**Архитектура вычислительных систем**

# 1. Однопроцессорные архитектуры

Архитектура - это средства превращения программного параллелизма в производительность. Процессор по очереди выбирает команды программы и также по очереди обрабатывает данные.

**1)SISD-архитектура**

Архитектуры данного класса предполагают выполнение только одного потока команд с последовательной обработкой единственного потока данных - SISD (Single Instruction Single Data, один поток команд - один поток данных).

В настоящее время имеется три воплощения SISD-архитектур:

- Coraplete Instruction Set Computing (CISC) - архитектура вычислений с полной системой команд;

- Reduced Instruction Set Computing (RISC) - архитектура вычислений с сокращенным набором команд;

- суперскалярная архитектура, реализующая многофункциональный параллелизм.

**CISC :** Компьютеры с CISC (Complex Instruction Set Computer) архитектурой имеют комплексную (полную) систему команд, под управлением которой выполняются всевозможные операции типа «память-память», «память-регистр», «регистр — память», «регистр — регистр».

Данная архитектура характеризуется:

- большим числом команд (более 200);

- переменной длиной команд (от 1 до 11 байт);

- значительным числом способов адресации и форматов команд;

- сложностью команд и многотактностыо их выполнения;

- наличием микропрограммного управления, что снижает быстродей­ствие и усложняет процессор.

Обмен с памятью в процессе выполнения команды делает практически невозможной глубокую конвейеризацию арифметики, т.е. ограничивается тактовая частота процессора, а значит, и его производительность.

**RISC :** Подразумевает обработку данных, хранящихся только в регистрах. Это дает преимущество в производительности перед CISC-архитектурой, но ценой включения в программу дополнительных команд обмена регистров процессора с оперативной памятью. Отсюда следует, что RISC архитектура производительнее в тех приложениях, где над каждой единицей данных выполняется большой объем вычислительной работы (инженерные и научные задачи), в противном случае часто приходится загружать данные из памяти.

**Суперскалярная архитектура :** базируется на многофункциональном параллелизме и имеет средства, позволяющие одновременно выполнять две или более скалярные операции. Это дает возможность увеличить производительность компьютера пропорционально числу одновременно выполняемых операций. Существует два способа реализации суперскалярной обработки: динамический и статический.

*Динамический способ* - реализуется в архитектуре процессора с предсказанием ветвлений и заключается в чисто аппаратном механизме выборки из ”кэша команд” несвязанных инструкций и их параллельном запуске на исполнение, например, как это имеет место в процессорах DEC серии Alpha, семействе процессоров HP РА-8х00, Intel Pentium-Pro и т.д. Основной проблемой динамической суперскалярной архитектуры является аппаратное предсказание ветвлений.

*Статический способ* реализации суперскалярной обработки основан на VLIW-архитектуре (Very Large Instruction Word) - архитектуре с очень длинным (широким) командным словом.

**Многопотоковая архитектура** основана на введении множества устройств выборки команд на процессорном кристалле, каждое из которых имеет окно выполнения декодированных операций одного потока. Данная и следующая архитектуры фактически переводят SISD-архитектуру процессорного кристалла в класс MIMD архитектур.

**Многоядерная архитектура** реализует несколько полноценных процессорных ядер на одном кристалле [38, 40, 47, 53, 54, 60, 62, 98, 103]. Каждое ядро при этом имеет все атрибуты традиционного процессора - набор исполнительных устройств, регистровый файл, устройство предсказания переходов, индивидуальный кэш первого уровня данных и команд, логику многопотоковой обработки и др. Общим для всех процессорных ядер одного кристалла является, как правило, только кэш второго уровня.

**2) SIMD –архитектура (один поток команд - много потоков данных)**

Различают два принципа реализации векторной обработки: матричный и векторно-конвейерный. **Матричный принцип** - построение системы из четырех квадрантов по 64 процессорных элемента (ПЭ) и 64 модуля локальной памяти при каждом ПЭ, объединенных коммутатором на основе сети типа гиперкуб. Все ПЭ квадранта обрабатывают одну векторную инструкцию, которую им направляет процессор команд.

**Векторно-конвейерный принцип обработки данных** - здесь применяется единственный конвейер операций, имеющий один вход, по которому поступают операнды, и один выход результата, тогда как в матричных системах существует множество входов по данным в процессорные элементы и множество выходов из них.

# 2. Многопроцессорные архитектуры

Многопроцессорные системы могут быть представлены двумя базовыми типами архитектуры в зависимости от параллелизма данных - MISD (Multiple Instruction Single Data, множество потоков команд - один поток данных) и MIMD (Multiple Instruction Multiple Data, множество потоков команд - множество потоков данных).

**MIMD-архитектура** включает все уровни параллелизма от конвейера операций до независимых заданий и программ. Поэтому любая вычислительная система этого класса в частных приложениях может выступать как SISD или SIMD-система.

Чтобы использовать все возможности MIMD архитектуры, необходимо загрузить множество процессоров системы работой в виде множества вычислительных процессов. Существует ’’дерево” многопроцессорных систем. Различают современные высокопроизводительные многопроцессорные вычислительные системы, построенные на основе MIMD-архитектур с сильной (SMP-вычислители) и со слабой (МРР-вычислители) связью процессоров с оперативной памятью вычислительной системы.

SMP -системы. Вычислительные системы с сильной связью ("истинные мультипроцессоры” ), называемые также системами с симметричной мультипроцессорностью, - основаны на объединении процессоров на общем поле памяти, которые совместно с интерфейсной шиной или коммутатором’’ процессоры-память” при увеличении числа процессоров становится узким местом системы.

МРР-системы. Вычислительные системы со слабой связью состоят из узлов (процессор, память, система ввода вывода), работающих под управлением своей копии операционной системы на своем уникальном адресном пространстве. Узлы объединяются между собой коммутационными сетями регулярной структуры (решетка, гиперкуб, тор). МРР-системы называют ’’системами с управлением потоком данных” ("потоковые машины” ) или "системы с архитектурой, отличной от архитектуры фон Неймана” .

# 3. Принципы организации CISC- и RISС- процессоров

Лидером в разработке микропроцессоров c полным набором команд (CISC - Complete Instruction Set Computer) считается компания Intel со своей серией x86 и Pentium. Эта архитектура является практическим стандартом для рынка микрокомпьютеров. Для CISC-процессоров характерно: сравнительно небольшое число регистров общего назначения; большое количество машинных команд, некоторые из которых нагружены семантически аналогично операторам высокоуровневых языков программирования и выполняются за много тактов; большое количество методов адресации; большое количество форматов команд различной разрядности; преобладание двухадресного формата команд; наличие команд обработки типа регистр-память.

*4 основных принципа RISC- архитектуры*.

1. Любая операция, вне зависимости от ее типа, должна выполняться за один такт.
2. Система команд должна содержать минимальное количество наиболее часто используемых простейших инструкций одинаковой длины с минимумом адресных форматов.
3. Операции обработки данных реализуются только в формате "регистр-регистр” . Обмен ’’регистр-память” выполняется только с помощью команд ”загрузки-записи” .
4. Состав системы команд должен быть ”удобен” для компиляции операторов языков высокого уровня.

Специфика организации RISC-процессоров во многом определяется "целевой функцией” их базовой архитектуры, которая предполагает достижение возможного максимума производительности путем использования мощных средств обработки. Это достигается тремя структурными приемами:

1. Полноразрядной обработкой данных (каждый арифметический конвейер за каждый так т формирует один 64-разрядный результат, что является оптимальным форматом данных с плавающей и фиксированной точкой для вычислительных задач).
2. Конвейерной организацией арифметических и других функциональных исполнительных устройств.
3. Суперскалярной структурой процессора с комплексом средств динамического прогнозирования ветвлений, большим ко­личеством регистров (минимум 32 регистра общего назначения, выполняющих роль сверхоперативной памяти), многоуровневым кэшем.

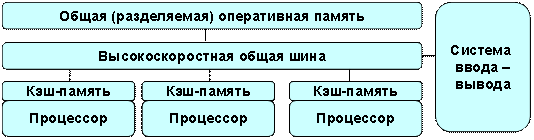
# 4. Принципы выбора архитектуры

1. Производительность компьютера определяется главным образом двумя характеристиками - тактовой частотой и разрядностью обработки. Конвейерная организация процессора "работает” на усиление значимости тактовой частоты и является необходимым атрибутом современных компьютеров.
2. Для высокопроизводительных вычислений целесообразно применять RISC-процессоры или процессоры с RISC-ядром. Системы на их основе специально ориентированы на повышение тактовой частоты обработки и ”прозрачность” параллелизма архитектуры для прикладного программиста.
3. Предпочтительней использовать один быстрый однопроцессорный компьютер, чем много медленных или многопроцессорную систему с маломощными процессорами.
4. Проблему создания необходимого количества рабочих мест лучше решать с помощью многопользовательских систем на базе сервера и терминалов, чем с помощью ЛВС персональных компьютеров
5. Вычислительные системы с массовым параллелизмом следует использовать только при полной уверенности в реально существующем параллелизме приложений и только при возможности привлечения программистов высшей квалификации.

# 5. Подходы к организации многопроцессорных вычислительных систем

**SMP — симметричная многопроцессорная архитектура**

*Главной особенностью систем с архитектурой SMP  является наличие общей физической памяти, разделяемой всеми процессорами.*



***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*.

**CMP - перестраиваемая симметрично многопроцессорная архитектура**

Это перестраиваемая SMP-архитектура, которая позволяет комбинировать SMP и кластерные технологии [57]. Данная архитектура многопроцессорных серверных платформ предложена компанией Unisys. В основе СМР лежит модель однородного общего поля оперативно!! памяти. Строительные блоки в СМР - процессорные элементы (Unisys subpods), ориентированы на применение процессоров Intel Pentium Xeon и Mersed в рамках одной СМР-системы.

роцедуры, основанные на каталогах. В архитектуру СМР заложены уникальные возможности по статическому и динамическому парционированию (разбиению) SM P-сервера, приводящие к преобразованию в кластер всей SM P-системы, построенной из SM P-серверов с числом процессоров, кратным четырем.  
В СМР возможно три типа разделения поля оперативной памяти между разделами:

- каждая ОС использует только свою память;

- каждая ОС имеет свою память и образуется еще одна общая для разных ОС область памяти;

- каждая ОС имеет свою память и образуется несколько областей памяти, разделяемых некоторыми ОС.

**МРР - многопроцессорная архитектура с распределенной памятью**

Многопроцессорные вычислительные системы с распределенной (индивидуальной) памятью строятся на основе МРР архитектуры и относятся к классу NUMA. Узлы в архитектуре М РР обычно состоят из одного процессора, памяти и устройств ввода-вывода. В каждом узле работает своя копия ОС, а узлы объединяются между собой специализированным соединением.

