянного хранения. Прежде всего в постоянную память записывают программу управления работой самого процессора. В ПЗУ находятся программы управления дисплеем, клавиатурой, принтером, внешней памятью, программы запуска и остановки компьютера, тестирования устройств. Данные записываются в ПЗУ в процессе производства. Для этого изготавливается трафарет с определенным набором битов, который накладывается на фоточувствительный материал, а затем открытые (или закрытые) части поверхности вытравливаются. Единственный способ изменить программу в ПЗУ — поменять всю микросхему.

Однако, чтобы компаниям было проще разрабатывать новые устройства, основанные на ПЗУ, были выпущены **программируемые ПЗ**У (Programmable **ROM**, **PROM**). В отличие от обычных ПЗУ, их можно программировать в условиях эксплуатации, что позволяет сократить время выполнения заказа.

Следующая разработка этой линии — **стираемое программируемое ПЗУ** (Erasable PROM, **EPROM**), которое можно программировать в условиях эксплуатации, а также стирать с него информацию. Если кварцевое окно в данном ПЗУ подвергать воздействию сильного ультрафиолетового света в течение 15 минут, все биты установятся в 1. Если нужно сделать много изменений во время одного этапа проектирования, стираемые ПЗУ гораздо экономичнее, чем обычные программируемые ПЗУ, поскольку их можно использовать многократно.

Следующий этап — **электронно-перепрограммируемое ПЗУ** (Electronically EPROM, EEPROM), с которого можно стирать информацию, не подвергая воздействию ультрафиолетовых лучей. Кроме того, чтобы перепрограммировать данное устройство, его не нужно вставлять в специальный аппарат для программирования, в отличие от стираемого программируемого ПЗУ.

Более современный тип электронно-перепрограммируемого ПЗУ — **флэш- память**. **Перепрограммируемая постоянная память** (**Flash Memory**) — энергонезависимая память. Важнейшая микросхема Flash-памяти — модуль **BIOS**. Роль BIOS двоякая: с одной стороны это неотъемлемый элемент аппаратуры, а с другой стороны — важный модуль любой операционной системы. BIOS (Basic Input/Output System — базовая система ввода-вывода) — совокупность программ,

предназначенных для автоматического тестирования устройств после включения питания компьютера и загрузки операционной системы в оперативную память.

Наиболее широко среди пользователей компьютеров известна BIOS материнской платы, но BIOS присутствуют почти у всех компонентов компьютера: у видеоадаптеров, сетевых адаптеров, модемов, дисковых контроллеров, принтеров. По своей сути BIOS является посредником между аппаратным и программным обеспечением компьютера.

Главная функция BIOS материнской платы — инициализация устройств, подключённых к материнской плате, сразу после включения питания компьютера. BIOS проверяет работоспособность устройств (т. н. самотестирование, англ. POST — Power-On Self Test), задаёт низкоуровневые параметры их работы (например, частоту шины центрального микропроцессора), и после этого ищет загрузчик операционной системы (англ. Boot Loader) на доступных носителях информации и передаёт управление операционной системе. Операционная система по ходу работы может изменять большинство настроек, изначально заданных в BIOS. В некоторых реализациях BIOS позволяет производить загрузку операционной системы через интерфейсы, изначально для этого не предназначенные, в том числе USB и IEEE 1394. Также возможна загрузка по сети (применяется, например, в т. н. «тон- ких клиентах»).

Также BIOS содержит минимальный набор сервисных функций (например, для вывода сообщений на экран или приёма символов с клавиатуры), что и обусловливает расшифровку её названия: *Базовая система ввода-вывода*.

В некоторых BIOS’ах реализуется дополнительная функциональность (например, воспроизведение аудио-CD или DVD-дисков), поддержка встроенной рабочей среды (например, интерпретатор языка Basic) и др.

В отличие от стираемого ПЗУ, которое стирается под воздействием ультрафиолетовых лучей, и от электронно-перепрограммируемого ПЗУ, которое стирается по байтам, флэш-память стирается и записывается блоками.

* + - 1. Кэш-память

Процессоры всегда работали быстрее, чем память. Так как процессоры и память совершенствуются параллельно, это несоответствие сохраняется. На практике такое несоответствие в скорости работы приводит к тому, что, когда процессор обращается к памяти, проходит несколько машинных циклов, прежде чем он получит запрошенное слово. Чем медленнее работает память, тем дольше процессору приходится ждать, тем больше циклов проходит.

