Вопросы к экзамену

по учебной дисциплине «Архитектура компьютеров»

(**2 курс ПИ, 2020/21** учебный год)

1. Структура и функции компьютера.

Центральная вычислительная машина содержит следующие основные устройства: (арифметическое устройство); (запоминающее устройство) – память; (устройство управления); устройство ввода (Вв) и вывода (Выв); пульт ручного управления (клавиатуру). Арифметическое устройство производит арифметические и логические операции над поступающими в него словами. Запоминающее устройство (память) хранит информацию, передаваемую в него из других устройств, в том числе поступающую в машину извне через устройство ввода, и выдает во все другие устройства информацию, необходимую для продолжения вычислительного процесса. Память машины, как правило, состоит из двух частей: (оперативное запоминающее устройство) и (внешнее запоминающее устройство). Если быстродействие оперативное запоминающее устройства оказывается недостаточным для получения нужной скорости работы машины, между АУ и оперативное запоминающее устройство размещают С(верх) оперативно запоминающего устройства. Устройство управления автоматически управляет вычислительным процессом, посылая всем другим устройствам сигналы, предписывающие им те или иные действия. С помощью пульта ручного управления (клавиатуры) оператор запускает и останавливает машину, а при необходимости может вмешиваться в процесс решения задачи.

1. Два способа представления архитектуры.

Архитектура компьютера, характеризующая его логическую организацию, может быть представлена как множество взаимосвязанных компонент, включающих, на первый взгляд, элементы различной природы –ПО, аппаратное обеспечение, алгоритмическое обеспечение, специальное фирменное обеспечение – и поддерживающих его слаженное функционирование в форме единого архитектурного ансамбля, позволяющего вести эффективную обработку различных объектов. Архитектура вычислительной системы – это совокупность основных функциональных возможностей системы, сфер применения (научно-техническая, экономическая, управление и т.д.), режимов работы (пакетный, мультипрограммный, разделения времени, диалоговый и т.д.), характеристик вычислительной системы (ВС) (быстродействие, набор и объем памяти, набор периферийных устройств и т.д.), особенностей структуры (одно-, многопроцессорная) и реализующей базы, т.е. совокупность software, hardware, brainware и firmware. Архитектура ВС – вычислительные и логические возможности, аппаратные средства, ПО

1. История нововведений в архитектуру компьютеров.

• IBM 701 (1953 г.), IBM 704 (1955 г.): разрядно-параллельная память, разряднопараллельная арифметика.

• IBM 709 (1958 г.): независимые процессоры ввода-вывода.

• «IBM Stretch» (1961 г.): опережающий просмотр, расслоение памяти.

• «Atlas» (1963 г.): конвейер команд.

• CDC 6600 (1964 г.): независимые функциональные устройства (ФУ).

• CDC 7600 (1969 г.): конвейерные независимые функциональные устройства.

• ILLIAC IV (1974 г.): матричные процессоры.

• «Cray-1» (1976 г.): векторно-конвейерные процессоры.

На современном этапе высокопроизводительная техника развивается по следующим основным направлениям.

1. Векторно-конвейерные компьютеры.
2. Массово-параллельные компьютеры с распределенной памятью.
3. Параллельные компьютеры с общей памятью.
4. Компьютеры с кластерной архитектурой.
5. Грид-системы (Grid system).
6. Вычислительные и логические возможности компьютеров.

Вычислительные и логические возможности ВС. Возможности вычислительной системы обусловливаются системой команд (СК), характеризующей гибкость программирования, форматами данных и скоростью выполнения операций, определяющих класс задач, наиболее эффективно решаемых на ВС. Система команд ВС, базирующихся на архитектуре фон Неймана, сегодня мало чем отличается от СК ЭВМ 1950-х гг. Большинство достижений в этой области остались незамеченными проектировщиками. СК – в зависимости от класса решаемых задач. Как видно из схемы, для решения задач любого класса необходимы команды, востребованные задачами управления. Следовательно, эти типы команд должны присутствовать в любом компьютере. Большое влияние на точность выполнения операций оказывают форматы данных. Алгоритмы выполнения операций достаточно полно отражают производительность только однопроцессорных ВС.