**Кластеры - разновидность МРР-систем**

В области компьютерных технологий понятие ’’кластер” применяют в двух значениях. Различают вычислительные кластеры и кластеры, используемые для повышения надежности и отказоустойчивости серверов приложений (кластеры приложений). С содержательной точки зрения кластеры имеют МРРархитектуру с более дешевой, чем у вычислителей с массовым параллелизмом, коммуникационной подсистемой. В этом смысле кластеры являются одним из направлений развития компьютеров с массовым параллелизмом.

Кластер состоит из двух или более узлов, удовлетворяющих следующим требованиям:

- каждый узел работает со своей копией ОС;

- каждый узел работает со своей копией приложения;

- узлы делят общий пул других ресурсов (в кластерах приложений).

В отличие от кластера приложений в МРР-системах узлы не делят ресурсы для хранения. Это главное отличие между кластерными SMP-системами и традиционными МРР-системами.

Архитектура CC -NUMA данная архитектура является развитием SMP-архитектур и представляет собой совокупность многопроцессорных SMP-узлов с собственной памятью, объединенных между собой скоростной общей шиной либо коммутатором. Объединение узлов выполняется через оперативную память. Каждый процессор имеет собственную локальную память и может устанавливать статические и динамические соединения с модулями памяти других процессоров. По существу, CC-NUMA - это кэш-когерентный доступ к неоднородной памяти. В системе CC-NUMA физически распределенная память объединяется в единый массив [13, 34, 35, 89]. Существует одна карта памяти (единое адресное пространство) с частями, физически связанными специальной соединительной шиной, и аппаратно-реализованная кэш-когерентность.

# 6. Доменная архитектура многопроцессорных вычислительных систем

Доменная архитектура основана на концеппии разбиения SMP вычислителей, предусматривающей сегментирование ресурсов вычислительной системы и создание изолированных друг от друга разделов, обеспечивающих независимую работу разных приложений [51, 70]. Данная архитектура позволяет объединить в одном вычислителе несколько полнофункциональных независимых серверных систем, каждая из которых имеет свои процессоры, память, подсистему ввода-вывода и экземпляр операционной системы.

**Системные разделы**

Системное разбиение позволяет создать на базе одного аппаратного сервера несколько разделов, каждый из которых представляет собой полнофункциональную серверную систему, работающую под управлением собственного экземпляра ОС. Системное разбиение является необходимой функциональностью для провайдеров приложений. Системный раздел включает все необходимые для автономной работы ОС ресурсы - процессоры, память, устройства ввода вывода.

**Разделение приложений**

Разделение приложений является удобным механизмом управления ресурсами вычислительной системы при предоставлении информационных услуг. Возможность закрепления за конкретным приложением или группой пользователей необходимых ресурсов упрощает достижение соглашений об уровне обслуживания (QoS). При разделении приложений наборы ресурсов для рабочих областей приложений, контролируемых одним экземпляром ОС, выделяются в ’’ прикладные разделы” . Разделение приложений дополняет системное разбиение и допустимо только в определенных аппаратных границах; на одном автономном сервере, в серверном кластере, в одном системном разделе. Следующий уровень программного разбиения обеспечивает ” планирование классов” - сегментирование ресурсов прикладного раздела на несколько прикладных или пользовательских классов и явное выделение части ресурсов группе пользователей или приложений, входящих в класс. В основе разделения приложений лежит выделение некоторого числа процессорных блоков на сервере или системном разде­ле в процессорный набор.

# Архитектурные приемы

Простейший фактор, влияющий на производительность цп – его **тактовая частота**. Основной путь повышения производительности - построение архитектур, выполняющих за **такт** более одной команды.

**Способы увеличения производительности**:

1. Конвейеризация
2. Суперскалярные архитектуры
3. Неупорядоченное выполнение
4. Переименование регистров
5. Обходы и продвижение данных
6. Прогнозирование переходов
7. Превращение CISC-архитектуры в RISC
8. Блоки вычислений с плавающей точкой

## Конвейеризация

В большинстве классических CISC-архитектур **5 ступеней обработки** конвейера:

1) ВЫБОРКА КОМАНДЫ

2) ДЕКОДИРОВАНИЕ КОМАНДЫ

3) АДРЕСАЦИЯ И ВЫБОРКА ОПЕРАНДОВ

4) ИСПОЛНЕНИЕ КОМАНДЫ

5) ЗАПИСЬ РЕЗУЛЬТАТА

Конвейерная обработка представляет собой **параллельное выполнение всех ступеней обработки** в пределах одного и того же процесса. Каждая ступень конвейера обрабатывает свою фазу команды **независимо.** В каждый момент времени конвейерный процессор работает над выполнением различных стадий нескольких команд.

В общем случае различные команды процессора имеют неодинаковые длительности выполнения различных фаз.

Конвейерное выполнение команд имеет **максимальную эффективность**, когда продолжительность выполнения всех этапов команд одинакова, бесперебойно подаются команды и данные, на каждом этапе отсутствуют зоны ожидания выполнения аналогичного этапа очередной команды.

*Что нарушает непрерывность конвейерной реализации команд:*

1. Когда для выполнения следующей команды требуется результат действий, реализованных предыдущей командой.  
    Или когда предыдущей командой определяется адрес операнда следующей команды (модификация адреса). Тогда возникает **задержка начала выполнения следующей команды**, связанная с ожиданием выборки операнда следующей команды или с преобразованием адресов;
2. При ветвлении программы по результатам проверки условий командой условного перехода команды, находящиеся в процессе конвейерной обработки, остаются невыполненными, и требуется повторная загрузка конвейера, начиная с момента выборки команды условного перехода;
3. Когда в кэш-памяти отсутствуют требуемые данные или команды, необходимо передать их в кэш-память из основной памяти;
4. Когда предшествующая команда изменяет содержание последующей.  
   Или когда изменяется содержимое регистра состояния программы, задающего область выполнения программы, последующая команда должна ожидать завершения предшествующей команды;
5. В случае возникновения прерывания и перехода к программе его обработки команды, находящиеся в это время на командном конвейере, остаются незавершенными и приходится заново загружать конвейер командами, входящими в программу обработки прерывания;
6. Когда операция, реализуемая машинной командой, имеет сложный характер и требует для выполнения много машинных циклов, последующая команда долго не может достичь стадии выполнения операции. В современных процессорах пять основных ступеней делятся на многоступенные операции, позволяющие снизить сложность каждой ступени и обеспечить однородность конвейера, - такая организация вычислений называется **суперконвейерной**.

Идея суперконвейера заключается в сокращении числа логических операций, которые необходимо выполнить на каждой ступени конвейера с тем, чтобы можно было быстрее переходить со ступени на ступень и, следовательно, повысить тактовую частоту.

## 1.2 Суперскалярные архитектуры

Процессор *с единственным конвейером* называется **скалярным**. Процессоры с *несколькими конвейерами* называются **суперскалярными** (многопотоковыми).

Как правило, микропроцессоры имеют ограничения на типы команд, обрабатываемых одновременно. **Существует два подхода к отображению параллелизма обработки данных** на архитектурном уровне в системе команд :

1. Система команд не содержит внутри процессора никакого указания на параллельную обработку. В таких процессорах параллелизм исполнения команд формируется динамически аппаратурой микропроцессора.
2. В специальных полях команды каждому из параллельных исполнительных устройств указывается действие, которое устройство должно совершить. Это **процессоры с длинным командным словом** (VLIW). Загрузка параллельных исполнительных устройств этих микропроцессоров формируется специальными компиляторами с языков высокого уровня.

**Основная идея** суперскалярных микропроцессоров, состоит в том, что компиляторы и логика микропроцессора сами, без вмешательства программиста, обеспечивают загрузку параллельно работающих устройств микропроцессора.

## 1.3 Неупорядоченное выполнение

В конвейерах возможны **заторы (зависания)**, обусловленные взаимозависимостями команд в программе и способами построения суперскалярных архитектур.

1. Существуют архитектуры, использующие **«упорядоченное поступление и обработку»** и **«упорядоченное завершение».** В таких архитектурах все, что затрудняет завершение команды в одном конвейере, останавливает и другой конвейер, т. к. команды должны покидать конвейеры точно в том же порядке, в каком поступали на них. *Это упрощает конструкцию процессора, но увеличивает производительность.*
2. Архитектуры с **«неупорядоченным завершением»** и **«неупорядоченным исполнением»** позволяют одному из конвейеров продолжать работать при «заторе» в другом. При этом команды, стоящие в программе позже, могут быть выполнены раньше предыдущих команд, «застрявших» в другом конвейере. *Но процессор должен гарантировать, что результаты не будут записаны в память, а регистры модифицироваться в неправильной последовательности.*
3. **«Неупорядоченная обработка»** развивает предыдущую концепцию дальше, позволяя процессору «выдавать» и «обрабатывать» команды также не в порядке их следования по программе. Для реализации неупорядоченной обработки обычно требуется *буфер команд (или «окно команд»)* между ступенями декодирования и исполнения конвейера.

## 1.4 Переименование регистров

Производительность процессора ограничена из-за того, что некоторые операции над связанными данными принципиально нельзя завершить до выполнения других.

1. В выражении **A x B + C** сложение и умножение нельзя выполнить одновременно согласно правилам арифметики. Это **«истинная взаимозависимость данных»**, или взаимозависимостью **«чтение после записи»**, т.е. входные данные для одной операции зависят от результата другой.
2. Производительность может снижаться, кроме того, **«ложными взаимозависимостями»** двух видов;
   1. **Взаимозависимость по выходу**, или взамозависимость **«запись после записи»** возникает при выполнении двух, следующих друг за другом команд, которые записывают свои результаты в один и тот же регистр;
   2. **Антивзаимозависимость,** или взаимозависимость **«запись после чтения»** возникает, когда вторая команда может испортить данные, необходимые в качестве входных для первой.

Частое использование одних регистров приводит к возникновению взаимозависи­мостей и снижению производительности. *Путь повышения производительности - нейтрализация ложных взаимозависимостей с помощью переименования регистров.*

* **RISC-процессоры** имеют большое количество регистров и **ложные взаимозависимости возникают редко.**
* **CISC-процессоры**, ориентированные на широкое разнообразие методов адресации, оснащены небольшим числом регистров, что провоцирует **частое возникновение ложных взаимозависимостей**.

Если команде требуется регистр, процессор динамически ставит в соответствие этому логическому (архитектурному) регистру один из множества физических регистров. Если другая команда пытается обратиться к тому же логическому регистру, процессор для предотвращения конфликта может поставить ему в соответствие другой физический регистр. Такие переименования действуют, пока команды продвигаются по конвейерам.

## 1.5 Обходы и продвижение данных

Переименование регистров не может совсем исключить возникновение ИСТИННЫХ взаимозависимостей данных. Для нейтрализации таких взаимозависимостей используются методы **ОБХОДА ДАННЫХ** и **ПРОДВИЖЕНИЯ ДАННЫХ**.