**КЭШ-память** – это буферная, не доступная для пользователя быстродействующая память, автоматически используемая компьютером для ускорения операций с информацией, хранящейся в более медленно действующих запоминающих

устройствах. Например, для ускорения операций с основной памятью организуется регистровая КЭШ-память внутри микропроцессора (КЭШ-память первого уровня) или вне микропроцессора на материнской плате (КЭШ-память второго уровня); для ускорения операций с дисковой памятью организуется КЭШ-память на ячейках электронной памяти.

Принцип его действия основан на том, что простой (не работа) более быстрого устройства сильно влияет на общую производительность (во-первых) и что скорее всего запрашиваются данные, сохраненные недавно (во-вторых). Поэтому между устройствами помещают буфер быстрой памяти (небольшой по сравнению со всеми хранимыми данными), что позволяет снизить потери быстрого устройства, как на записи, так и на чтении.

Основная идея кэш-памяти проста: в ней находятся слова, которые чаще всего используются. Если процессору нужно какое-нибудь слово, сначала он обращается к кэш-памяти. Только в том случае, если слова там нет, он обращается к основной памяти. Если значительная часть слов находится в кэш-памяти, среднее время доступа значительно сокращается. Таким образом, успех или неудача зависит от того, какая часть слов находится в кэш-памяти. Давно известно, что программы не обращаются к памяти наугад. Если программе нужен доступ к адресу *А*, то скорее всего после этого ей понадобится доступ к адресу, расположенному поблизости от *А*. Практически все команды обычной программы (за исключением команд перехода и вызова процедур) вызываются из последовательных областей памяти. Кроме того, большую часть времени программа тратит на циклы, когда ограниченный набор команд выполняется снова и снова. Точно так же при манипулировании матрицами программа скорее всего будет обращаться много раз к одной и той же матрице, прежде чем перейдет к чему-либо другому.

Ситуация, когда при последовательных обращениях к памяти в течение некоторого промежутка времени используется только небольшая ее область, называется **принципом локальности**. Этот принцип составляет основу всех систем кэш- памяти. Идея состоит в том, что, когда определенное слово вызывается из памяти, оно вместе с соседними словами переносится в кэш-память, что позволяет при очередном запросе быстро обращаться к следующим словам. Общее устройство процессора, кэш-памяти и основной памяти иллюстрирует рисунок 12.



Рисунок 12 - Связь процессора, кэш-памяти и основной памяти

Если слово считывается или записывается *k* раз, компьютеру требуется сделать 1 обращение к медленной основной памяти и *k - 1* обращений к быстрой кэш- памяти. Чем больше *k*, тем выше общая производительность.

Мы можем сделать и более строгие вычисления. Пусть *с* — время доступа к кэш-памяти, *m* — время доступа к основной памяти и *h* — коэффициент кэш- попаданий, который показывает соотношение числа обращений к кэш-памяти и общего числа всех обращений к памяти. В нашем примере *h = (k - 1)/k*.

Некоторые авторы выделяют **коэффициент кэш-промахов**, равный *1-h*. Таким образом, мы можем вычислить среднее время доступа:

*tдост = с + (1 - h)m*

Если *h → 1*, то есть все обращения делаются только к кэш-памяти, то время доступа стремится к *с*. С другой стороны, если *h → 0*, то есть каждый раз нужно обращаться к основной памяти, то время доступа стремится к *с + m*: сначала требуется время *с* для проверки кэш-памяти (в данном случае безуспешной), а затем - время *m* для обращения к основной памяти. В некоторых системах обращение к основной памяти может начинаться параллельно с использованием кэш-памяти, чтобы в случае кэш-промаха цикл обращения к основной памяти уже начался. Однако эта стратегия требует способности останавливать процесс обращения к основной памяти в случае кэш-попадания, что усложняет разработку подобного компьютера.

Основная память и кэш-память делятся на блоки фиксированного размера с учетом *принципа локальности*. Блоки внутри кэш-памяти обычно называют **строками кэша** (cache lines). При кэш-промахе из основной памяти в кэш-память загружается вся строка, а не только необходимое слово. Например, если строка состоит из 64 байт, обращение к адресу 260 влечет за собой загрузку в кэш-память всей строки (байты с 256 по 319) на случай, если через некоторое время понадобятся другие слова из этой строки. Такой путь обращения к памяти более эффективен, чем вызов каждого слова по отдельности, потому что однократный вызов *k* слов происходит гораздо быстрее, чем вызов одного слова *k* раз.

Кэш-память очень важна для высокопроизводительных процессоров. Однако здесь возникает ряд вопросов. *Первый вопрос* — **объем кэш-памяти**. Чем больше объем, тем лучше работает память, но тем дороже она стоит. *Второй вопрос* — **размер строки кэша**. Кэш-память объемом 16 Кбайт можно разделить на 1024 строки по 16 байт, 2048 строк по 8 байт и т. д. *Третий вопрос* — механизм организации кэш-памяти, то есть то, как она определяет, какие именно слова находятся в ней в данный момент. *Четвертый вопрос* — должны ли команды и данные находиться вместе в общей кэш-памяти. Проще всего разработать объединенную кэш-память (unified cache), в которой будут храниться и данные, и команды. В этом случае вызов команд и данных автоматически уравновешивается.