1. Аппаратные средства.

Простейшая вычислительная система включает модули пяти типов: центральный процессор, основную память, каналы, контроллеры и внешние устройства. Процессор (устройство управления + арифметико-логическое устройство + запоминающее устройство) управляет работой системы и обеспечивает вычисления непосредственно по программе. Выполнение машинных команд, команд ввода-вывода (1/0), обращение к памяти, управление состоянием устройств инициализируются или выполняются с помощью процессора. Основная память предназначена для хранения команд и данных и обеспечивает адресный доступ к ним от процессора. Каналы – спецустройства, управляющие обменом данных с внешними устройствами. Это, по сути, спецпроцессор ввода-вывода, обеспечивающий работу внешних устройств, контроль информации и т.д. Контроллеры ввода-вывода - это устройства управления внешнее устройство. Внешние устройства служат для ввода-вывода информации с различных носителей.

1. Программное обеспечение.

Программное обеспечение является составной частью архитектуры компьютера и существенно влияет на весь вычислительный процесс. Операционная система (ОС) управляет ресурсами, разрешает конфликтные ситуации, оптимизирует функционирование системы в целом. Широкий спектр языков программирования позволяет описывать практически любые задачи. Выбор языка программирования и ОС существенно влияет на скорость решения задачи. На сегодняшний день сложилась следующая классификация программного обеспечения:

• системные программы, управляющие работой компьютера, а также выполняющие некоторые вспомогательные функции: создание копий используемой информации, выдача справочной информации о компьютере, проверка работоспособности устройств и т.д.;

• прикладные программы, выполняющие нужные пользователю работы: редактирование текстов, рисование изображений, обработка информационных массивов, игры и т.д.;

• инструментальные системы, или системы программирования, которые используются при разработке новых программ для компьютеров.

Роль программного обеспечения (ПО) необычайно велика при решении тематических задач.

1. Архитектура фон Неймана.

Архитектурные принципы формулировались фон Нейманом применительно к созданию автоматического устройства для решения дифференциальных уравнений. Основные характеристики архитектуры неймановского типа следующие:

• для хранения программ и данных используется последовательно адресуемая единственная память линейного типа;

• команды и данные различаются через идентификатор неявным способом лишь при выполнении операций;

• назначение данных определяется лишь логикой программы.

Указанные свойства были исключительно важными для своего времени. Однако появление языков высокого уровня (ЯВУ), новых методов решения, логических способов ускорения операций, более совершенной элементной базы выдвигает необходимость принципиально новых подходов к возможностям архитектуры. Требования ЯВУ имеют следующие особенности:

• память состоит из набора дискретных именуемых переменных;

• ЯВУ наряду с линейными данными оперируют и с многомерными: массивами, структурами, списками;

• в ЯВУ четко разграничены операции и данные;

• данные определяют и операции над ними.

Архитектура фон Неймана плохо приспособлена к выполнению программ на языке высокого уровня. Тем не менее ЯВУ, к сожалению, имитируют в своей структуре архитектуру неймановского типа.

1. Особенности архитектуры программного обеспечения.

