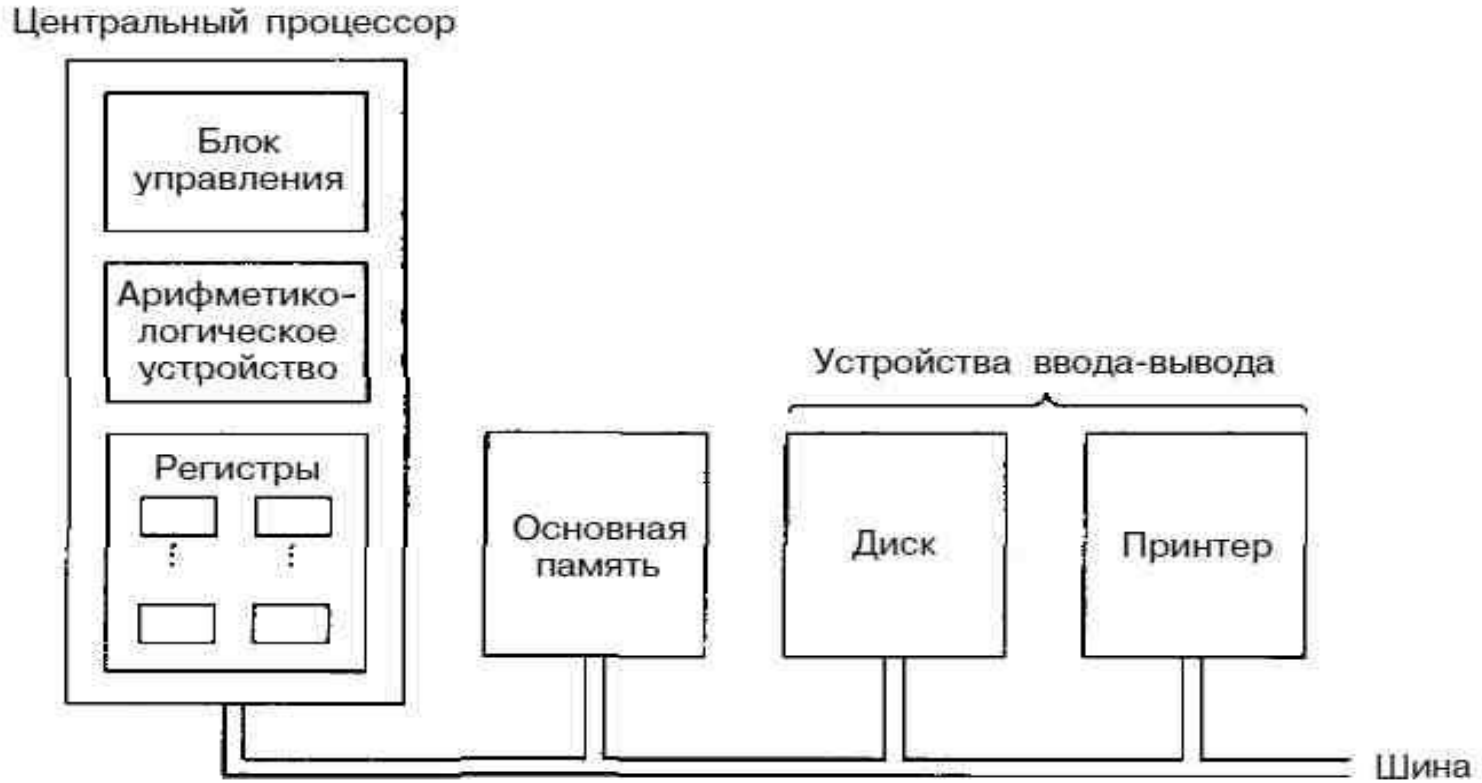
Аппаратная часть компьютерной системы

Схема простого компьютера

Концептуально простой персональный компьютер можно представить в виде модели, аналогичной изображенной на рисунке. Центральный процессор, память и устройства ввода-вывода соединены системной шиной, по которой они обмениваются информацией друг с другом. Современные персональные компьютеры имеют более сложную структуру и используют несколько шин, которые мы рассмотрим чуть позже.



Центральный процессор

Центральный процессор выбирает команды из памяти и выполняет их. Обычный цикл работы центрального процессора выглядит так: выборка из памяти первой команды, ее декодирование для определения ее типа и операндов, выполнение этой команды, а затем выборка, декодирование и выполнение последующих команд. Этот цикл повторяется до тех пор, пока не закончится программа.

Для каждого типа центрального процессора существует определенный набор команд, которые он может выполнять. Поэтому x86 не может выполнять программы, написанные для ARM-процессоров, а те, в свою очередь, не в состоянии выполнять программы, написанные для x86. Поскольку доступ к памяти для получения команды или данных занимает намного больше времени, чем выполнение команды, у всех центральных процессоров есть несколько собственных регистров для хранения основных переменных и промежуточных результатов. Соответственно набор команд содержит, как правило, команды на загрузку слова из памяти в регистр и на запоминание слова из регистра в память. Другие команды объединяют два операнда из регистров, памяти или обоих этих мест для получения результата — например, складывают два слова и сохраняют результат в регистре или в памяти.