Однако в настоящее время существует тенденция к использованию разделенной кэш-памяти (split cache), когда команды хранятся в одной кэш-памяти, а данные — в другой. Такая архитектура также называется **гарвардской** (Harvard architecture), поскольку идея использования отдельной памяти для команд и отдельной памяти для данных впервые воплотилась в компьютере Маге III, который был создан Говардом Айкеном (Howard Aiken) в Гарварде. Современные разработчики пошли по этому пути, поскольку сейчас широко распространены конвейерные архитектуры, а при конвейерной организации должна быть возможность одновременного доступа и к командам, и к данным (операндам). Разделенная кэш- память позволяет осуществлять параллельный доступ, а общая — нет. К тому же, поскольку команды обычно не меняются во время выполнения программы, содержание кэша команд не приходится записывать обратно в основную память.

Наконец, *пятый вопрос* — **количество блоков кэш-памяти**. В настоящее время очень часто кэш-память первого уровня располагается прямо на микросхеме процессора, кэш-память второго уровня — не на самой микросхеме, но в корпусе процессора, а кэш-память третьего уровня — еще дальше от процессора.

Кэш-памятью управляет специальное устройство – контроллер, который, анализируя выполняемую программу, пытается предвидеть, какие данные и команды вероятнее всего понадобятся в ближайшее время процессору, и подкачивает их в кэш-память.



* + - 1. Внешняя память

Внешняя память относится к внешним устройствам ПК и используется для долговременного хранения любой информации, которая может когда-либо потре- боваться для решения задач. В частности, во внешней памяти хранится все про- граммное обеспечение компьютера. Внешняя память содержит разнообразные ви- ды запоминающих устройств, но наиболее распространенными, имеющимися практически на любом компьютере, являются накопители на жестких (НЖМД).

# Магнитные диски

Магнитный диск состоит из одной или нескольких алюминиевых поверхностей, в настоящее время компания IBM делает их из стекла, покрытых магнитным слоем. Изначально их диаметр составлял 50 см, сейчас — от 3 до 12 см, у портативных компьютеров — меньше 3 см, причем это значение продолжает уменьшаться. Головка диска, содержащая индукционную катушку, двигается над поверхностью диска, опираясь на воздушную подушку. Когда головка проходит над намагниченной областью, в ней (в головке) возникает положительный или отрицательный ток, что дает возможность считывать записанные ранее биты. Поскольку диск вращается под головкой, поток битов может записываться, а потом считываться. Конфигурация дорожки диска показана на рисунке 13.



**Дорожкой** называется круговая последовательность битов, записанных на диск за его полный оборот. Каждая дорожка делится на секторы фиксированной длины. Каждый сектор обычно содержит 512 байт данных. Перед данными располагается преамбула (preamble), которая позволяет головке синхронизироваться перед чтением или записью. После данных идет код исправления ошибок (Error-Correcting Code, ECC), позволяющий исправлять множественные ошибки. Между соседними секторами находится межсекторный интервал. Емкость форматированного диска обычно на 15 % меньше емкости неформатированного, на нем учитываются преамбулы, ЕСС-коды и межсекторные интервалы. У всех дисков есть кронштейны, они могут перемещаться туда и обратно по радиусу на разные расстояния от шпинделя, вокруг которого вращается диск. На разных расстояниях от оси записываются разные дорожки. Таким образом, дорожки представляют собой ряд концентрических кругов, расположенных вокруг шпинделя. Ширина дорожки зависит от величины головки и от точности ее перемещения. На сегодняшний момент диски имеют от 5000 до 10 000 дорожек на см, то есть ширина каждой дорожки составляет от 1 до 2 микрон.

Следует отметить, что дорожка - это не углубление на поверхности диска, а просто кольцо намагниченного материала, которое отделяется от других дорожек небольшими пограничными областями. Плотность записи битов на концентрических дорожках различается в зависимости от расстояния от центра диска и зависит главным образом от качества поверхности диска и чистоты воздуха.

Чтобы достичь высокого качества поверхности и достаточной чистоты воздуха, диски герметично закрываются. Такие диски называются **винчестерами**.

Большинство магнитных дисков состоит из нескольких пластин, расположенных друг под другом, как показано на рисунке 14.