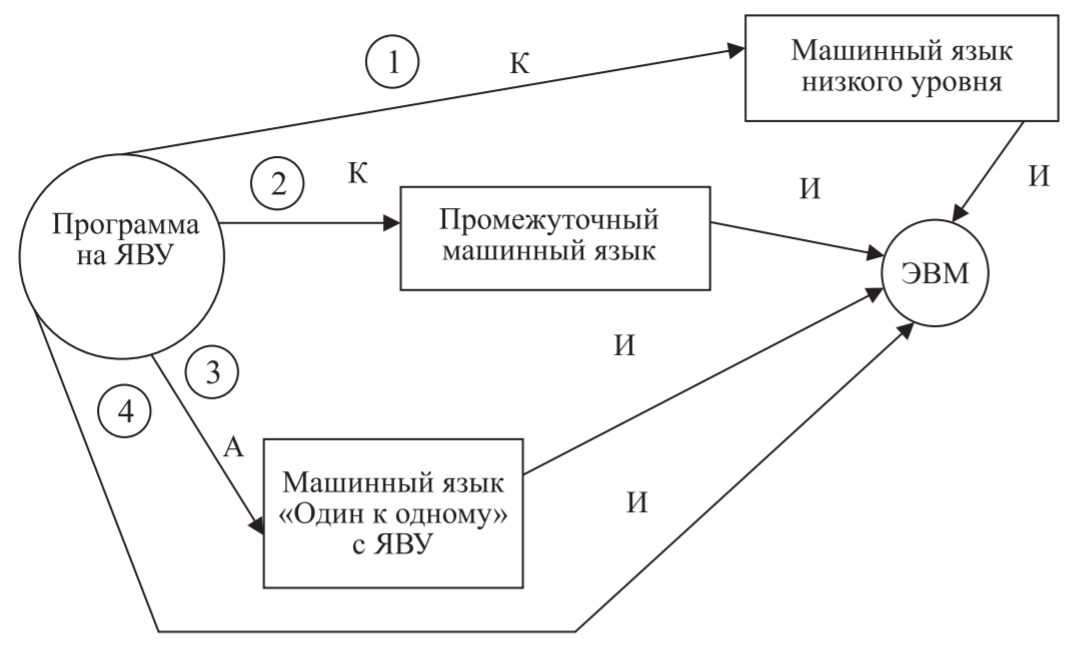
Общая взаимосвязь между язык высокого уровня (ЯВУ) и компьютером в зависимости от уровня языка машины может быть представлена **схемой**:

1 – традиционный подход: после компилирования (К) программа переводится на машинный язык, а затем интерпретируется (И) машиной;

2 – проводится компиляция на машинный язык более высокого уровня;

3 – ЯВУ можно рассматривать как язык ассемблера, т.е. имеется взаимно однозначное соответствие между типами операторов и знаками операций ЯВУ с командами машинного;

4 – машинный язык является языком высокого уровня; идет процесс интерпретации программы на компьютере (электро́нно-вычисли́тельная маши́на для таблицы)



1. RISC и CISC – архитектура.

Двумя основными архитектурами набора команд, используемыми компьютерной промышленностью на современном этапе развития вычислительной техники, являются архитектуры CISC и RISC. RISC-архитектура (Restricted (reduced) Instruction Set Computer) – это регистроориентированная архитектура. Компьютеры с такой архитектурой иногда называют компьютерами с сокращенным набором команд. Суть ее состоит в выделении наиболее употребительных операций и создании архитектуры, приспособленной для их быстрой реализации. В компьютерной индустрии наблюдается настоящий бум по производству систем с RISC-архитектурой. CISC-архитектура (Complete Instruction Set Computer) – архитектура с полным набором команд. Основоположником ее можно считать фирму IBM. Простота архитектуры RISC-процессора обеспечивает его компактность, практическое отсутствие проблем с охлаждением кристалла, чего нет в процессорах фирмы «Intel», упорно придерживающейся пути развития архитектуры CISC. На сегодняшний день CISC-процессоры почти монопольно занимают на компьютерном рынке сектор персональных компьютеров, однако RISC-процессорам нет равных в секторе высокопроизводительных серверов и рабочих станций. CISC – многобайтовые команды, малое количество регистров, сложные команды, одна или менее команд за один цикл процессора, одно исполнительное устройство. RISC – однобайтовые команды, большое количество регистров, простые команды, несколько команд за один цикл процессора, несколько исполнительных устройств.

1. Компьютер в сфере управления технологическим процессом.