В дополнение к регистрам общего назначения, которые обычно применяются для хранения переменных и промежуточных результатов, у многих процессоров есть ряд специальных регистров, доступных программисту. Один из этих регистров, называемый **счетчиком команд**, содержит адрес ячейки памяти со следующей выбираемой командой. После выборки этой команды счетчик команд обновляется, переставляя указатель на следующую команду.

Другой специальный регистр, называемый **указателем стека**, ссылается на вершину текущего стека в памяти. Стек содержит по одному фрейму (области данных) для каждой процедуры, в которую уже вошла, но из которой еще не вышла программа. В стековом фрейме процедуры хранятся ее входные параметры, а также локальные и временные переменные, не содержащиеся в регистрах.

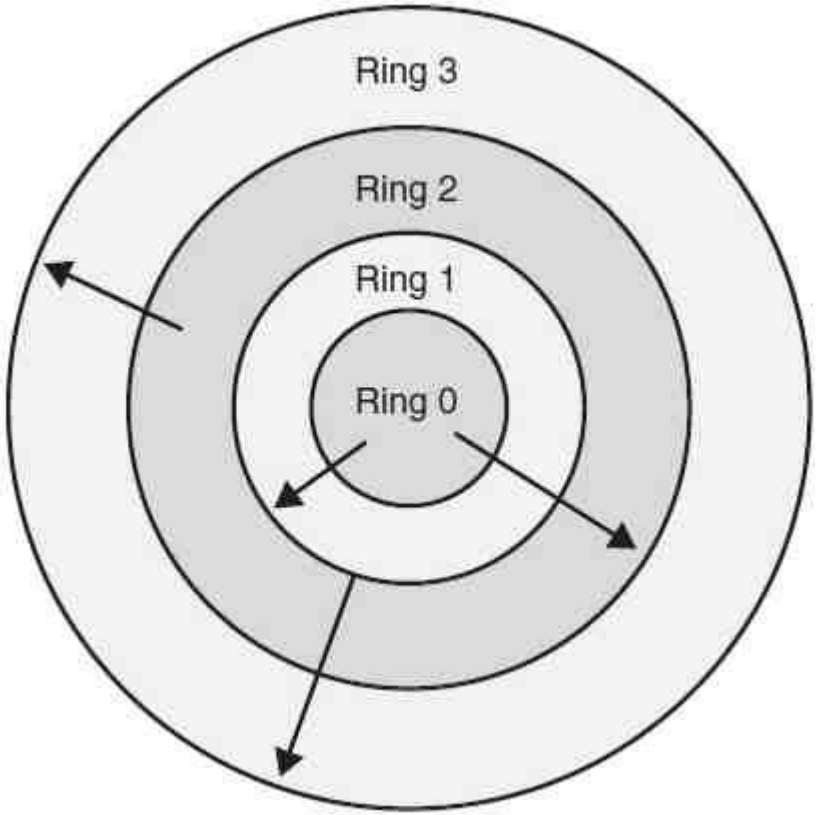
Еще один регистр содержит **слово состояния программы** — PSW (Program Status Word). В этом регистре содержатся биты кода условия, устанавливаемые инструкциями сравнения, а также биты управления приоритетом центрального процессора, режимом (пользовательским или ядра) и другие служебные биты. Обычно пользовательские программы могут считывать весь регистр PSW целиком, но записывать — только в некоторые из его полей. Регистр PSW играет важную роль в системных вызовах и операциях ввода-вывода.

Большинство центральных процессоров, за исключением самых простых, используемых во встраиваемых системах, имеют два режима работы: режим ядра и пользовательский режим (режим пользователя). Обычно режимом управляет специальный бит в слове состояния программы — PSW. При работе в режиме ядра процессор может выполнять любые команды из своего набора и использовать любые возможности аппаратуры. На настольных и серверных машинах операционная система обычно работает в режиме ядра, что дает ей доступ ко всему оборудованию. На большинстве встроенных систем в режиме ядра работает только небольшая часть операционной системы, а вся остальная ее часть — в режиме пользователя.

Пользовательские программы всегда работают в режиме пользователя, который допускает выполнение только подмножества команд и дает доступ к определенному подмножеству возможностей аппаратуры. Как правило, в пользовательском режиме запрещены все команды, касающиеся операций ввода-вывода и защиты памяти. Также, разумеется, запрещена установка режима ядра за счет изменения значения бита режима PSW.