Каждая поверхность снабжена кронштейном и головкой. Кронштейны скреплены таким образом, что одновременно могут перемещаться на разные расстояния от оси. Совокупность дорожек, расположенных на одном расстоянии от центра, называется **цилиндром**.

# Диски CD-ROM

Компакт-диск изготавливается с использованием очень мощного инфракрасного лазера, который выжигает отверстия диаметром 0,8 микрон в специальном стеклянном мастер-диске. Поэтому мастер-диску делается шаблон с выступами в тех местах, где лазер прожег отверстия. В шаблон вводится жидкая смола (поликарбонат), и, таким образом, получается компакт-диск с тем же набором отверстий, что и в стеклянном диске. На смолу наносится очень тонкий слой алюминия, который, в свою очередь, покрывается защитным лаком. После этого наклеивается этикетка. Углубления в нижнем слое смолы называются **лунками** (pits), а ровные пространства между лунками — **площадками** (lands).

Лунки и площадки записываются по спирали. Запись начинается на некотором расстоянии от отверстия в центре диска и продвигается к краю, занимая 32 мм диска. Спираль проходит 22 188 оборотов вокруг диска (примерно 600 на 1 мм).

Если спираль распрямить, ее длина составит 5,6 км. Спираль изображена на рисунке 15.

# DVD диски

Изначально аббревиатура DVD расшифровывалась как Digital Video Disk (цифровой видеодиск), сейчас она официально превратилась в Digital Versatile Disk (цифровой многоцелевой диск). DVD-диски в целом похожи на компакт-диски. Как и обычные компакт-диски, они имеют 120 мм в диаметре, создаются на основе поликарбоната и содержат лунки и площадки, которые освещаются лазерным диодом и считываются фотодетектором. Однако существует несколько различий:

* меньший размер лунок (0,4 микрона вместо 0,8 микрона, как у обычногоком- пакт-диска);
* более плотная спираль (0,74 микрона между дорожками вместо 1,6 микрона);
* красный лазер (с длиной волны 0,65 микрона вместо 0,78 микрона).

# Диски Blu-Ray

Преемником DVD можно считать технологию Blu-Ray, предусматривающую применение синего лазера вместо красного. Синий лазер отличается более короткой длиной волны, а значит, повышенной точностью; за счет этого обстоятельства он позволяет уменьшать размеры лунок и площадок. На односторонних дисках Blu- Ray умещается около 25 Гбайт данных; на двухсторонних — 50 Гбайт. Скорость передачи данных составляет 4,5 Мбит/с.

# «Флэшки»

У классической flash карты (как usb flash, так и карты памяти Compact Flash, SD, MMC и т.п.) нет механических частей, она не нуждается ни в батарейках, ни в аккумуляторах, флешка это набор микросхем, в чипах которых способна хранится цифровая информация. Это устройство компактное, быстрое, дешёвое и не самое надёжное.

Каждая из микросхем флешки состоит из своего рода гнёзд (NAND Flash). При перезаписи информация записывается в одно и то же гнездо, и через некоторое время сектор может начать сбоить, что зачастую и приводит к потере данных — это самая большая проблема флешек, обратная сторона удобства и дешевизны.

Чем больше объем flash памяти, тем больше внутри микросхем. Все вместе они выстроены в рейд-массив, но алгоритм их сборки и взаимодействия значительно сложней, чем классический рейд-массив из жестких дисков в компьютере. Более того, на каждой из микросхем для ускорения работы и повышения надёжности применяются алгоритмы схожие с принципами действия рейд-массивов. Вдобавок к этим сложностям во флешке существует самая главная микросхема — контролер, управляющий всеми данными на flash, запоминающий какие секторы флешки уже начали сбоить, считающий контрольные суммы. При загрузке обычной фотографии на flash информация не выкладывается единым куском, а распространяется по многим «хранилищам» довольно причудливым образом. Контролер «знает» о том, куда и какие части файла были положены.

* + - 1. Основные компоненты памяти

Значки, которые используются для изображения трех основных типов вентилей, показаны на рисунке 16 *а-в*. Там же показаны режимы работы функции для каждой схемы.

Здесь *А* и *В* — входные сигналы, *X* — выходной сигнал. Каждая строка таблицы определяет выходной сигнал для различных комбинаций входных сигналов.

**Защелки**. Чтобы создать один бит памяти, нужна схема, которая каким-то образом «запоминает» предыдущие входные значения. Такую схему можно сконструировать из двух вентилей НЕ-ИЛИ, как показано на рисунке 17 (а).



Аналогичные схемы можно построить из вентилей НЕ-И, которые, по существу, идентичны схемам с вентилями НЕ-ИЛИ.