Для автоматизации управления сложным производственным или технологическим процессом в контур управления включают компьютер, т.е. управляющую вычислительную машину (УВМ). Наиболее часто в качестве управляющего компьютера используют цифровую электро́нно-вычисли́тельная маши́на ЭВМ благодаря следующим ее качествам:

• наличию больших и высоконадежных Запоминающих устройств (ЗУ) различного типа;

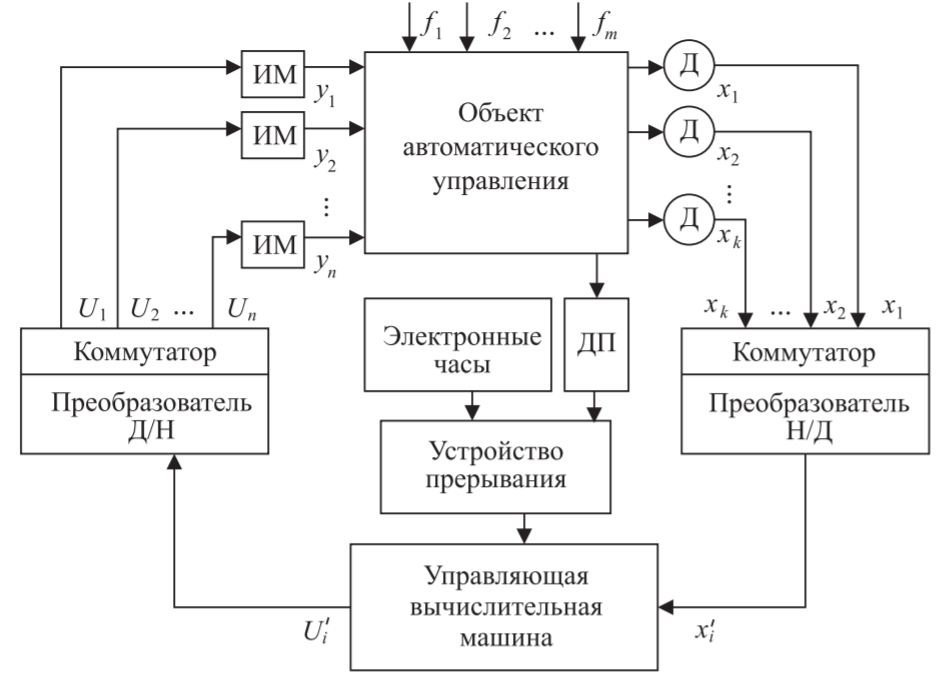
• возможности решения сложных вычислительных и логических задач;

• гибкости (за счет ПО);

• надежности и быстродействию.

В общем случае система автоматического управления с УВМ определяет собой замкнутый контур. **(СХЕМА)**

На вход УВМ от датчика Д передается измерительная информация о текущем значении параметров x1, x2, …, xk. Согласно алгоритму управления, УВМ определяет величину управляющих воздействий U1, U2, …, Un, которые необходимо приложить к Исполнительному Механизму для изменения регулируемых параметров y1, y1, …, yn, с тем чтобы управляющий процесс протекал оптимально. В режиме советчика УВМ обрабатывает измерительную информацию с объекта и рассчитывает управляющие воздействия для оптимизации процесса. Эта информация служит рекомендацией оператору, управляющему технологическим процессом.



1. Организация конвейера.

Конвейерная обработка в общем случае основана на разделении подлежащей исполнению функции на более мелкие части, называемые ступенями, и выделении для каждой из них отдельного блока аппаратуры. Конвейерная обработка – это технология, при которой несколько команд выполняются одновременно в процессе обработки, т.е. используется параллелизм инструкций в последовательном потоке команд. Выполнение типичной команды можно разделить на следующие этапы:

• выборка команды If (Instruction Fetch): по адресу, заданному счетчиком команд, из памяти извлекается команда;

• декодирование команды Id (Instruction Decode);

• обращение к памяти Mem (Memory);

• выполнение операции Ex (Execute);

• запоминание результата Wb (Write Back).