1. При ОБХОДАХ результаты выполнения одной команды сразу пересылаются следующей и таким образом исключаются задержки на модификацию и повторное чтение из регистра или памяти.
2. При ПРОДВИЖЕНИИ ДАННЫХ процессор может выполнять некоторые команды параллельно, немедленно передавая результаты одной из них в другую, которой они потребуются на более поздней ступени конвейерной обработки. При этом также исключается ожидание завершения операций записи и чтения адресуемых операндов.

## **1.6 Прогнозирование переходов**

*Суперскалярность, неупорядоченная обаботка, продвижение данных* могут реализовать свой потенциал только при разрешении проблемы **процедурных взаимозависимостей**.

**Причина возникновения**: из-за наличия условных и безусловных переходов в программе (изменения последовательности выполнения команд). Т. к. *условные переходы* (ветвления) разрешаются только на исполнительной ступени конвейера, процессор не знает, какие команды следует направлять в конвейер за командой перехода.   
Здесь используется «предположение о пути ветвления» - **прогнозирование ветвления**. После прогноза процессор начинает:

* выборку команд с предсказанного адреса перехода до того, как он узнает, верным ли был его прогноз;
* исполнение по предположению «спекулятивное»; при этом процессор не может модифицировать архитектурные регистры или память до уточнения достоверности предсказанного перехода.

Некоторые процессоры обеспечивают несколько уровней предположений, прогнозируя дополнительные ветвления до разрешения первого.

*Приемы прогнозирования ветвлений:*

1. Использование **буфера адреса перехода** на 256 и более позиций, который отслеживает и хранит данные о результатах 256 последних ветвлений.

* Каждая строка буферной таблицы указателей перехода **включает адрес команды**, задаваемый командой перехода, **адрес указателя направления перехода** команды перехода и **предысторию команды перехода**.
* **Предыстория** содержит информацию о выполнении или невыполнении условий перехода данной команды до текущего момента.
* Во время **декодирования команды** перехода по предыстории прогнозируется выполнение или невыполнение условий команды перехода и производятся вызов и декодирование команд из прогнозируемой ветви программы.

1. Буфер адресов переходов дополняется **стеком возвратов**, который отслеживает переходы в паре команд CALL/RETURN.
2. Информация о переходах добавляется к каждой строке кэша команд.

* Здесь дается адрес первого перехода в строке кэшпамяти и указывается, как следует прогнозировать ветвление - как *выполняемое* или *невыполняемое*.
* Недостаток в том, что нельзя сохранить информацию о более чем одном переходе в строке кэша.

## 1.7 Превращение CISC-архитектуры в RISC

Один из подходов повышения быстродействия микропроцессора и обеспечения преемственности основан на RISC-ядре и схемах преобразования команд CISС во внутренние RISC-команды.

**Главная проблема декодирования** **CISC-команд**: определение *границ команд* и расположение их в такой последовательности, которая обеспечит наиболее эффективную обработку дешифратором, т. к. быстрое декодирование команд является необходимым условием для обработки нескольких команд за такт.

**Эффективное решение проблемы декодирования** состоит в распознавании сложных команд х86 до стадии декодирования на этапе загрузки команд из основной памяти в кэш-память команд. Это **предварительное декодирование** сопровождается формированием битов предварительного декодирования. При этом каждый байт в кэш-памяти команд имеет биты данных декодирования, указывающих начало и конец команды и число основных RISC-подобных операций, необходимых для ее выполнения.

Процесс предварительного декодирования требует дополнительной кэш-памяти. При выборке команд из кэша границы команд уже определены и RISC-преобразователи формируют серии RISC-команд.

## 1.8 Блоки вычислений с плавающей точкой

**Операции с плавающей точкой имеют высокую трудоемкость**. В связи с этим блоки вычислений с плавающей точкой в современных микропроцессорах реализуются в виде **многоступенчатых конвейеров**, позволяющих увеличить тактовую частоту исполнительных устройств.

Еще одним способом увеличения скорости выполнения операций с плавающей точкой является SIMD-реализация обработки векторных данных. В настоящее время параллельная обработка данных с плавающей точкой с помощью SIMD-расширений базового набора команд становится необходимым атрибутом практически всех микропроцессоров от различных производителей.

## 1.9 Кэш-память микропроцессора

Для сокращения потерь, связанных с доступом к внекристальной памяти, в составе микропроцессора имеется **внутренний кэш первого уровня** наиболее популярной гарвардской архитектуры *с разделением кэш-памяти на отдельные секции команд и данных.*

**Преимущество:** Возможность динамического разделения при работе программ большого объема или обрабатывающих большие массивы данных.

**Недостаток:** Возникновение конфликтов между доступом к коду и данным на различных стадиях конвейерной обработки. Поэтому чаще всего на первом уровне иерархической памяти в кристалле микропроцессора реализуется *раздельный кэш для данных и команд*.

Важнейшим фактором, определяющим быстродействие вычислительной системы, является **емкость кэша**. Объем кэша первого уровня современных микропроцессоров колеблется от 32 Кбайт до 1,5 или 2,25 Мбайт.

## 1.10 Многопотоковые и многоядерные микропроцессоры

* **Многопотоковые микропроцессоры** позволяют на основе одного набора исполнительных функциональных устройств орга­низовать *исполнение нескольких независимых потоков команд*. Данный подход направлен на увеличение загрузки устройств выполнения команд, которые в традиционных архитектурах загружены на реальных приложениях в среднем на 20-30%.
* Многопотоковые микропроцессоры требуют реализации для каждого потока блоков выборки команд и переименования регистров, механизмов неупорядоченного выполнения команд и прогнозирования переходов. При этом накладные расходы на организацию вычислений в каждом отдельном потоке, как правило, не превышают 5-10%.
* **Многоядерные микропроцессоры** содержат на кристалле несколько самостоятельных процессорных ядер. Это позволяет реализовать все рассмотренные выше архитектурные приемы увеличения быстродействия процессоров, в том числе многопотоковую архитектуру.

# **Тенденции развития микропроцессоров**

Основные усилия архитекторов микропроцессоров сосредоточены на следующих направлениях:

* Повышение тактовой частоты;
* Увеличение объема и пропускной способности подсистемы памяти;
* Увеличение количества параллельно работающих функциональных исполнительных устройств;
* Введение блоков обработки мультимедийных данных;
* Интеграция на кристалле функций управления памятью и периферийными устройствами, для исполнения которых в традиционных микропроцессорах используются наборы микросхем («чипсеты»);
* Интеграция на кристалле интерфейсов сетевых и телекоммуникационных систем, позволяющая соединять эти микропроцессоры друг с другом и телекоммуникационными и вычислительными сетями без дополнительных адаптеров;
* Создание многопроцессорных и многопотоковых (многонитевых) систем на одном кристалле, обеспечивающих распараллеливание на уровне процессов и потоков (нитей).

# **Архитектура микропроцессора PentiumPro (Р6)**

## 3.1 Общая характеристика

Повышение производительности микропроцессорной вычислительной системы возможно за счет:

* разработки высокопроизводительных компонент, окружающих микропроцессор;
* увеличения кэш-памяти 2-го уровня;
* повышения частоты микропроцессора;
* совершенствования микроархитектуры микропроцессора.

В архитектуре Р6 реализованы следующие способы увеличения производительности:

* суперскалярность (до 3-х команд х86 одновременно);
* суперконвейерность (14-ступенчатый конвейер);
* выполнение команд с изменением последовательности;
* предсказание переходов;
* исполнение по предположению;
* переименование регистров;
* преобразование команд х86 в RISC-подобные микрокоманды;
* кэш первого и второго уровней неблокирующего типа;
* кэш второго уровня
* на частоте микропроцессора;
* шина кэша второго уровня отделена от шины оперативной памяти;
* использование шины транзакций, допускающей одновременное выполнение операций ввода-вывода и обращений к оперативной памяти.

Весь набор архитектурных приемов, использованных в Р6, получил общее название **«Динамическое исполнение команд»**.

В Р6 использованы **2** основных *метода повышения быстродействия ЦП*:

1. повышение тактовой частоты благодаря суперконвейеризации
2. увеличение степени параллелизма (количества одновременно выполняемых операций в течение каждого периода тактовой частоты) благодаря механизму суперскалярного исполнения.

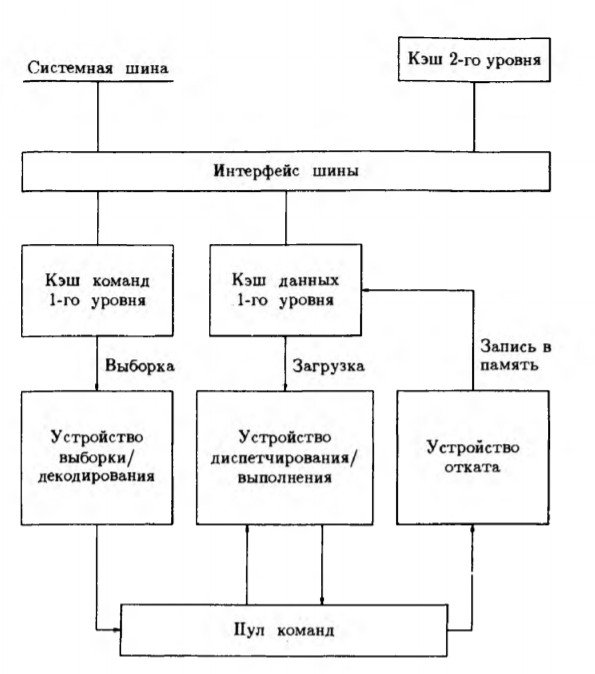
## 3.2 Архитектура суперконвейера

Упорядоченное устройство – устройство, которое работает в соответствии с исходным порядком команд в программе.

Беспорядочное устройство – устройство, которое не обращает внимания на исходный порядок команд в программе.