Схема, изображенная на рисунке, называется **SR-защелкой**. У нее есть два входа: S (Setting - установка) и R (Resetting - сброс). У нее также есть два ком- плиментарных выхода: *Q* и *Q* .

Часто удобно, чтобы защелка меняла состояние только в определенные моменты. Чтобы достичь этой цели, немного изменим основную схему и получим **синхронную SR-защелку** (рисунок 18)**.**



Эта схема имеет дополнительный синхронизирующий вход, который по большей части равен 0. Если этот вход равен 0, то оба выхода вентилей И равны 0, и независимо от значений S и R защелка не меняет свое состояние. Когда значение синхронизирующего входа равно 1, действие вентилей *И* прекращается, и состояние защелки становится зависимым от S и R. Для обозначения факта появления единицы на синхронизирующем входе часто используются термины **включение** и **стробирование**.

В случае, если *S = R= 1* возникает неопределенность защёлки. Чтобы не дать ей возникнуть организуется схема только с одним входом. Такая схема (рисунок 19), которая называется **синхронной D-защелкой**, представляет собой память объемом 1 бит.



**Триггеры**. Многие схемы при необходимости выбирают значение на определенной линии в заданный момент времени и запоминают его. В такой схеме, которая называется **триггером**(flip-flop), смена состояния происходит не тогда, когда синхронизирующий сигнал равен 1, а при переходе синхронизирующего сигнала с 0 на 1 (фронт) или с 1 на 0 (спад). Следовательно, длина синхронизирующего импульса не имеет значения, поскольку переходы происходят быстро.

Различие между триггером и защелкой определяется тем, что триггер запускается перепадом сигнала, а защелка запускается уровнем сигнала.

Простейшая схема триггера (рисунок 20):



**Регистры**. Регистр представляет собой цифровую электронную схему (функциональный узел), служащую для временного хранения двоичных чисел, состоящую из триггеров и защёлок.

По существу, это наборы триггеров с независимыми информационными входами и (обычно) общим тактовым входом. В качестве регистров подобного рода могут быть использованы без дополнительных элементов многие типы синхронных триггеров. 

# Лекция 5. Архитектура систем

* + - * 1. SMP-архитектура

**SMP** (*symmetric multiprocessing*) – симметричная многопроцессорная архитектура. Главной особенностью систем с архитектурой SMP является наличие *общей физической памяти*, разделяемой всеми процессорами.



Память служит, в частности, для передачи сообщений между процессорами, при этом все вычислительные устройства при обращении к ней имеют равные права и одну и ту же адресацию для всех ячеек памяти. Поэтому SMP-архитектура называется *симметричной*. Последнее обстоятельство позволяет очень эффективно обмениваться данными с другими вычислительными устройствами.

SMP-система строится на основе высокоскоростной системной шины (SGI PowerPath, Sun Gigaplane, DEC TurboLaser), к слотам которой подключаются функциональные блоки: процессоры (ЦП), подсистема ввода/вывода (I/O) и т. п. Для подсоединения к модулям I/O используются уже более медленные шины (PCI, VME64). Наиболее известными SMP-системами являются SMP-cерверы и рабочие станции на базе процессоров Intel (IBM, HP, Compaq, Dell, ALR, Unisys, DG, Fujitsu и др.) Вся система работает под управлением единой ОС (обычно UNIX- подобной, но для Intel-платформ поддерживается Windows NT). ОС автоматически (в процессе работы) распределяет процессы по процессорам, но иногда возможна и явная привязка.

Основные преимущества SMP-систем:

* *простота и универсальность для программирования*. Архитектура SMP не накладывает ограничений на модель программирования, используемую при создании приложения: обычно используется модель параллельных ветвей, когда все процессоры работают независимо друг от друга. Однако можно реализовать и модели, использующие межпроцессорный обмен. Использование *общей памяти* увеличивает *скорость* такого обмена, пользователь также имеет доступ сразу ко всему объему памяти. Для SMP-систем существуют довольно эффективные средства автоматического распараллеливания;
* простота эксплуатации. Как правило, SMP-системы используют систему кондиционирования, основанную на воздушном охлаждении, что облегчает их техническое обслуживание;
* относительно невысокая цена.

Недостатки:

* системы с *общей памятью* плохо масштабируются.

Этот существенный недостаток SMP-систем не позволяет считать их по- настоящему перспективными. Причиной плохой *масштабируемости* является то, что в данный момент шина способна обрабатывать только одну транзакцию, вследствие чего возникают проблемы разрешения конфликтов при одновременном обращении нескольких процессоров к одним и тем же областям *общей физической памяти*. Вычислительные элементы начинают друг другу мешать. Когда произойдет такой конфликт, зависит от скорости связи и от количества вычислительных элементов. В настоящее время конфликты могут происходить при наличии 8-24 процессоров. Кроме того, системная шина имеет ограниченную (хоть и высокую) пропускную способность (ПС) и ограниченное число слотов. Все это очевидно препятствует увеличению производительности при увеличении числа процессоров и числа подключаемых пользователей. В реальных системах можно задействовать не более 32 процессоров. Для построения масштабируемых систем на базе SMP используются кластерные или NUMA-архитектуры.