Для представления работы конвейера используются временные диаграммы.

Такая организация функционирования направлена на возможно более полную загрузку всех компонент системы. Конвейеризация увеличивает пропускную способность процессора (количество команд, завершающихся в единицу времени), но не сокращает время выполнения отдельной команды.

1. Классификация конфликтов и методы их разрешения.

При реализации конвейерной обработки возникают ситуации, препятствующие выполнению очередной команды из потока команд в предназначенном для нее такте. Такие ситуации называются конфликтами.

Существуют три класса конфликтов:

• структурные конфликты, возникающие из-за конфликтов по ресурсам, когда аппаратные средства не могут поддерживать все возможные комбинации команд в режиме одновременного выполнения с совмещением;

• конфликты по данным, возникающие в случае, когда выполнение одной команды зависит от результата выполнения предыдущей команды;

• конфликты по управлению, возникающие при конвейеризации команд переходов и других команд, которые изменяют значение счетчика команд.

Структурные конфликты возникают, например, и в машинах, в которых имеется единственный конвейер памяти для команд и данных. В этом случае одна команда, содержащая обращение к памяти за данными, будет конфликтовать с выборкой более поздней команды из памяти. Чтобы разрешить эту ситуацию, можно приостановить конвейер на один такт в момент обращения к памяти за данными.

1. Конфликты по данным.

Конфликт по данным возникает всегда, когда имеет место зависимость между командами и они расположены по отношению друг к другу так близко, что совмещение операций, происходящее при конвейеризации, может привести к изменению порядка обращения к операндам.

Рассмотрим две команды – i и j, при этом i предшествует j. Возможны следующие конфликты:

• RAW (чтение после записи): команда j пытается прочитать операнд-источник данных прежде, чем команда i запишет туда данные. Таким образом, j может некорректно получить старое значение.

• WAR (запись после чтения): команда j пытается записать результат в приемник прежде, чем он считывается оттуда командой i, так что i может некорректно получить новое значение.

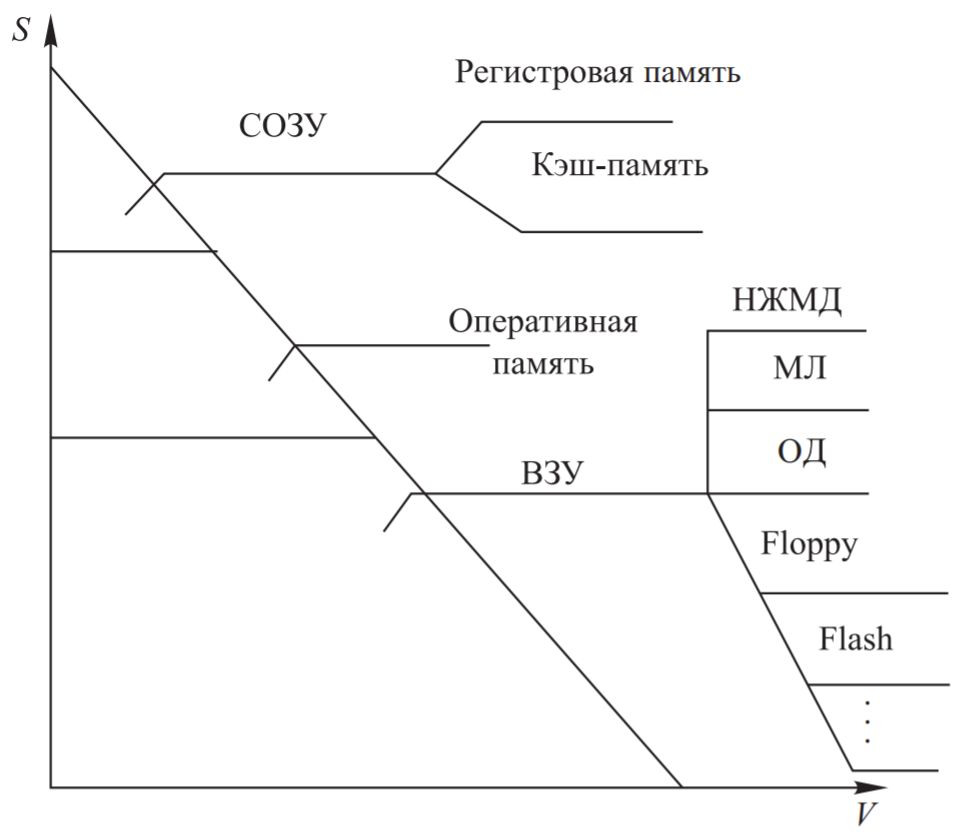
• WAW (запись после записи): команда j пытается записать операнд прежде, чем будет записан результат команды i, т.е. записи заканчиваются в неверном порядке, оставляя в приемнике значение, записанное командой i, а не j.