Конвейер Р6 имеет **14 ступеней (фаз)**, разделенных на **три независимых блока (устройства)**, взаимодействующих через пул команд:

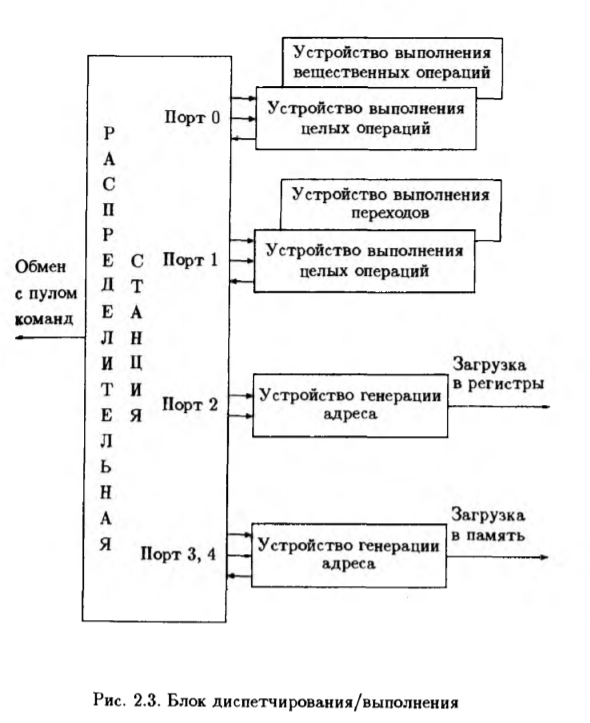
1. **входной блок выборки/декодирования команд**, состоящий из 8 ступеней и являющийся ” упорядоченным” устройством, работающим в соответствии с исходным порядком команд в коде программы;
2. **блок диспетчирования/выполнения команд** с изменением последовательности, где происходит собственно выполнение команд; имеет **3 ступени** и является ’’ беспорядочным” устройством, которое не обращает внимания на исходный порядок команд в программе, а планирует их выполнение с учетом зависимостей по данным и доступности ресурсов, вре­менно сохраняя результаты опережающего выполнения в пуле команд;
3. **блок отката (вывода) команд из последовательности**, состоящий из **3 ступеней** - ’’упорядоченное” устройство, отвечающее за перевод временных результатов опережающего выполнения в постоянное состояние вычислительной системы.

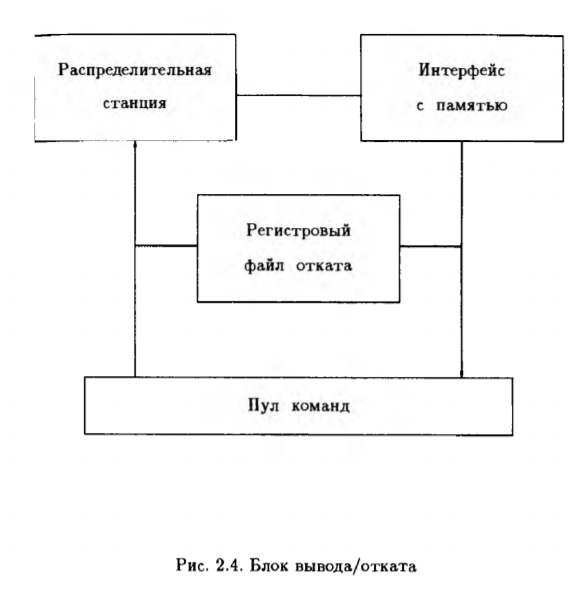


**Интерфейс шины** является ’’частично упорядоченным” устройством, отвечающим за связь 3 блоков суперконвейера с внешним миром. Интерфейс шины взаимодействует непосредственно с кэшем второго уровня и поддерживает до четырех параллельных обращений к кэшу.



**Блок выборки/декодирования** реализует функции определения указателя на следующую команду, выравнивания команд перед посылкой на дешифраторы, трансляции команд х86 (1-6-я ступени конвейера), отображения регистров (7-я, 8-я ступени конвейера). Преобразование команд х86 в RISC-подобные микрокоманды позволяет устранить ограничения набора команд х86, связанные с нерегулярностью кодирования команд, переменной длиной непосредственных операндов и частыми пересылками регистр-память. Выборка кода и трансляция его в микрокоманды происходит на первых шести ступенях конвейера Р6.





## **3.3 Предсказание переходов**

По оценкам экспертов, в типичной программе до 10% команд могут быть безусловными переходами и еще до 10-20% - условными переходами. При ошибке в предсказании перехода Рб может потерять от 4 до 15 тактов.

Существует два основных метода предсказания переходов - **статический и динамический.**

1. **Статические методы** предписывают всегда выполнять или не выполнять определенные типы переходов.
2. **Динамические методы** оценивают поведение команд перехода за предшествующий период времени, поскольку один и тот же переход часто выполняется более чем один раз, например в цикле.

Р6 применяет статические методы для предсказания поведения команд переходов, предыстория которых не была проанализирована с помощью динамических методов.   
В Р6 используется структура - буфер адреса перехода (branch target buffer - ВТВ) и предсказания, основанные на битах предыстории ветвлений.

# 3. Шинные интерфейсы

**Интерфейс** это совокупность аппаратных, программных и конструктивных средств и правил, обеспечивающих взаимодействие компонентов вычислительной системы или сети и учитывающих информационную, электрическую и конструктивную совместимость их элементов.

**Протокол** определяет правила взаимодействия элементов вычислительной системы и способ выполнения определенных функций.

**Канал** определяет среду распространения сигналов.

**Шина** это совокупность линий интерфейса, сгруппированных по функциональному назначению. Различают **шину данных, шину адреса, шину управления и состояния.**

Таким образом, шиной называют аппаратный стандарт, на основе которого выполняется управление связями между блоками компьютера. **Пропускная способность** любой шины измеряется в **килобайтах (мегабайтах) в секунду** и определяется ее **тактовой частотой и разрядностью (шириной)** шины данных.

**Пиковая скорость передачи** данных по шине определяется произведением этих параметров.

В общем случае в компьютере имеется **системная шина**, объединяющая цп с памятью и интерфейсом подключения внешних устройств, шина ввода-вывода, позволяющая подключать к компьютеру платы управления устройствами ввода-вывода (контроллеры), и периферий­ные шины, используемые для подключения к групповым платам управления множества однотипных устройств ввода-вывода.

## Шины ввода-вывода

Шину ввода-вывода называют также **шиной расширения**, поскольку она позволяет подключить к компьютеру широкую номенклатуру дополнительных периферийных устройств. Шинные интерфейсы ввода-вывода используются **только** в качестве шины для подключения контроллеров внешней периферии.

### Шина PCI и унаследованные шины

Шина PCI стала в настоящее время стандартом подключения внешних устройств для широкого разнообразия вычислительных архитектур, предлагаемых различными производителями. Для настольных систем реализуется обычно 32-разрядная версия стандарта, работающая на частоте 33 МГц, а для серверных платформ и высокопроизводительных рабочих станций используется четыре версии; 32 бит/33 МГц, 32 бит/66 МГц, 64 бит/33 МГц и самая скоростная 64 бит/66 МГц.

В настоящее время интерфейс PCI принят в качестве стандарта на шину ввода-вывода для вычислительных платформ на базе процессоров Alpha фирмы DEC и SPARC компании Sun Microsystems.

Для подключения большого числа контроллеров периферии, разработанных в стандарте ISA (EISA), по-прежнему широко используется данный шинный интерфейс. Однако самые современные материнские платы для старших моделей микропроцессоров уже не содержат слотов расширения в стандарте ISA.

### Ускоренный графический порт AGP

С ростом производительности вычислительных систем происходит дальнейшее расслоение шинных интерфейсов - так для управления видеоподсистемой современного компьютера используется специальный ускоренный графический порт - Accelerated Graphics Port (AGP), являющийся самостоятельной шиной обмена графической информацией.

Порт AGP имеет самостоятельный доступ к памяти и позволяет разгрузить шину PCI для обслуживания других быстродействующих устройств. Разрядность шины AGP составляет 32 бит, рабочая частота - 66 МГц, в шине поддерживаются режимы передачи х1, х2, х4. В этих режимах за один такт передаются соответственно одно, два или четыре 32-разрядных слова, а пиковая производительность в режиме х1 - 266 Мбайт/с, х2 - 533 Мбайт/с и х4 - 1066 Мбайт/с.

В общем случае существует несколько основных способов повышения скорости отображения графической информации:

* ускорение передачи данных из основной памяти, передача большего числа функций генерации изображения видеоконтроллеру, повышение тактовой частоты контроллера,
* увеличение разряд­ности локальной шины между видеоконтроллером и расположенной на видеокарте памятью,
* применение быстродействующей видеопамяти,
* а также увеличение объема этой памяти для кэширования данных.

### Интерфейс ввода-вывода на основе коммутатора

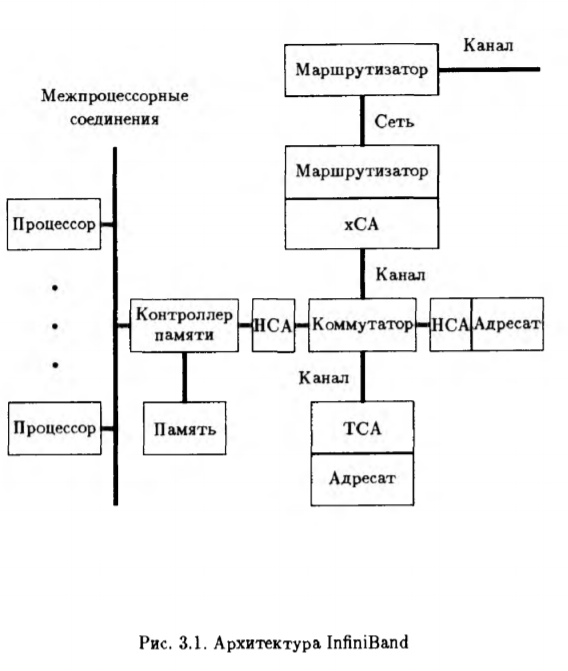
Наряду с шиной PCI-X появились две принципиально новые архитектуры локальных шин - **Future I/O (FIO)** и **Next Generation I/O (NGIO).**

В отличие от разделяемой архитектуры шины PCI, характеризующейся совместным использованием шины всеми подключенными устройствами, Future I /O опирается на коммутируемую структуру или матрицу с прямыми соединениями между устройствами. Как и в любой коммутируемой среде, подключение очередного устройства не сказывается на доступной любому из них пропускной способности, а наоборот ведет к увеличению совокупной пропускной способности, так как каждое из устройств имеет свое отдельное соединение. Первоначально пропускная способность соединений Future I/O будет составлять 1 Гбайт в каждом направлении. Предполагается, что Future I/O будет применяться для межсоединений процессоров в кластерах, подсистем и процессоров в сетях хранения, а также для подключения высокоскоростной периферии и сетей на базе Fibre Channel, TJltraS SCSI и Gigabit Ethernet.

Архитектура шины Next Generation I /O так же, как и Future I/O , основывается на коммутируемой технологии. Кроме того, NGIO использует канальную архитектуру, реализующую эффективный механизм ввода-вывода для обработки запросов от периферии. Канальная архитектура NGIO включает главный канальный адаптер - Host Channel Adapter (НСА), интерфейс с контроллером памяти вычислителя и целевые канальные адаптеры - Target Channel Adapter (ТСА). Главный канальный адаптер содержит механизмы прямого доступа к памяти. Целевые канальные адаптеры служат для подключения контроллеров ввода-вывода к коммутатору. НСА и ТСА могут подключаться либо к другу.