* + - * 1. MPP-архитектура

**MPP** (*massive parallel processing*) – ***массивно-параллельная архитектура***. Главная особенность такой архитектуры состоит в том, что память физически разделена. В этом случае система строится из отдельных модулей, содержащих процессор, локальный банк операционной памяти (ОП), *коммуникационные процессоры* (рутеры) или *сетевые адаптеры*, иногда – жесткие диски и/или другие устройства ввода/вывода. По сути, такие модули представляют собой полнофункциональные компьютеры.



Доступ к банку ОП из данного модуля имеют только процессоры (ЦП) из этого же модуля. Модули соединяются специальными коммуникационными каналами. Пользователь может определить логический номер процессора, к которому он подключен, и организовать обмен сообщениями с другими процессорами.

Используются *два варианта работы операционной системы* (ОС) на компьютерах MPP- архитектуры. В одном полноценная операционная система (ОС) работает только на управляющем компьютере (front-end), на каждом отдельном модуле функционирует сильно урезанный вариант ОС, обеспечивающий работу только расположенной в нем ветви параллельного приложения. Во втором варианте на каждом модуле работает полноценная UNIX-подобная ОС, устанавливаемая отдельно.

**Главным преимуществом** систем с раздельной памятью является хорошая *масштабируемость*: в отличие от SMP-систем, в компьютерах с раздельной памятью каждый процессор имеет доступ только к своей *локальной памяти*, в связи с чем не возникает необходимости в потактовой синхронизации процессоров. Практически все рекорды по производительности на сегодня устанавливаются на компьютерах именно такой архитектуры, состоящих из нескольких тысяч процессоров (ASCI Red, ASCI Blue Pacific).

Недостатки:

* *отсутствие общей памяти* заметно снижает *скорость межпроцессорного обмена*, поскольку нет общей среды для хранения данных, предназначенных для обмена между процессорами. Требуется специальная техника программирования для реализации обмена сообщениями между процессорами;
* каждый процессор может использовать *только ограниченный объем* локального банка памяти;
* вследствие указанных архитектурных недостатков требуются значительные усилия для того, чтобы максимально использовать системные ресурсы. Именно этим определяется *высокая цена программного обеспечения* для массивно-параллельных систем с раздельной памятью.

Системами с раздельной памятью являются суперкомпьютеры МВС-1000, IBM RS/6000 SP, SGI/CRAY T3E, системы ASCI, Hitachi SR8000, системы Parsytec.

Компьютеры серии CRAY T3E от SGI, основанные на базе процессоров Dec Alpha 21164 с пиковой производительностью 1200 Мфлопс/с (CRAY T3E-1200), способны масштабироваться до 2048 процессоров.

* + - * 1. Гибридная архитектура NUMA

Главная особенность гибридной архитектуры **NUMA** (*non uniform memory access*) – неоднородный доступ к памяти.

Гибридная архитектура совмещает достоинства систем с общей памятью и относительную дешевизну систем с раздельной памятью. Суть этой архитектуры – в особой организации памяти, а именно: память физически распределена по различным частям системы, но логически она является общей, так что пользователь видит единое адресное пространство. Система построена из однородных базовых модулей (плат), состоящих из небольшого числа процессоров и блока памяти.

Модули объединены с помощью высокоскоростного коммутатора. Поддерживается единое адресное пространство, аппаратно поддерживается доступ к удаленной памяти, т.е. к памяти других модулей. При этом доступ к локальной памяти осуществляется в несколько раз быстрее, чем к удаленной. По существу, архитектура NUMA является MPP-архитектурой, где в качестве отдельных вычислительных элементов берутся SMP-узлы. Доступ к памяти и обмен данными внутри одного SMP-узла осуществляется через локальную память узла и происходит очень быстро, а к процессорам другого SMP-узла тоже есть доступ, но более медленный и через более сложную систему адресации.

Структурная схема системы с гибридной сетью: четыре процессора связываются между собой при помощи кроссбара в рамках одного SMP-узла. Узлы связаны сетью типа "бабочка" (Butterfly):



Понятие **когерентности кэшей** описывает тот факт, что все центральные процессоры получают одинаковые значения одних и тех же переменных в любой момент времени. Действительно, поскольку кэш-память принадлежит отдельному компьютеру, а не всей многопроцессорной системе в целом, данные, попадающие в кэш одного компьютера, могут быть недоступны другому. Чтобы этого избежать, следует провести синхронизацию информации, хранящейся в кэш-памяти процессоров.