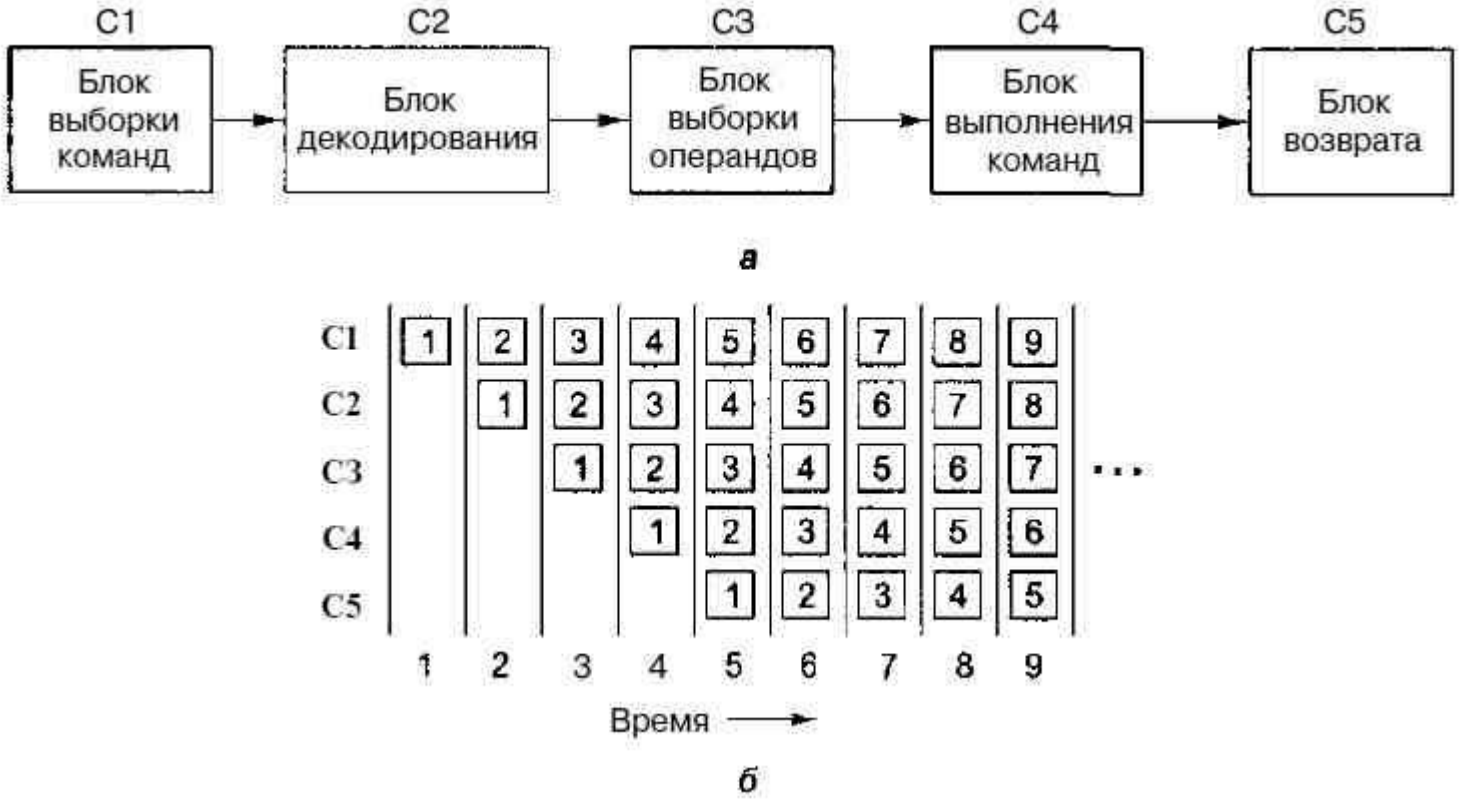
Для получения услуг от операционной системы пользовательская программа должна осуществить **системный вызов**, который перехватывается внутри ядра и вызывает операционную систему.

В процессорах с набором команд x86 или x86-64 предусмотрено 4 уровня привилегий, от наиболее привилегированного уровня 0 до наименее привилегированного уровня 3 . Большинство современных ОС используют уровень 0 для ядра и уровень 3 для пользовательских приложений.



Конвейерная обработка команд

Для повышения производительности процессоров их разработчики давно отказались от простой модели извлечения, декодирования и выполнения одной команды за один цикл. Многие современные процессоры способны одновременно выполнять более одной команды. Например, у процессора могут быть отдельные блоки для выборки, декодирования и выполнения команд, тогда во время выполнения команды *n* он сможет декодировать команду *n* + 1 и осуществлять выборку команды *n* + 2. Подобная организация работы называется **конвейером**. На рисунке внизу показан конвейер с пятью стадиями обработки.