### Архитектура ввода-вывода InfiniBand

Стандарт InfiniBand описывает архитектуру и спецификации на передачу данных между процессорами и интеллектуальными устройствами ввода-вывода. Принципиальное отличие InfiniBand от PCI заключается в замене общей шины коммутаторами, что обеспечивает широкие возможности кластеризации и масштабирования вычислительных систем.



### Интеллектуальный ввод-вывод I2О

Залупа какая то я в душе не ебу че тут писать

## 3.3. Периферийные шины

**Периферийные шины** являются компонентами следующего уровня и располагаются между интерфейсными платами контроллеров и периферийными устройствами. Эти интерфейсы изменяются так же быстро, как и остальная часть вычислительной системы.

### Шина EIDE

Простейшей современной шиной *для подключения устройств долговременной памяти* является расширенный интерфейс подключения к компьютерам АТ накопителей со встроенным контроллером - АТ Attachment/Enhanced Integrated Drive Electronics (ATA/EIDE).

Эта шина имеет **два канала** (Primery и Secondary), к каждому из которых можно подключить по **два устройства** (Master и Slave).

Теоретическая пропускная способность составляет 16,7 Мбайт/с.

**Длина** интерфейсного кабеля до 45 см.

**Недостатком** является то, что значительную часть работы по управлению устройствами в стандарте EIDE выполняет процессор. Кроме того, для управления дисками большой емкости используются неэффективные методы.

Шина EIDE используется в основном в настольных персональных компьютерах.

### Семейство шин SCSI

С точки зрения максимальной теоретической пропускной способности и числа подключаемых периферийных устройств ведущее положение среди периферийных шин в настоящее время занимает интерфейс малых вычислительных систем - Small Computer Systems Interface (SCSI). Д

анный интерфейс допускает реализацию в контроллере шины (адаптере) до трех 8- или 16-разрядных каналов, называемых узкими (Narrow) или широкими (Wide) соответственно, и позволяет подключать до 7 или 15 устройств к каждому каналу в зависимости от его разрядности.

Имеется несколько спецификаций шины, совместимых снизу вверх, - то есть при поддержке контроллером и подключенным устройством разных модификаций стандарта использоваться будет менее скоростная.

Шина может находиться в одном из следующих состояний (фаз):

* свободна (BUS FREE);
* выбор (SELECTION);
* передача команд (COMMAND);
* передача статуса (STATUS);
* передача сообщения (MESSAGE);
* арбитраж (ARBITRATION);
* повторный выбор (RESELECTION).

Терминирование бывает двух типов: пассивное и активное.

### Шина IEEE-1394

IEEE-1394, FireWire и i.LINK - это три названия высокоскоростного интерфейса последовательной шины стандарта SCSI. Последовательный дешевый аналог параллельных шин.

Данный интерфейс обладает всеми достоинствами (преимуществами) действующего интерфейса SCSI и пропускной способностью 100-400 Мбит/с.

Кроме того, предусмотрены пути модернизации шины до гигабитных скоростей. Шина поддерживается фирмой Apple Computers и разработана на базе оригинальной шины FireWire, предложенной Apple Computers.

**Область применения** IEEEM-1394 - высокоскоростной доступ к устройствам хранения информации (жесткие диски, приводы CD и DVD), устройствам ввода данных (сканеры), аудио- и видеооборудованию.

Стандарт обладает гибкостью и простотой использования, обеспечивает приоритетную передачу, подключение и отключение устройств в ’’ горячем режиме” (без выключения питания и перезагрузки), способен конфигурироваться в автоматическом режиме.

### Шина Fibre Channel

**Fibre Channel** - комплект протоколов, определяющий высокую скорость передачи данных (до 1 Гбайт/с) между компьютерами, периферией и другими системами через волоконнооптический интерфейс. Является конкурентом SCSI.

Fibre Channel чаще всего используется в сетях хранения данных - Storage Area Network (SAN). Первоначально стандарт основывался на использовании только оптоволоконной среды передачи, однако в процессе разработки в спецификацию был включен и кабель с медными жилами.

Fibre Channel представляет собой высокопроизводительный последовательный канал ’’точка-точка” как самостоятельно работающий, так и позволяющий работать поверх него протоколам SCSI, FDDI (протокол построения локальных сетей) и другим.

В стандарте предусмотрены коммутаторы, обеспечивающие соединения ’’точка-многоточка”, что делает его пригодным для применения в локальных сетях высокой пропускной способности.

### Универсальная последовательная шина USB

**Universal Serial Bus (USB)** - универсальная последовательная шина - новый стандарт с пропускной способностью 12 Мбит/с для подключения низко- и среднескоростных устройств типа: клавиатуры, мыши, джойстика, микрофона, принтера, модема, дисковода CD-ROM и т.д.

Общее число подключаемых устройств - до 127. Поддерживается фирмами Intel, Microsoft Corp.

Данный интерфейс позволяет снять проблему недостаточного количества прерываний, каналов DMA, адресов ввода-вывода при подключении большого числа медленных устройств. В ближайшие годы по мере роста числа производимых периферийных устройств с портами USB произойдет полный переход на данный тип интерфейса в персональных компьютерах. При этом классические последовательные и параллельные порты, порты для подключения клавиатуры и мыши будут вытеснены.

Кроме того, с внедрением USB будет вытеснена и шина ввода-вывода ISA, через которую реализовывалось подключение низко- и среднескоростных устройств.

Архитектурой USB предусматривается топология **"звезда”** . Система должна состоять из одного ведущего (корневого) концентратора с контроллером, управляемым операционной системой, требуемого количества концентраторов и узлов (периферийных устройств). Концентраторы могут каскадироваться, образуя древовидную структуру с поддеревьями. Узлы подключаются к концентраторам. Всего узлов может быть 127. Концентратор также считается устройством.

При построении цепочек дерева наиболее скоростные устройства (монитор и т.п.) следует подключать ближе к корневому концентратору, а наименее скоростные (клавиатура и т.п.) - в конце цепочки. Это обеспечивает постепенное снижение интенсивности трафика от корня к концу цепочки и приоритетное обслуживание быстродействующих устройств.

# Подсистема памяти

## Архитектура многоуровневой памяти.

Главными требованиями, которым должны удовлетворять подсистема памяти, являются достаточно большая ёмкость, высокое быстродействие и экономическая эффективность с точки зрения технической реализации. При минимальных физических размерах память должна обладать как можно большей информационной ёмкостью.

Различные уровни иерархической памяти работают асинхронно. Для этого существуют буферные устройства, которые позволяют параллельно выполнять операции доступа к различным уровням.

Подсистема памяти имеет многоуровневую иерархическую структуру. На верхнем уровне – регистровая память, на следующем уровне – буферная память (кэш-память), затем оперативная память произвольного доступа, на самом нижнем – внешняя память.

Регистры общего назначения (РОН) – сверхоперативное запоминающее устройство (СОЗУ). Объём СОЗУ обычно составляет несколько десятков регистров. Регистровая память предназначена для хранения адресов, операндов и результатов выполнения операций.

Кэш-память – высокоскоростная память произвольного доступа, используемая процессором компьютера для временного хранения информации. Увеличивает производительность.

Внешняя память располагается на дисковых и ленточных запоминающих устройствах. Память самая медленная, но самая ёмкая.

Есть несколько уровней кэш-памяти (до трёх уровней). Первый (внутренний) расположен на кристалле процессора. Ёмкость колеблется от нескольких десятков килобайт до полутора мегабайт, время доступа – один-два тактов. Второй уровень реализуется либо на кристалле процессора, либо в виде отдельной микросхемы на одной плате, либо как статическая память. Ёмкость – от нескольких сотен килобайт до нескольких мегабайт, время доступа – три-пять тактов. Кэш третьего уровня имеет внешнее исполнение. Ёмкость до нескольких мегабайт, время доступа – до десятков тактов.

## Статическая и динамическая память

Ячейка динамической памяти представляет собой конденсатор на полупроводниковом кристалле. Его зарядка означает переход в состояние 1. Разрядка – в состояние 0. Одна из причин ограниченного быстродействия этого типа памяти – длительная зарядка и разрядка ёмкости, ограниченность во времени хранения заряда конденсатором. Чтобы избежать потери данных, необходимо восстановление в ней информации – недостаток динамической памяти. Достоинства – дешевизна и минимальное потребление мощности.

Статическая память строится на элементах памяти с двумя устойчивыми состояниями – триггерах. Бит отображается состоянием триггера – электронного ключа, состоящего из четырёх транзисторов и двух резисторов. Переход триггера из одного состояния в другое происходит быстрее, чем зарядка/разрядка. Минусы: потребляет большую мощность, чем динамическая память; значительно дороже и при одинаковой степени интеграции с динамической, обладает меньшей информационной ёмкостью.

## Кэш-память

Кэш используется для обмена между процессором и памятью, а также между оперативной памятью и внешним накопителем. В основе работы кэш-памяти лежит принцип временной и пространственной локальности программ. Принцип пространственной локальности предполагает считывание в кэш нескольких соседних ячеек памяти (блока инф-ии). Каждый блок хранится в строке буфера, набор таких строк составляет кэш-память.

Чем больше размер блока, тем выше коэффициент удачных обращений. С другой стороны, чем больше размер блока, тем меньше их помещается в кэше – растёт число операций пересылки из памяти в кэш и обратно.

Информация из основной памяти загружается в кэш блоками по несколько слов и временно хранится в нём.

# Классификация типов кэша

Различают классификации по способам хранения данных и команд, организации записи данных и организации отображения памяти. Для хранения данных и команд применяют общие и раздельные кэши. Раздельный используется в кэш-памяти первого уровня. Общие кэши – на втором и третьем уровнях иерархической памяти.

Существует два способа обновления информации, гарантирующих адекватность содержимого кэша и основной памяти: сквозная запись и обратная запись. Сквозная запись: информация записывается как в кэш, так и в оперативную память. Обратная запись: сохраняет блок кэша в оперативной памяти только при вытеснении его из кэша и при наличии в нём изменений.

В зависимости от способа отображения различают типы кэшей:

* С прямым отображением
* Полностью ассоциативный
* Множественный ассоциативный

## Кэш с прямым отображением

Является самым простым типом буферной памяти. Кэш состоит из строк, содержащих информационную и признаковую части. Обычно размер блока составляет 32 или 64 байта.

При обращении центрального процессора к адресному объекту, выполняется:

1. Из адреса объекта выделяются три поля:

- номер байта в блоке;

- индекс, он же номер блока в кэше, в котором должен находится адресуемый объект;

- признак, позволяющий отличить один блок оперативной памяти от другого;

1. По индексу, выделенному из адреса объекта, находится строка кэше, в которой должен находиться нужный блок памяти;
2. Признак, выделенный из адреса объекта, сравнивается с признаком, хранящимся в элементе признаковой памяти найденной строки кэша.