Наиболее известными системами архитектуры cc-NUMA являются: HP 9000 V-class в SCA-конфигурациях, SGI Origin3000, Sun HPC 15000, IBM/Sequent NUMA-Q 2000. На сегодня максимальное число процессоров в cc-NUMA- системах может превышать 1000 (серия Origin3000). Обычно вся система работает под управлением единой ОС, как в SMP. Возможны также варианты динамического "подразделения" системы, когда отдельные "разделы" системы работают под управлением разных ОС.

* + - * 1. PVP–архитектура

**PVP** (Parallel Vector Process) – параллельная архитектура с векторными процессорами. Основным признаком PVP-систем является наличие специальных векторно-конвейерных процессоров, в которых предусмотрены команды однотипной обработки векторов независимых данных, эффективно выполняющиеся на *конвейерных функциональных устройствах*. Как правило, несколько таких процессоров (1-16) работают одновременно с общей памятью (аналогично SMP) в рамках многопроцессорных конфигураций. Несколько узлов могут быть объединены с помощью коммутатора (аналогично MPP). Поскольку передача данных в векторном формате осуществляется намного быстрее, чем в скалярном (максимальная скорость может составлять 64 Гбайт/с, что на 2 порядка быстрее, чем в скалярных компьютерах), то проблема взаимодействия между потоками данных при *распараллеливании* становится несущественной. И то, что плохо *распараллеливается* на скалярных машинах, хорошо *распараллеливается* на векторных.

Таким образом, системы PVP-архитектуры могут являться *компьютерами общего назначения* (general purpose systems). Однако, поскольку *векторные процессоры* весьма дорого стоят, эти компьютеры не могут быть общедоступными.

Наиболее популярны три типа компьютеров PVP-архитектуры:

1. **CRAY X1**, **SMP**-архитектура. Пиковая производительность системы в стандартной конфигурации может составлять десятки терафлопс.
2. **NEC SX-6**, **NUMA**-архитектура. Пиковая производительность системы может достигать 8 Тфлопс, производительность одного процессора составляет 9,6 Гфлопс. Система масштабируется с единым образом операционной системы до 512 процессоров.
3. **Fujitsu-VPP5000** (**vector parallel processing**), **MPP**-архитектура. Производительность одного процессора составляет 9.6 Гфлопс, пиковая производительность системы может достигать 1249 Гфлопс, максимальная емкость памяти – 8 Тбайт. Система масштабируется до 512.

**FLOPS** (или *flops* или *flop*/s, или **флопс**) — величина, используемая для измерения производительности компьютеров, показывающая, сколько операций с плавающей запятой в секунду выполняет данная вычислительная система.

5. Кластерная архитектура

*Кластер*представляет собой два или более компьютеров (часто называемых узлами), объединяемые при помощи *сетевых технологий* на базе шинной архитектуры или коммутатора и предстающие перед пользователями в качестве единого информационно-вычислительного ресурса. В качестве узлов *кластера* могут быть выбраны серверы, рабочие станции и даже обычные персональные компьютеры. Узел характеризуется тем, что на нем работает единственная копия операционной системы. Преимущество кластеризации для повышения работоспособности становится очевидным в случае сбоя какого-либо узла: при этом другой узел *кластера* может взять на себя нагрузку неисправного узла, и пользователи не заметят прерывания в доступе. Возможности *масштабируемости кластеров* позволяют многократно увеличивать производительность приложений для большего числа пользователей технологий (Fast/Gigabit Ethernet, Myrinet) на базе шинной архитектуры или коммутатора. Такие суперкомпьютерные системы являются самыми дешевыми, поскольку собираются на базе *стандартных комплектующих элементов* ("off the shelf"), процессоров, коммутаторов, дисков и внешних устройств.

Кластеризация может осуществляться на разных уровнях компьютерной системы, включая аппаратное обеспечение, операционные системы, программы-утилиты, системы управления и приложения. Чем больше уровней системы объединены кластерной технологией, тем выше надежность, *масштабируемость* и управляемость *кластера*.

Типы кластеров

Условное деление на классы предложено Язеком Радаевским и Дугласом Эдлайном:

* **Класс I**. Класс компьютеров строится целиком из стандартных деталей, которые продают многие поставщики компьютерных компонентов (низкие цены, простое обслуживание, аппаратные компоненты доступны из различных источников).
* **Класс II**. Система имеет эксклюзивные или не слишком широко распространенные детали. Таким образом, можно достичь очень хорошей производительности, но при более высокой стоимости.