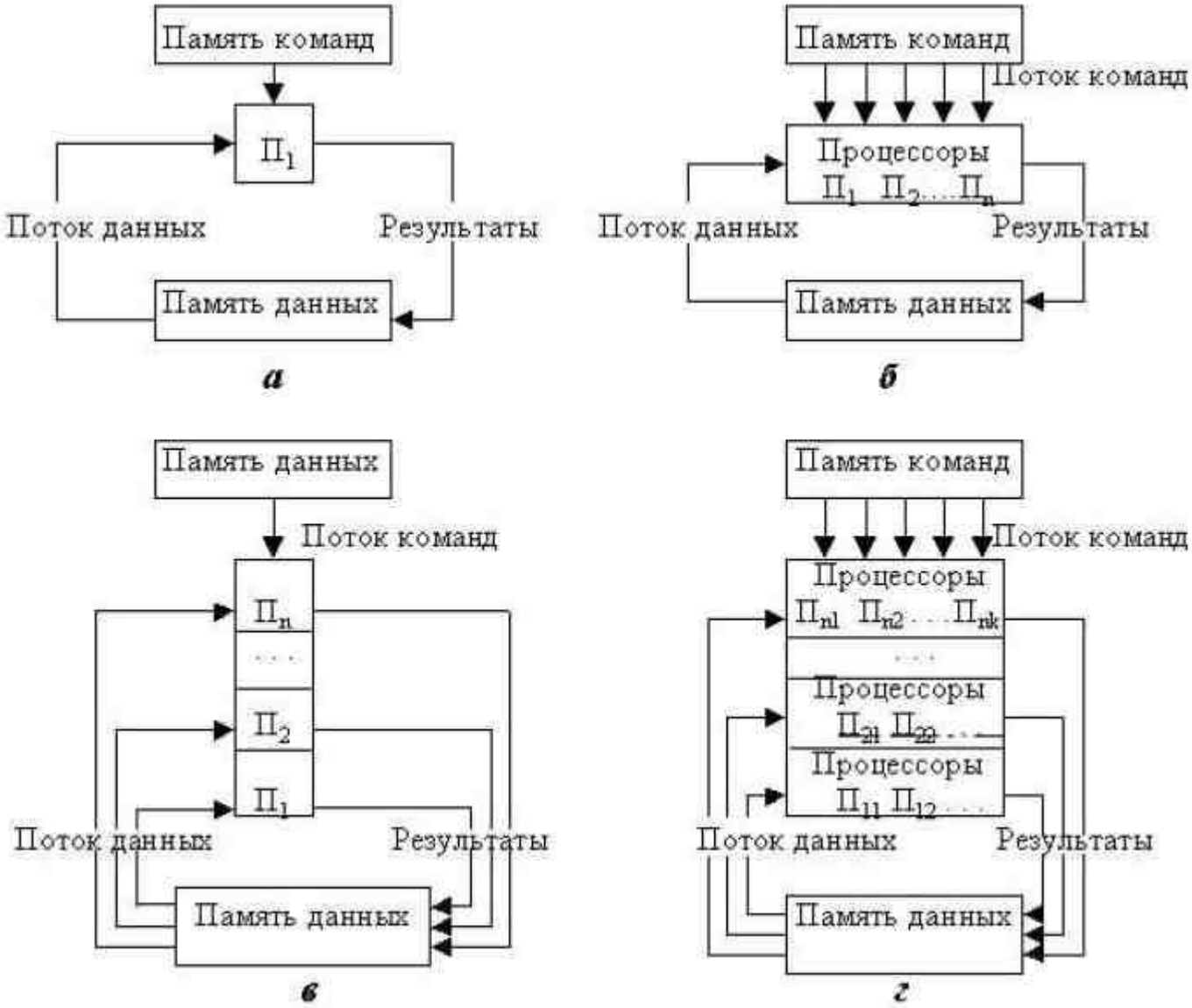


Более совершенной конструкцией по сравнению с конвейерной обладает **суперскалярный** процессор. Он имеет несколько исполнительных блоков, например: один — для целочисленной арифметики, другой — для арифметики чисел с плавающей точкой, третий — для логических операций. Одновременно выбираются две и более команды, которые декодируются и помещаются в буфер хранения, в котором ожидают возможности своего выполнения.

Типы архитектур аппаратных средств

Наиболее общий способ классификации архитектур базируется на понятиях потока команд и потока данных:

- SISD – Simple Instruction / Simple Data – архитектура с простым потоком команд и простым потоком данных
- MISD – Multiple Instruction / Simple Data – архитектура с множественным потоком команд и простым потоком данных
- SIMD – Simple Instruction / Multiple Data – архитектура с простым потоком команд и множественным потоком данных
- MIMD – Multiple Instruction / Multiple Data – архитектура с множественным потоком команд и множественным потоком данных


Память команд
П1
Поток данных
Результаты
Память данных
а
Память команд
Поток команд
Процессоры
П1 П2 ... Пn
Поток данных
Результаты
Память данных
б
Память данных
Поток команд
Пn
...
П2
П1
Поток данных
Результаты
Память данных
в
Память команд
Поток команд
Процессоры
Пn1 Пn2 ... Пnk
...
Процессоры
П21 П22 ...
Процессоры
П11 П12 ...
Поток данных
Результаты
Память данных
г

SISD – включает компьютеры, в которых имеется только один поток команд, команды обрабатываются последовательно и каждая команда инициирует одну операцию с одним потоком данных.