Достоинства: необходимость только одного сравнения признака, выбранного из строки и признака, выделенного из адреса объекта. Недостатки: конфликт адресов блоков памяти, отображаемых на одну строку кэша. При возникновении конфликта адресов, старый блок переписывается из кэша в память, а на его место помещается новый.

## Полностью ассоциативный кэш

Любой блок оперативной памяти может быть отображен на любую строку кэша. Так как между блоками нет определённых взаимосвязей, в строку кэша должен записываться полный адрес объекта и сам блок. Этот подход позволяет решить проблему конфликта адресов.

Пока кэш не заполнен, блок помещается в первую свободную строку. Когда буфер полон, один из блоков должен быть из него вытеснен и переписан в основную память, а на его место помещён новый.

## Множественный ассоциативный кэш

Множество строк разбиваются на группы по А≥1 строк. А целочисленное – коэффициент ассоциативности кэша. А = 2 – двухвходовый множественный ассоциативный кэш, А = 4 – четырёхвходовый и т д.

Индекс адресует номер группы и адресуемый блок может располагаться в любой строке группы. Между группами множественный ассоциативный кэш является кэшем прямого отображения, а внутри группы – полностью ассоциативным.

С ростом ассоциативности, с одной стороны, растёт частота «попаданий» в кэш, а с другой – растут и накладные расходы в виде количества сравнений признаков.

При А = 1 множественный ассоциативный кэш превращается в кэш прямого отображения, при А = числу строк – в полностью ассоциативный кэш.

# Принципы организации оперативной памяти

## Элемент динамической памяти

В состав ячейки динамической памяти входит конденсатор и транзистор.

Вот чёто вроде надо написать, а чё, я не ебу

## Массивы ячеек и структура микросхем динамической памяти

Микросхемы динамической памяти организованы в виде массивов строк и столбцов. На все ячейки одной строки приходится общая числовая шина, а на все ячейки одного столбца отводится общая пара разрядных шин, усилитель и буфер обмена. Место каждой ячейки определяется адресом строки и столбца.

Микросхема DRAM имеет матричную организацию, каждый элемент которой хранит бит данных и адресуется с помощью стробирующих сигналов адреса строки – RAS и адреса столбца – CAS. Сигналы RAS и CAS используются также для восстановления данных. Продолжительность процесса восстановления определяется количеством строк матрицы. Соотношение количества строк и столбцов информационного поля DRAM влияет на энергопотребление микросхемы, способ адресации и совместимость с микросхемами различных поколений. Может ещё дописать, нахрена только это

## Многоблочная структура памяти и расслоение адресов

Для ускорения выполнения операций чтения и записи группы последовательных байт оперативная память подразделяется на множество независимых модулей (блоков). Это разбиение позволяет повысить эквивалентное быстродействие памяти с помощью расслоения адресов по независимым блокам.

# Обзор технологий FPM, EDO, SDRAM, DR DRAM, DDR DRAM

FPM DRAM – технология адресации, при которой выбор данных в пределах одной страницы (строки) выполняется с указанием полного адреса страницы и столбца только в момент считывания первого байта, а последующие байты той же страницы считываются без указания её адреса. Сигнал CAS используется для адресации различных ячеек в пределах строки и для задания конца периода доступности адреса, в течение которого выполняется считывание данных.

EDO DRAM – тип памяти с расширенным временем удержания данных выходе за счёт применения элементов конвейеризации. EDO – технология адресации, при которой в качестве сигнала конца операции чтения используется незанятый сигнал OE. Это даёт возможность начать адресацию нового столбца до завершения чтения данных из предыдущего.

SDRAM – технология адресации последовательности слов в пакете, при которой после выборки первого слова не используется адрес текущего столбца. Особенностью этой технологии является зависимость всех управляющих сигналов от общего системного тактового сигнала.

DR DRAM – (новый) тип памяти, основанный на шине Direct Rambus, в которой управление адресацией отделено от работы с данными.

DDR DRAM – тип памяти. Который конкурирует с памятью DR DRAM. Плюсом данной памяти является сохранение в её основе технологии SDRAM, в рамках которой за такт передаётся два пакета данных. Используется не только в оперативной памяти. Но и в высокопроизводительных графических платах.

# Типы модулей памяти

SIMM – модуль с односторонним расположением контактов, имеет исполнение двух видов: 30-контактные 8-разрядные и 72-контактные 32-разрядные.

DIMM – модуль с двухсторонним расположением контактов – 168-контактный и 64-разрядный модуль. Синхронная память SDRAM выполняется только в конструктиве DIMM. Модули оперативной памяти имеют три испольнения: обычные, с простой четностью и с коррекцией ошибок.

# Банки памяти

Оперативная память с точки зрения процессора – совокупность нескольких отдельно адресуемых банков в виде набора SIMM или DIMM модулей памяти, общая разрядность которых соответствует максимальной разрядности шины данных процессора. Один банк памяти в компьютерах на базе процессора Pentium, состоит из двух 32-разрядных SIMM модулей или одного 64-битного DIMM.

В целях минимизации нагрузки и повышения силы выходного тока производится согласование шин данных с помощью специальных буферов-микросхем, устанавливаемых на плате SIMM или DIMM. Модуль с буфером называется buffered, а небуферизованный – unbuffered. Несмотря на преимущество буферизованных модулей памяти, их использование ограничено: необходимо чтобы данный тип модулей памяти поддерживался микропроцессорным набором вычислительной платформы.

При выборе модулей памяти, руководствуются правилом: модуль памяти с большим числом микросхем малой ёмкости нагружает шины сильнее, чем модуль с меньшим числом микросхем большей ёмкости.

# Механизм динамического преобразования адресов:

Реально существующую основную память называют физической, а её адреса – физическими, логическую память – виртуальной, а её адреса – виртуальными.

Обращение к таблице преобразования адресов при преобразовании виртуального адреса в реальный занимает много времени. Для ускорения этой процедуры в вычислительных системах имеется механизм динамического преобразования адресов, основанный на кэш-памяти ассоциативного типа. В эту кэш-память записываются номера наиболее часто используемых страниц и номера блоков, соответствующих этим страницам в основной памяти. При преобразовании адресов сначала проверяется ассоциативный кэш, в случае промаха – основная таблица преобразования, расположенная в оперативной памяти.

# Дисковые запоминающиеся устройства

## Факторы, определяющие производительность

Конструкция головок чтения-записи, запоминающая среда, каналы обмена, скорость вращения, количество дисковых пластин.

Время, необходимое дисковому накопителю для обслуживания запроса пользователя, имеет четыре составляющие: время обработки команды, время поиска дорожки, задержка вращения, время передачи данных.

Время обработки команды зависит от интерфейса диска, типа команды и возможности использования буфера или кэша жёсткого диска.

Важным фактором, определяющим время поиска дорожки, является размер и вес компонентов диска.

Задержка вращения определяется временем ожидания заданного сектора подл магнитными головками.

Время передачи данных зависит от скорости передачи и размера передаваемого блока.

## Влияние плотности записи битов

Плотность битов (линейная плотность) показывает, сколько битов можно сохранить на одной дорожке. От этого зависит число секторов на дорожке. Чем выше линейная плотность, тем больше секторов на дорожке.

Увеличение числа битов на дюйм оказывает влияние на следующие рабочие характеристики диска:

* Возрастает скорость передачи носителя
* Возникают ограничения на скорость вращения
* Уменьшается число переключений головки при переходе на следующую дорожку данного цилиндра и на дорожку следующего цилиндра
* Растёт размер цилиндра, выраженный в количестве секторов, так как увеличивается число секторов на дорожке

## Влияние плотности размещения дорожек

При одинаковом числе секторов на дорожке и одинаковом количестве дорожек в цилиндре более высокая плотность дорожек означает короткую дистанцию физического перемещения головки от одного блока к другому и, следовательно, обеспечивает меньшее время поиска. Для более узких дорожек с более плотным расположением требуется большее время позиционирования головки.

Повышение плотности записи даёт возможность получать диск заданной ёмкости с меньшим числом пластин и меньшим числом головок. Фактор роста плотности записи позволяет снизить стоимость дискового устройства и приводит к снижению установившейся скорости передачи данных размера цилиндров.

# Средства обеспечения отказоустойчивости и масштабируемости

## Отказоустойчивые массивы дисков (RAID)

На уровне дисковой подсистемы предпринимаются технические меры, направленные на снижение потерь до стоимости вышедшего из строя оборудования. В основе таких мер лежит резервирование и избыточность, реализуемые в виде массива независимых резервных дисков (RAID). Существует программная и аппаратная реализация RAID-массива.

RAID-система состоит из управляющей программы, или контроллера, и массива НЖМД. RAID-массив строится на основе распределения данных между отдельными дисками. Пространство каждого диска разбивается на сегменты, размер которых может составлять от одного сектора (512 байт) до нескольких мегабайт. Различные логические формы организации RAID-массива называются уровнями RAID.

Повышение уровня готовности и отказоустойчивости достигается с помощью ряда аппаратных средств:

* Средств изменения конфигурации без отключения сервера
* Возможности наращивания ёмкости системы в процессе работы
* Возможности изменения «на ходу» уровня RAID
* Устройств «горячей замены», горячего резерва или избыточных дисков, контроллеров, блоков питания, вентиляторов охлаждения
* Средств резервного питания кэш-памяти контроллера RAID-массива.

Высокая производительность обеспечивается за счёт кэширования.

## Уровни RAID

Существует девять уровней (типов) RAID-массивов, различающихся по скорости, надёжности и стоимости изготовления: 0, 1, 2, 3, 4, 5, 7, 10, 53.

RAID 0 – дисковый массив без дополнительной отказоустойчивости.

Достоинства:

* Высокая производительность за счёт распределения операций ввода-вывода между всеми дисками массива
* Не производится подсчёт контрольных сумм, что увеличивает производительность
* Простота конструкции и лёгкость в изготовлении

Недостатки:

* Выход из строя одного из дисков приводит к потере всех данных, хранящихся в дисковой подсистеме

RAID 1 – дисковый массив с зеркалированием данных.

Достоинства:

* Скорость записи та же, что и для одного диска
* Скорость чтения в два раза выше, чем для одного диска
* Высокая скорость восстановления данных из-за их стопроцентной избыточности
* Самый простой конструктив из всех типов RAID-массивов
* Единственный тип RAID-массива, позволяющий получить отказоустойчивую дисковую подсистему на двух дисках

Недостатки:

* Низкий коэффициент использования дискового пространства, высокая стоимость.

RAID 2 – дисковый массив с использованием алгоритма Хэмминга для проверки/восстановления данных.