Указанные конфликты могут быть разрешены с помощью достаточно простой аппаратной техники, которая называется пересылкой или продвижением данных (data forwarding), обходом (data bypassing), иногда закороткой (short-circuiting).

Существуют аппаратные методы, позволяющие изменить порядок выполнения команд программы так, чтобы минимизировать приостановки конвейера. Эти методы получили общее название методов динамической оптимизации.

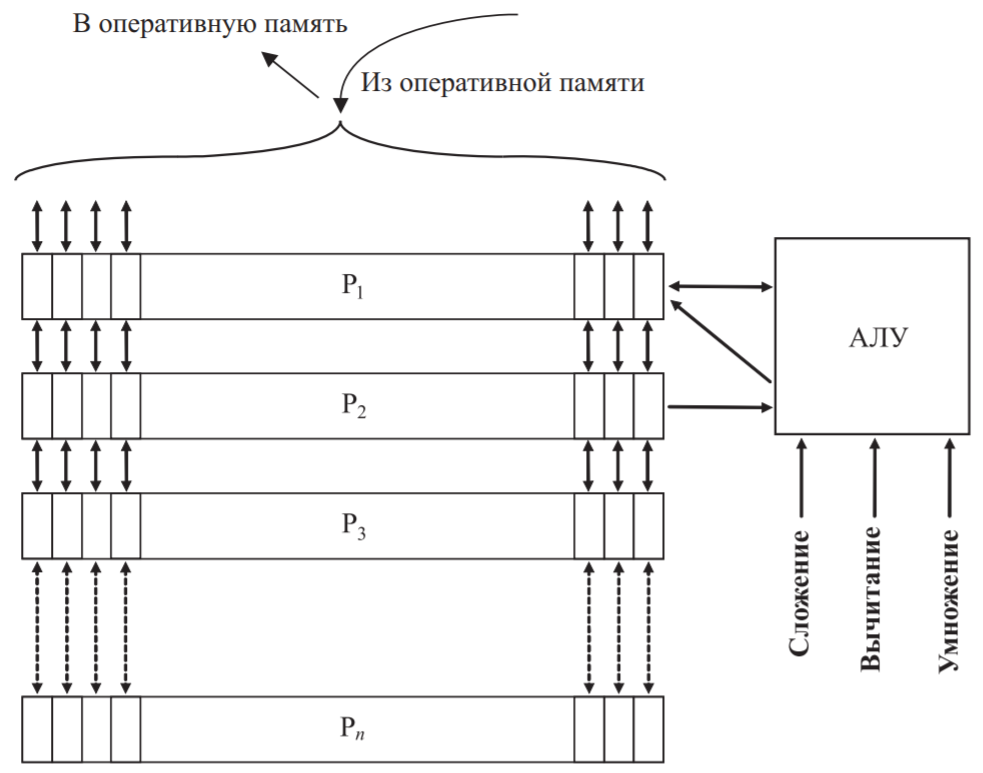
1. Иерархия памяти.