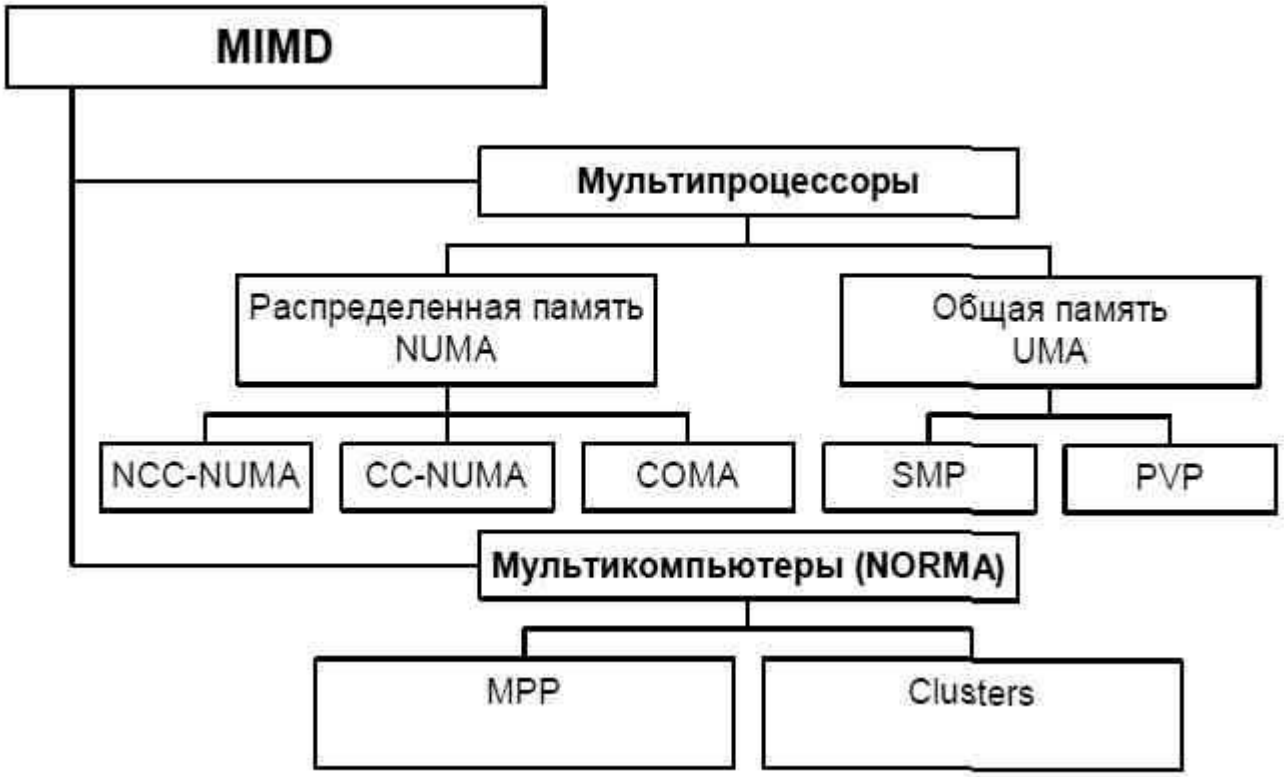
MISD - к данному классу относят конвейерные системы. Некоторые специалисты считают, что пока данный класс пуст.

SIMD – к этому классу относят системы, где множество элементов данных подвергается параллельной, но однотипной обработке.