Достоинства:

* Исправление ошибок данных «на лету»
* Высокая скорость передачи данных, увеличивающаяся с ростом количества дисков в массиве
* Коэффициент использования дискового пространства увеличивается с ростом количества дисков в массиве
* Относительно простой конструктив контроллера

Недостатки:

* Очень низкий коэффициент использования дискового пространства в случае малого размера слова данных

RAID 3 – дисковый массив с вычислением контрольной суммы параллельно с передачей данных.

Достоинства:

* Очень высокая скорость чтения и записи данных
* Выход из строя одного диска незначительно влияет на общую производительность массива
* Высокий коэффициент использования дискового пространства

Недостатки:

* Контроллер имеет конструктив средней сложности
* Высокая трудоёмкость программной реализации, требующая значительной вычислительной мощности

RAID 4 – дисковый массив с независимыми дисками данных и общим диском для хранения контрольных сумм.

Достоинства:

* Высокая скорость чтения данных
* Высокий коэффициент использования дискового пространства

Недостатки:

* Наименьшая из всех типов RAID-массивов скорость записи
* Достаточно сложный конструктив контроллера
* Сложный и неэффективный алгоритм восстановления данных при выходе из строя одного из дисков

RAID 5 – дисковый массив с независимыми дисками данных и равномерным распределением контрольных сумм между дисками.

Достоинства:

* Высокая скорость чтения и записи данных
* Высокий коэффициент использования дискового пространства

Недостатки:

* Выход из строя дисковода оказывает заметное влияние на производительность
* Сложный конструктив контроллера
* Сложный алгоритм восстановления данных в случае выхода из строя одного из дисков

RAID 6 – дисковый массив с независимыми дисками данных и двумя независимыми схемами контрольных сумм, распределёнными между дисками.

Достоинства:

* Высокая скорость чтения данных
* Высокий отказоустойчивость

Недостатки:

* Сложный конструктив контроллера
* Большая нагрузка на контроллер при вычислении контрольных сумм и адресов, по которым они должны быть размещены на дисках
* Очень малая скорость записи
* Низкий коэффициент использования дискового пространства

RAID 7 – дисковый массив с асинхронным вводом-выводом и высокой скоростью передачи данных.

Все операции ввода-вывода проводятся в асинхронном режиме, то есть независимо от выполнения других.

Достоинства:

* Общая производительность на операциях записи на 25-90% выше, чем для одного диска, и в 1,5-6 раз лучше, чем у RAID-массивов других типов
* Очень высокая скорость доступа к данным в многопользовательской среде, реализуемая за счёт кэширования
* Скорость чтения и записи растёт при увеличении числа дисков в массиве

Недостатки:

* Высокая стоимость

RAID 10 – комбинация технологий RAID 1 и RAID 0 в одном дисковом массиве.

Достоинства:

* Имеет такую же отказоустойчивость, как и RAID 1
* Скорость чтения и записи выше, чем у RAID 1

Недостатки:

* Высокая стоимость
* Самый низкий коэффициент использования дискового пространства из всех типов RAID-массивов

RAID 53 – комбинация технологий RAID 0 и RAID 3 в одном дисковом массиве.

Достоинства:

* имеет такую же отказоустойчивость, как и RAID 3
* Скорость чтения и записи выше, чем у RAID 3

Недостатки:

* Высокая стоимость
* Низкий коэффициент использования дискового пространства из всех типов RAID-массивов

Централизованные и локальные RAID-массивы

RAID-системы разделяют на:

* Системы с централизованным управлением (host-based) – вся интеллектуальная начинка RAID-массива размещается на плате, устанавливаемой в сервер.
* Локальные системы (SCSI-to-SCSI) – управляющая часть расположена в RAID-стойках, подключаемых к серверу через SCSI-контроллер.

# Кластеры

Кластеризация вычислительных систем – технология, с помощью которой два и более серверов функционируют как один вычислитель, воспринимаемый и клиентскими приложениями, и пользователями как единая вычислительная система.

Используется для обеспечения:

* Отказоустойчивости
* Постоянной доступности
* Масштабируемости, достигаемой реализацией распределения вычислительной нагрузки

Существует три модели кластерных систем:

* Модель с разделяемой памятью
* Модель с разделяемыми дисками
* Модель без разделяемых ресурсов

## 6.1 Модели состоятельности памяти.

При увеличении процессорных мощностей встает вопрос о построении памяти, имеющей единое адресное пространство и доступ к объектам через операции чтения и записи. При этом требуется извещать процессор об изменениях в памяти. Рассмотрим наиболее используемые модели состоятельности памяти, которые решают данные вопросы:

1. Строгая – каждая операция чтения возвращает последнее записанное значение
2. Последовательная – все процессоры в системе соблюдают один и тот же порядок выполнения записи и чтения. *Процессор, выполняющий запись, приостанавливается до получения подтверждений об объявлении несостоятельными всех копий модифицируемых данных или о модификации этих копий.*
3. Процессорная – наблюдаемый в двух процессорах порядок выполнения операций чтения-записи может не совпадать, но порядок выполнения записей, производимых каждым процессором, должен быть одним и тем же.
4. Слабая - вводит разграничение между обычным и синхронизованным доступом в память при выполнении состоятельности только для обращений к памяти в точках синхронизации различных уровней памяти. Разрешается несогласованность данных вне точек синхронизации.
5. Свободная – уточняет модель слабой состоятельности и требует состоятельности только между парой операций синхронизации *(acquire – release),* которые открывают интервал и обеспечивают исключительный доступ процессора к разделяемым данным и закрывают его и разрешают всем процессорам доступ к разделяемым данным. Доступны следующие модификации:
   1. Неторопливо-свободная – модификация разделяемой памяти откладывается до момента непосредственного обращения к ней.
   2. Нетерпеливо-свободная – модель с передачей изменений разделимой памяти при выполнении release, независимо от того, когда изменения будут востребованы.
   3. Интервально-свободная- модель с разбиением на локальные и глобальные интервалы. Гарантирует состоятельность разделяемых данных только в конце глобального интервала
   4. Последовательно-свободная – модель, требующая доступа к разделяемым данным, защищённым синхропримитивами.

## 6.2 Неявная(аппаратная) когерентность.

Применяются аппаратные средства для обеспечения когерентности данных в памяти, например, для кэшей с помощью межузловых пересылок.

## 6.2.1 Сосредоточенная память.

Для систем с сосредоточенной памятью в симметричных многопроцессорных архитектурах применяется алгоритм поддержки когерентности кэшей MESI. Он обеспечивает когерентность кэш-памяти с обратной записью и минимизирует пересылки данных между основной памятью и кэшем.   
Алгоритм предполагает, что вычислитель имеет общую разделяемую память, в каждом процессоре содержится локальная кэш память, а объединены они между собой посредством шины.

(*В современных компьютерах доступ к памяти могут одновременно иметь несколько независимых процессоров (ядер, трэдов). Каждый из них имеет свои приватные кэши, в которых хранятся копии необходимых линий, а некоторые из них при этом локально модифицированы. Встает вопрос, а что, если одна и та же линия одновременно понадобится нескольким процессорам. Не сложно сделать вывод, что для корректной работы системы необходимо обеспечить единое пространство памяти для всех процессоров.  
Для обеспечения этого были придуманы специальные протоколы когерентности. Когерентность кэша — свойства кэш-памяти, означающее целостность данных, хранящихся в локальных кэшах, разделяемой системы. Каждая ячейка кэша имеет флаги, описывающие, как ее состояние соотносится с состоянием ячейки с таким же адресом в других процессорах системы.  
При изменении состояния текущей ячейки необходимо каким-то образом сообщить об этом остальным кэшам. Например, генерируя широковещательных сообщения, доставляемые по внутренней сети многопроцессорной системы.  
Было придумано множество протоколов когерентности, отличающиеся алгоритмами, количеством состояний и, как следствие скоростью работы и масштабируемостью. Большинство современных протоколов когерентности представляют вариации протокола MESI*)

В данной схеме каждая линию кэша может находиться в одном из четырех состояний:  
**М-строка модифицирована** (*Таким флагом может быть помечена только линия в одном кэше. Данное состояние означает, что данная линия была изменена, но до памяти эти изменения еще не дошли. Хозяин такой линии может спокойно читать и писать в нее без опроса остальных.)***E-строка монопольно копированная** (*Обозначенная таким флагом линия, так же, как и M-линия может находиться только в одном кэше. Содержащиеся в ней данные полностью идентичны данным в оперативной памяти. Записывать и читать из нее можно без внешних запросов, так как она хранится только в одном кэше. После записи такая строка должна быть помечена, как модифицированная.)***S – строка множественно копированная или разделяемая.** (Линия *может одновременно содержаться в кэшах нескольких устройств и использоваться совместно. Запросы на запись в такую линию всегда идут на общую шину, что привод к тому, что все линии с таким адресов в остальных кэшах помечаются как недействительные. При этом содержание основной памяти также обновляется. Чтение же из такой линии, не требует никаких внешних запросов.)*  
**I – строка невозможная к использованию.** (*Такая линия считается не валидной, и попытка прочитать приведет к кэш-промаху. Линия помечается недействительной в случае, если она пусти или содержит устаревшую информацию.)*

## 6.2.2 Распределённая память.

ВС с памятью, физически распределенной по узлам(модулям), имеющей единое пространство адресов применяют алгоритмы реализации когерентности.

*Например, ПРОСТЕЙШИЙ заключается в том, что при промахе в кэш в любом процессоре инициируется запрос требуемой строки из резидентного (того, где строка размещена) блока. Запрос передается в узел с резидентным блоком памяти, откуда строка пересылается в узел, где произошёл промах. Так обеспечивается начальное заполнение кэшей.*  
*Для обеспечения когерентности после выполнения записи процессор приостанавливается до завершения этой последовательности действий:  
-измененная строка пересылается в резидентный блок  
- разделяемая строка пересылается из резидентной памяти во все узлы, перечисленные в списке разделяющих эту строку;  
- после получения подтверждений о том, что все копии заменены, резидентный узел пересылает разрешение продолжить вычисления в процессор, приостановленный после записи  
Недостатком является время простоя, DASH лучше.*

Алгоритм DASH. Каждый блок памяти имеет для каждой строки, резидентной в узле, список узлов, в кэшах которых размещены копии строк. Каждая резидентная строка может находиться в трех глобальных состояниях:

1. Некэшированнная – копия строки не находится в кэше узлов, кроме, возможно резидентного для этой строки
2. Удаленно-разделенная – копии строки размещены в кэшах других узлов.
3. Удаленно-измененная – строка изменена операцией записи в каком-либо узле

Также строка кэша может находиться в одном из 3 локальных состояний:

1. Невозможная к использованию
2. Разделяемая - есть неизменная копия, которая может размещаться в других кэшах
3. Измененная – если копия изменена операцией записи.