*Кластеры* могут существовать в различных конфигурациях. Наиболее распространенными типами *кластеров* являются:

* системы высокой надежности;
* системы для высокопроизводительных вычислений;
* многопоточные системы.

Границы между этими типами *кластеров* до некоторой степени размыты, и *кластер* может иметь такие свойства или функции, которые выходят за рамки перечисленных типов. Более того, при конфигурировании большого *кластера*, используемого как *система общего назначения*, приходится выделять блоки, выполняющие все перечисленные функции.

*Кластеры* для высокопроизводительных вычислений предназначены для параллельных расчетов. Эти *кластеры* обычно собраны из большого числа компьютеров. Разработка таких *кластеров* является сложным процессом, требующим на каждом шаге согласования таких вопросов как инсталляция, эксплуатация и одно- временное управление большим числом компьютеров, технические требования параллельного и высокопроизводительного доступа к одному и тому же систем- ному файлу (или файлам) и межпроцессорная связь между узлами, и координация работы в параллельном режиме. Эти проблемы проще всего решаются при обеспечении единого образа операционной системы для всего *кластера*. Однако реализовать подобную схему удается далеко не всегда и обычно она применяется лишь для не слишком больших систем.

***Многопоточные системы*** используются для обеспечения единого интерфейса к ряду ресурсов, которые могут со временем произвольно наращиваться (или сокращаться). Типичным примером может служить группа web-серверов.

В 1994 году Томас Стерлинг (Sterling) и Дон Беккер (Becker) создали 16-узловой *кластер* из процессоров Intel DX4, соединенных сетью 10 Мбит/с Ethernet с дублированием каналов. Они назвали его "Beowulf" по названию старинной эпической поэмы. *Кластер* возник в центре NASA. Beowulf-*кластер*, как правило, является системой, состоящей из одного серверного узла (который обычно называется головным), а также одного или нескольких подчиненных (вычислительных) узлов, соединенных посредством стандартной компьютерной сети. Система строится с использованием стандартных аппаратных компонентов, таких как ПК, запускаемые под Linux, стандартные сетевые адаптеры (например, Ethernet) и коммутаторы. Нет особого программного пакета, называемого "Beowulf". Вместо этого имеется несколько кусков программного обеспечения, которые многие пользователи нашли пригодными для построения *кластеров* Beowulf. Beowulf использует такие программные продукты как операционная система Linux, системы передачи сообщений PVM, MPI, системы управления очередями заданий и другие стандартные продукты. Серверный узел контролирует весь *кластер* и обслуживает файлы, направляемые к клиентским узлам.

# Проблемы выполнения сети связи процессоров в кластерной системе

Архитектура кластерной системы (способ соединения процессоров друг с другом) в большей степени определяет ее производительность, чем тип используемых в ней процессоров. Критическим параметром, влияющим на величину производительности такой системы, является расстояние между процессорами. Так, соединив вместе 10 персональных компьютеров, мы получим систему для проведения высокопроизводительных вычислений. Проблема, однако, будет состоять в поиске наиболее эффективного способа соединения стандартных средств друг с другом, поскольку при увеличении производительности каждого процессора в 10 раз производительность системы в целом в 10 раз не увеличится.

Рассмотрим для примера задачу построения симметричной 16-процессорной системы, в которой все процессоры были бы равноправны. Наиболее естественным представляется соединение в виде плоской решетки, где внешние концы используются для подсоединения внешних устройств.



Но при соединении 16 процессоров друг с другом плоская схема является нецелесообразной. Для соединения 16 процессоров потребуется четырехмерный гиперкуб. Для его построения следует взять обычный трехмерный куб, сдвинуть в нужном направлении и, соединив вершины, получить гиперкуб размером 4.

Топология связи, 4-х мерный гиперкуб:



Архитектура гиперкуба является второй по эффективности, но самой наглядной. Используются и другие топологии сетей связи: трехмерный тор, "кольцо", "звезда" и другие.

Архитектура кольца с полной связью по хордам (Chordal Ring):



Наиболее эффективной является архитектура с топологией "толстого дерева" (fat-tree). Процессоры локализованы в листьях дерева, в то время как внутренние узлы дерева скомпонованы во внутреннюю сеть. Поддеревья могут общаться между собой, не затрагивая более высоких уровней сети.

Кластерная архитектура "Fat-tree":

Поскольку способ соединения процессоров друг с другом больше влияет на производительность кластера, чем тип используемых в ней процессоров, то может оказаться более целесообразным создать систему из большего числа дешевых компьютеров, чем из меньшего числа дорогих.

Кластерная архитектура "Fat-tree" (вид сверху на предыдущую схему):