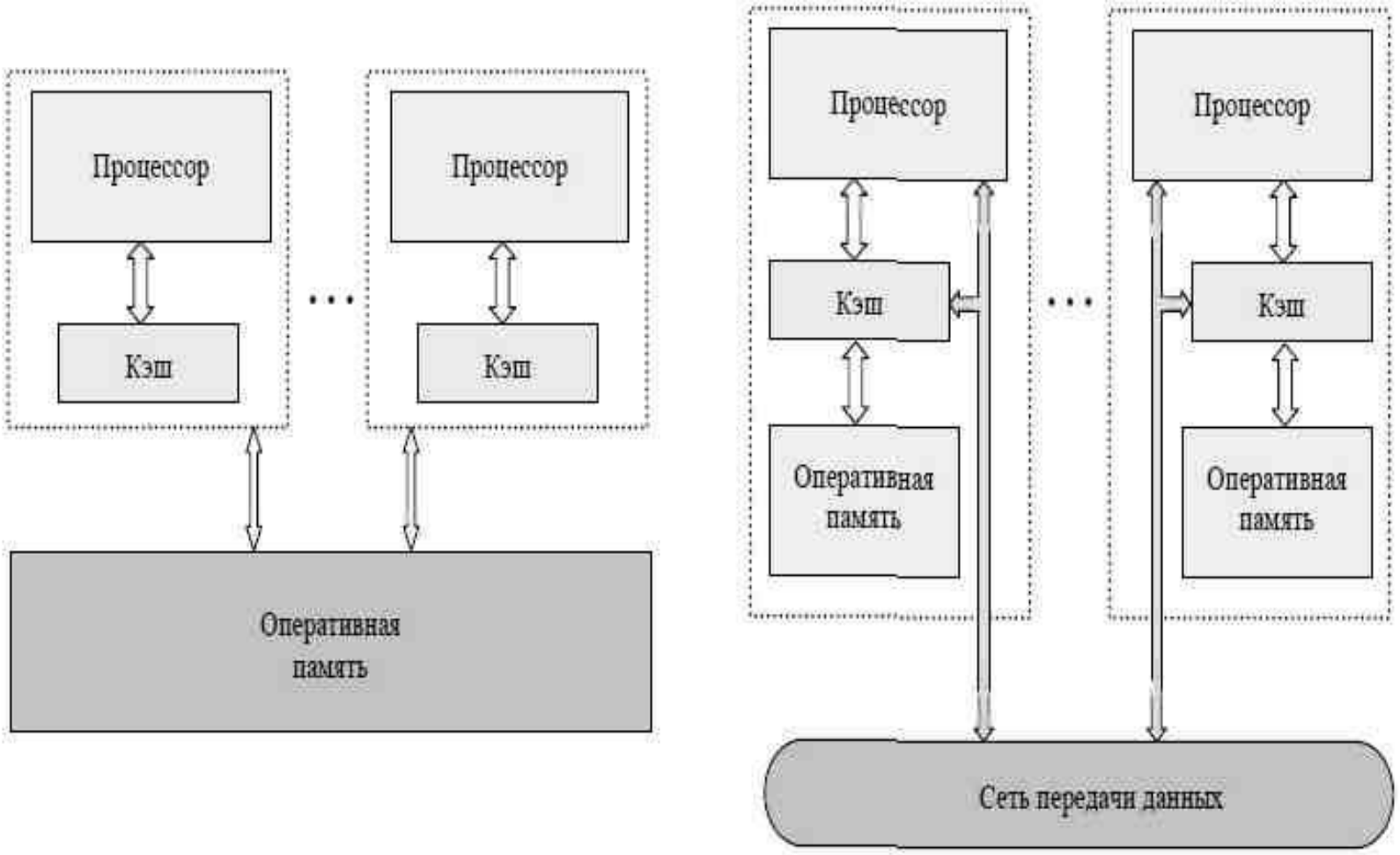
Многие технические задачи используют векторы, над компонентами которых выполняются однотипные операции (например, сложение двух векторов). Для работы с такими данными используются **матричные** и **векторные** процессоры. Матричный процессор содержит решетку вычислительных элементов, операции в которых выполняются одновременно. Векторный процессор содержит векторный регистр, в котором операции над компонентами вектора выполняются последовательно с помощью конвейерной обработки..

Машины с архитектурой SIMD используются в научных исследованиях.

MIMD – здесь несколько независимых процессоров работают как часть большой системы. Структурная схема класса MIMD приведена ниже.



Мультипроцессор представляет собой несколько процессоров в рамках одного вычислительного устройства Для систематики мультипроцессоров учитывается способ построения общей памяти. Единая общая память – рисунок слева – обеспечивает однородный доступ к памяти (uniform memory access, UMA). Распределенная общая память – рисунок справа – обеспечивает неоднородный доступ к памяти (non-uniform memory access, NUMA).


Процессор
Процессор
Кэш
Кэш
Оперативная память
Процессор
Процессор
Кэш
Кэш
Оперативная память
Оперативная память
Сеть передачи данных

SMP (Symmetric Multiprocessor) содержит несколько полностью равноправных процессоров.

PVP (parallel vector processor) параллельный векторный процессор.

Важной проблемой при организации параллельных вычислений является обеспечение информационной целостности (когерентности) кэшей (cache coherence problem).

COMA (cache-only memory architecture) - системы, в которых для представления данных используется только локальная кэш-память.

CC-NUMA (cache-coherent NUMA) - системы, в которых обеспечивается когерентность локальных кэшей разных процессоров.

NCC-NUMA (non-cache coherent NUMA) - системы, в которых обеспечивается общий доступ к локальной памяти разных процессоров без поддержки на аппаратном уровне когерентности кэша