*Реализация алгоритма когерентности с большим быстродействием возможна за счет учета специфики параллельных про­ грамм, в которых асинхронно используются одни и те же переменные на каждом временном интервале только одним процессором с последующим переносом обработки на другой процессор. Такой тип вычислений позволяет организовать более эффективные схемы передачи необходимых строк из кэша одного процессора в кэш другого.*

## 6.3 Явная когерентность(программная).

Основная задача программиста- эффективное программирование параллельных процессов, совмещающих вычисления и передачу данных между узлами и минимизирующих объем передаваемой информации. Тк память в узлах является могоуровневой, то очевидно, что получаемые по коммуникационному интерфейсу строки данных делают несостоятельными копии этих строк в кэш памяти. Поэтому нужно предусмотреть организацию когерентности прибывшей и кэшированной строк.

Различают следующие варианты поддержки когерентности:  
- при наличии дубликатов признаков (тегов) строк кэш-памяти в контроллере прямого доступа делать по получении строки несостоятельной только действительно необходимую строку;  
- при отсутствии признаковой памяти в контроллере делать по каждому приему строки несостоятельными все строки кэш-памяти.  
Применение явной когерентности в создаваемых вычислительных системах обусловлено либо слишком большим временем, либо недопустимо высокими аппаратными затратами на реализацию неявного механизма когерентности.

## Механизм когерентности на основе масштабируемого когерентного интерфейса SCI.

SCI принял в качестве стандарта ANSI. Реализация предусматривает дополнительный уровень кэш-памяти, размещенный в узле вычислительной системы. Поиск данный производится при промахе на предыдущих уровнях памяти и если данные присутствуют, то они и идут в верхние уровни памяти. Если данные в узле модифицируются, то по окончании также идут в верхние уровни. Если в узле нет данных, срабатывает промах. Стандарт SCI строит выч. системы с быстродействием более 1гбит/с.

## Коммуникационная среда MYRINET

Среда Myrinet стандартизует формат пакета, способ адресации вычислительных машин (ВМ), протокол передачи пакетов. Коммуникационная среда образуется адаптерами ’’шина компьютера - линк сети” и коммутаторами линков сети. Каждый ЛИНК содержит две однонаправленных линии и образует канал с общей пропускной способностью равной 80 Мбайт/с. Согласно требованиям, сеть Myrinet не должна блокироваться из-за неисправностей или обесточенных устройств. Если линк подключен к неработоспособному адаптеру, то передача не блокируется, но передаваемые пакеты теряются.

## Средства распараллеливания для систем с общим полем памяти (стандарт ОреnМР).

До появления OpenMP программы автоматически распараллеливались компилятором, используя ручную (или же явную) организацию параллельных задач с помощью обращений к специальной библиотеке примитивов. Стандарт же OpenMP является промежуточным подходом к распараллеливанию программ с применением “директив пользователя” и распараллеливающего по подсказкам пользователя компилятора. “Директивы пользователя” указывают компилятору на параллелизм

## Средства распараллеливания для систем с распределенной памятью (стандарт MPI)

Интерфейс передачи сообщений MPI включает в себя понятия процесса, группы процессов и коммуникатора. Процесс-исполнение программы на одном процессоре. Коммутатор является средой, для взаимодействия процессоров и обеспечивает обмен данными между ними и синхронизацию. Также есть внутригрупповые и межгрупповые коммуникаторы.

# 7. Принципы оценки производительности вычислителей

7.1. Цели исследований и показатели производительности

**ЦЕЛИ:**

* оценка с целью выбора ВС среди альтернатив для приобретения
* оценка производительности разрабатываемой ВС или ее компонент (планирование производительности или проектирование с заданной производительностью)
* контроль производительности (сбор и накопление данных о характеристиках ВС для прогноза влияния планируемых изменений конфигурации и ОС на производительность ВС)

В общем случае для оценки производительности необходимо предсказать характер прикладных задач, решаемых на ВС, и нагрузку на ВС.

**ПОКАЗАТЕЛИ ПРОИЗВОДИТЕЛЬНОСТИ:**

1. цикл выполнения задания в пакетных системах (время от момента поступления задания в ВС до его выполнения и возвращения результатов пользователю);
2. время ответа для интерактивных систем (время от момента ввода данных с клавиатуры до момента начала ответа);
3. дисперсию времени ответа, являющуюся мерой предсказуемости времени ответа в диалоговых системах; - время реакции в системах реального времени (время от момента ввода данных до момента выделения кванта на обслуживание запроса);
4. пропускную способность (число обслуженных заданий, запросов, транзакций в единицу времени);
5. загрузку ресурсов ВС.

Для **оценки** данных показателей используются следующие методы измерений:

1. Измерение элементарных времен, позволяющее оценить техническую производительность компонент и узлов вычислителя;
2. Измерение времени выполнения смеси команд, характерной для применения данной вычислительной установки;
3. Время выполнения образцовой программы, типичной для применения данной ВС (метод часто используется при построении компиляторов для выбора наиболее эффективного кода языковых конструкций программы);
4. Время выполнения измерительных программ, типичных для решаемого на ВС класса задач (обычно данный метод применяется при намерении сменить ВС или ОС для эксплуатации готового программного обеспечения);
5. Время выполнения синтетических программ, совмещающих в себе черты образцовых и измерительных программ;
6. Аналитическое моделирование процессов, протекающих в ВС. Аналитические модели позволяют быстро и наиболее наглядно увидеть достоинства и недостатки ВС, однако формализация вычислительных процессов всегда предусматривает их идегилизацию, после чего возникает вопрос адекватности аналитической модели реальной системе и проблема интерпретации полученных результатов;
7. Имитационное моделирование ВС, позволяющее исследовать трудноформализуемые при аналитическом моделировании элементы вычислительного процесса.

7.2. Пиковая и реальная производительность

**Пиковая** производительность определяет теоретический максимум быстродействия вычислительной системы в идеальных условиях.

Пиковая производительность определяется числом операций, выполняемых всеми исполнительными устройствами вычислителя, и равна произведению тактовой частоты, числа арифметическо-логических устройств процессора и количества процессоров вычислителя.

Измеряется в миллионах операций в секунду MIPS или миллионах операций с плавающей точкой в секунду MFLOPS.

**Реальная** производительность определяется классом решаемых задач. Время выполнения каждой из задач - основа для расчета индекса производительности данной вычислительной системы.

7.3. Тесты производительности

Три проблемы, связанные с анализом результатов тестов производительности:

* проблема достоверности оценок,
* проблема адекватности оценок (выбора тестов, наиболее точно характеризующих производительность при обработке типовых задач),
* проблема интерпретации (правильного истолкования результатов тестирования).

**ГРУППЫ ТЕСТОВ:**

Тесты можно разбить на **три группы**:

**Тесты производителей** средств вычислительной техники, используемые для оценивания качества собственных про­дуктов. Например, индекс ICOMP, индекс оценивает фактически производительность микропроцессора, а не вычислительной установки.

Стандартные **тесты, создаваемые независимыми аналитиками** и используемые для сравнения широкого спектра компьютеров. Например, тестовый пакет Linpack, тестовые наборы SPEC XX, тестовые пакеты ТРС, тестовый комплекс WebStone.

**Пользовательские тесты**, учитывающие специфику конкретного применения вычислительной системы. Данные тесты создаются крупными компаниями и используются для выбора средств вычислительной техники и программного обеспечения под определенные прикладные задачи. К этому классу можно отнести комплекс тестов NAS для оценки производительности многопроцессорных суперкомпьютеров с МРР-архитектурой.

**ТЕСТЫ:**

Комплекс **тестов NAS** общепризнанно считается лучшим тестовым набором для оценки многопроцессорных вычислительных систем с массовым параллелизмом. Тесты позволяют оценить вычислительные возможности компьютерной системы и скорость передачи данных между процессорами в массово-параллельных системах. Метрикой теста является относительная производительность по сравнению с показателями традиционного векторного суперкомпьютера

**Тесты Linpack** представляют собой совокупность программ решения задач линейной алгебры. Параметрами тестов являются порядок матрицы, формат значений элементов матрицы, способ компиляции, возможность применения оптимизированной библиотеки стандартных функций.

**Пакет SPEC 89**. Методика оценки производительности SPEC89 предполагает формирование десяти оценок SPECratioi, каждая из которых определяется как отношение времени выполнения i-й программы на тестируемом компьютере ко времени выполнения той же программы на вычислителе DEC VAX 11/780. SPECmark является средним геометрическим всех десяти частных оценок SPECratio.

К параметру SPECmark добавляются еще две оценки - SPECint89 и SPECfp89, раздельно характеризующие быстродействие компьютера при обработке целочисленных данных и вещественных чисел.

**Пакет тестовых программ SPEC 92** расширяет набор тестируемых функций по сравнению со SPEC89.

Также существуют **тестовые программы SPEC 95**.

Комплект процессорных **тестов SPEC2000** предназначен для тестирования процессора, иерархической памяти, компиляторов

**Пакет тестовых программ ТРС** применяется для оценки производительности при работе с базами данных.

Тесты ТРС включают набор тестов для измерения эффективности функционирования в различных режимах обработки данных: ТРС-А, ТРС-В, ТРС-С, TPC-D, TPC-W.

В основе тестового набора ТРС лежат следующие фундаментальные принципы:

* производительность соотносится с общей стоимостью тестируемой системы, включая аппаратную составляющую, программное обеспечение и эксплуатационные издержки в расчете на заданный срок эксплуатации;
* тест формулируется на высоком функциональном уровне без связи с конкретной программно-аппаратной платформой, допуская тем самым сравнение между собой различных реализаций;
* тест должен учитывать потенциальный рост мощности вычислительных систем и допускать масштабирование нагрузки по числу пользователей и объему базы данных.

*Тест ТРС-А используется для оценки эффективности исполнения транзакций типа ’’дебет-кредит”.*

*Тест ТРС-В аналогичен тесту ТРС-А, но ориентирован на пакетный режим обработки и имеет другую методику подсчета стоимости ВС.*

*Тест ТРС-С является основным продуктом ТРС и предназначен для оценки эффективности ВС при работе в режиме оперативной обработки транзакций*

*Тест TPC-D используется для оценки эффективности сложных информационных систем поддержки принятия решений с многонаправленными соединениями, сортировками и агрегированием данных отношений.*

*Тест TPC-W предназначен для оценки систем поддержки электронной коммерции, электронного бизнеса, коммерческих Web-приложений, корпоративных сетей.*

**Тест WebStone** используется для оценки конфигураций Web. Тест WebStone позволяет дать объективную оценку аппаратуры, программного обеспечения и дисциплины взаимодействия с сетью. Данный тест отражает специфику работы в глобальной сети с многократными переключениями, коррекцией ошибок, переадресациями и т.д. Основными метриками теста WebStone являются пропускная способность и время выполнения запроса (задержка), которые усредняются по множеству измерений за сеанс тестирования.