Увеличение производительности процессоров

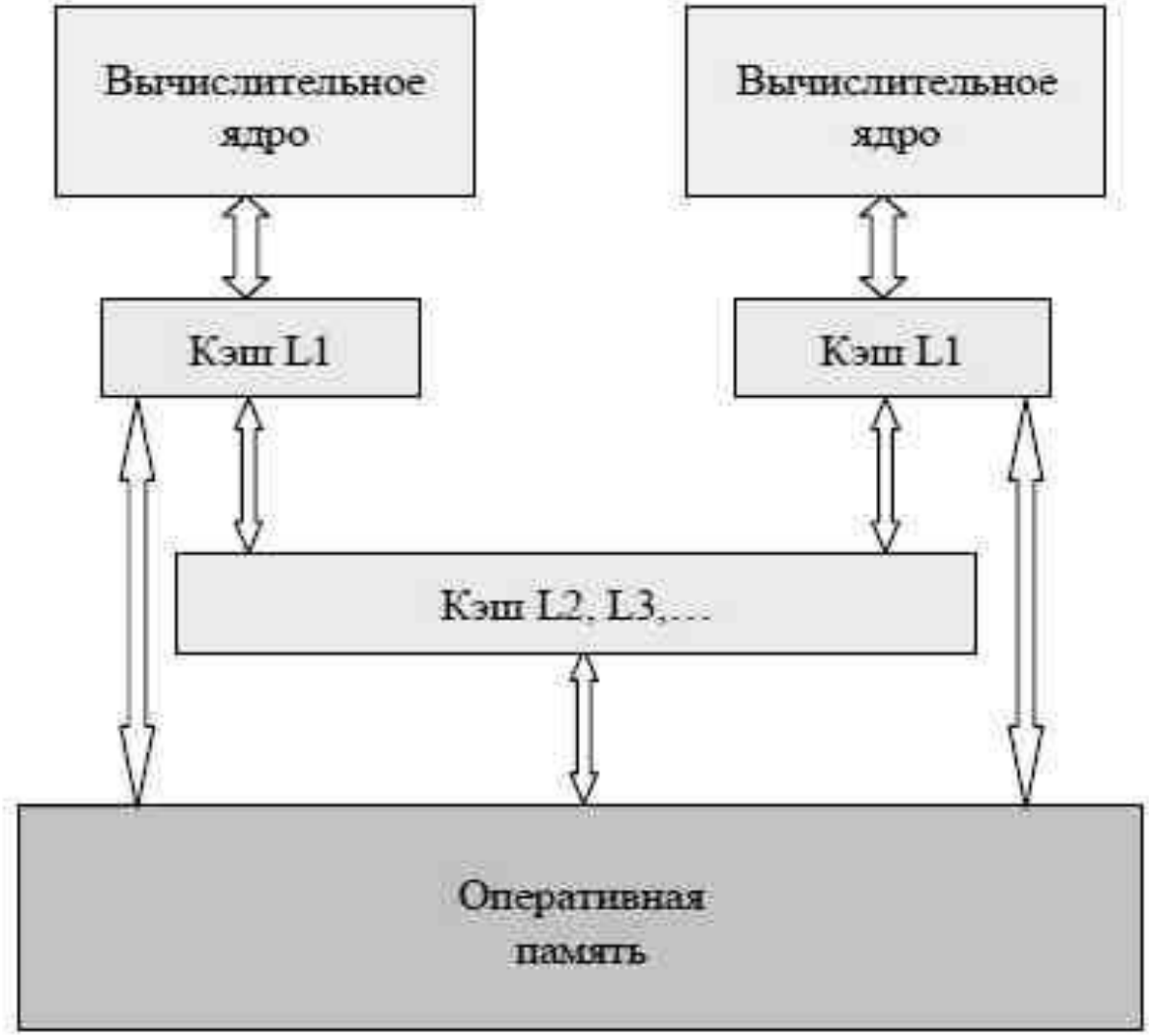
Идея поддержки одновременной многопоточности (simultaneous multithreading, SMT) была предложена в 1995 г. и позднее активно развита компанией Intel под названием технологии гиперпоточности (hyper threading, HT). В рамках такого подхода процессор дополняется средствами запоминания состояния потоков, схемами контроля одновременного выполнения нескольких потоков и т. д. За счет этих дополнительных средств на стадии выполнения может находиться несколько потоков; при этом одновременно выполняемые потоки конкурируют за исполнительные блоки единственного процессора, и как результат выполнение отдельных потоков может блокироваться, если требуемые в данный момент времени ресурсы оказываются уже задействованными. Важно при этом подчеркнуть, что аппаратно-поддерживаемые потоки на логическом уровне операционных систем Linux и Windows воспринимаются как отдельные процессоры, т. е., например, единственный процессор с двумя аппаратно-поддерживаемыми потоками в менеджере Task Manager операционной системы Windows диагностируется как два отдельных процессора.

Технология одновременной многопоточности позволяет достичь многопроцессорности на логическом уровне.

Возможное продвижение по направлению к большей вычислительной производительности может быть обеспечено за счёт реализации в единственном кремниевом кристалле несколько вычислительных ядер в составе одного многоядерного процессора, при этом по своим вычислительным возможностям эти ядра не уступают обычным процессорам.

На логическом уровне архитектура многоядерного процессора соответствует практически архитектуре симметричного мультипроцессора. На рисунке внизу приведена возможная архитектура двуядерного процессора – различия для разных многоядерных процессоров могут состоять в количестве имеющихся ядер и в способах использования кэш-памяти ядрами процессора – кэш-память может быть как общей, так и распределенной для разных ядер. Так, на рисунке кэш-память первого уровня L1 локальна для каждого ядра, в то же время кэш-память всех последующих уровней и оперативная память является общей.

Многоядерность позволяет повышать производительность процессоров, и данный подход обладает целым рядом привлекательных моментов (уменьшение энергопотребления, снижение сложности логики процессоров и т. п.). Все сказанное приводит к тому, что многоядерность становится одним из основных направлений развития компьютерной техники.


Вычислительное ядро
Вычислительное ядро
Кэш L1
Кэш L1
Кэш L2, L3,…
Оперативная память

В последнее время активно разрабатываются различные ускорители вычислений.

Компанией ClearSpeed Technology представлен ускоритель операций над данными с плавающей запятой, представленных в формате с одинарной и двойной точностью. Ускоритель является сопроцессором, разработанным специально для серверов и рабочих станций, которые основаны на 32-х или 64-х битной x86 архитектуре, и построен на базе двухядерного процессора.

Компания NVIDIA представила продукты семейства Tesla для построения высокопроизводительных вычислительных систем. Tesla поддерживает операционные системы семейств Windows и Linux. В комплект поставки входят следующие компоненты: CUDA Driver, CUDA Toolkit, CUDA SDK.

CUDA (Compute Unified Device Architecture) - программно аппаратное решение, позволяющее использовать видеопроцессоры для вычислений общего назначения.

Системы команд CISC и RISC

Основную идею CISC-архитектуры отражает ее название — «полный набор команд» (Complex Instruction Set Computer). В данной архитектуре стремятся иметь отдельную машинную команду для каждого возможного действия по обработке данных.

Исторически CISC-архитектура была одной из первых. Совершенствование процессоров шло по пути создания машин, способных выполнять как можно больше разных команд. Это упрощало работу программистов, которые писали программы на языке ассемблера. Использование сложных команд позволяло сократить размер и время разработки программы.

В итоге сложились следующие черты организации CISC-процессоров:

- большое количество различных машинных команд (сотни), каждая из которых выполняется за несколько тактов центрального процессора;
- устройство управления с программируемой логикой;
- небольшое количество регистров общего назначения (РОН);
- различные форматы команд с разной длиной;
- преобладание двухадресной адресации;
- развитый механизм адресации операндов, включающий различные методы косвенной адресации.

CISC-подход, однако, привел к тому, что некоторые команды стало невозможно выполнять чисто аппаратными средствами. В результате в процессорах появились блоки, заменяющие наиболее сложные команды последовательностями из более простых команд. Кроме того, из-за высокой сложности команд и их обилия устройство управления приходилось строить только на основе программируемой логики, то есть с применением медленной управляющей памяти.

Все эти факторы привели к повороту в сторону RISC-архитектуры (Reduce Instruction Set Computer). В то же время целый ряд несомненных достоинств CISC-архитектуры сохраняют ее актуальность (прежде всего, в глазах разработчиков программных приложений).

Именно поэтому ведущие фирмы-производители ВМ (Intel, AMD, IBM и др.) в своих последних разработках, по-прежнему, не отказываются от CISC-подхода.

Главные усилия в архитектуре RISC направлены на построение максимально эффективного конвейера команд, то есть такого, где все команды извлекаются из памяти и поступают в ЦП на обработку в виде равномерного потока, причем ни одна команда не должна находиться в состоянии ожидания, а ЦП должен оставаться загруженным на протяжении всего времени. Кроме того, идеальным будет вариант, когда любой этап цикла команды выполняется в течение одного тактового периода.

Для этого нужно, чтобы все команды имели стандартную длину, равную ширине шины данных, соединяющей ЦП и память. Помимо одинаковой длины команд важно иметь относительно простую подсистему декодирования и управления: сложное устройство управления будет вносить дополнительные задержки в формирование сигналов управления. Очевидный путь существенного упрощения устройства управления — сокращение числа выполняемых команд, форматов команд и данных, а также способов адресации.

Основная причина, препятствующая сведению всех этапов цикла команды к одному тактовому периоду — потенциальная необходимость доступа к памяти для выборки операндов и/или записи результатов. По этой причине желательно максимально сократить число команд, имеющих доступ к памяти.

Для упрощения выполнения большинства команд и приведения их к формату «регистр-регистр» требуется снабдить ЦП значительным числом регистров общего назначения.

Суммируя сказанное, концепцию RISC-процессора можно свести к следующим положениям:

- выполнение всех (или, по крайней мере, 75% команд) за один цикл;
- стандартная однословная длина всех команд, равная естественной длине слова и ширине шины данных и допускающая унифицированную конвейерную обработку всех команд;
- малое число команд (не более 128);
- малое количество форматов команд (не более 4);
- малое число способов адресации (не более 4);
- доступ к памяти только посредством команд «Чтение» и «Запись»;
- все команды, за исключением «Чтения» и «Записи», используют внутрипроцес-сорные межрегистровые пересылки;
- устройство управления с аппаратной логикой;
- относительно большой (не менее 32) процессорный файл регистров общего назначения (число РОН в современных RISC-микропроцессорах может превышать 500).

RISC-процессорами являются процессоры POWER (IBM), ARM (ARM Limited).