Память компьютера может быть организована как многоуровневая с различным объемом, временем доступа к ней и стоимостью в пересчете на бит – сверхоперативная, оперативная, внешняя **(РИСУНОК)**, так и одноуровневая, виртуальная. Почти всегда виртуальная память есть переупорядоченное подмножество реальной памяти. Причем более высокий уровень меньше по объему, быстрее и имеет большую стоимость в пересчете на бит, чем более низкий уровень. Уровни иерархии взаимосвязаны: все данные на одном уровне могут быть также найдены на более низком уровне, а все данные на этом более низком уровне могут быть найдены на следующем нижележащем уровне и т.д., пока не будет достигнуто основание иерархии.

Основная память предназначена для хранения команд и данных и обеспечивает адресный доступ к ним от процессора. Постоянная память предназначена только для чтения ее содержимого. Небольшая по объему полупостоянная память используется для хранения параметров конфигурации компьютера. Часто ее называют CMOS-памятью

1. Регистровая память.

Простейшая регистровая структура используется в организации стековой памяти. Стековая память представляет собой набор из n регистров, каждый из которых способен хранить одно машинное слово. Одноименные разряды регистров P1, P2, ..., Pn соединены между собой цепями сдвига. Поэтому весь набор регистров может рассматриваться как группа n-разрядных сдвигающих регистров, составленных из одноименных разрядов регистров P1, P2, ..., Pn. Информация в стеке может продвигаться между регистрами вверх и вниз. Движение вниз: (P1) → P2, (P2) → P3 и т.д.; P1 заполняется данными из главной памяти. Движение вверх: (Pn) → Pn–1, (Pn–1) → Pn–2; Pn заполняется нулями. Регистры P1 и P2 связаны с АЛУ, образуя два операнда для выполнения операции. Результат операции записывается в P1. Следовательно, АЛУ выполняет операцию (P1) ⊗ (P2) → P1. Одновременно с выполнением арифметической операции (АО) осуществляется продвижение операндов вверх, не затрагивая содержимого P1 т.е. (P3 ) → P2 , (P4 ) → P3 и т.д., а содержимое регистра Р1 остается без изменений.



1. Организация кэш-памяти.

Кэши – следующий уровень иерархии памяти. Они сравнительно небольшие по объему, но имеют высокую скорость. В кэш-памяти хранятся команды и данные, которые часто используются и требуют малых временных затрат для доступа к ним. Появление кэш-памяти обусловлено в первую очередь тем, что быстродействие центрального процессора намного превысило скорость работы памяти. Кэши прозрачны для программного обеспечения. Обычно никакое явное управление входами кэша невозможно. Данные распределяются и управляются кэшем автоматически. Некоторые процессоры осуществляют явный контроль над кэшированием данных.

Кэш-память позволяет организовать работу медленной оперативной памяти как быстродействующей памяти, оптимизируя следующие аспекты:

• максимизирование коэффициента попадания;

• уменьшение времени доступа;

• уменьшение штрафа промаха;

• уменьшение непроизводительных затрат для поддержания консистентности кэша.

Кэш-память – это способ организации совместного функционирования двух типов запоминающих устройств, отличающихся временем доступа и стоимостью хранения данных, что позволяет уменьшить среднее время доступа к данным за счет динамического копирования наиболее часто используемой информации из медленного запоминающего устройства в быстрое.

1. Замещение блоков в кэше. Алгоритмы записи в кэш.

В системах, оснащенных кэш-памятью, каждый запрос к оперативной памяти выполняется в соответствии со следующим алгоритмом:

1) просматривается содержимое кэш-памяти с целью определения, не находятся ли нужные данные в кэш-памяти;

2) при возникновении промаха контроллер кэш-памяти выбирает подлежащий замещению блок.

Как правило, для замещения блоков применяются две основные стратегии – случайная и LRU (Least Recently Used – вытеснение давно неиспользуемого).

В первом случае, чтобы иметь равномерное распределение, блоки-кандидаты выбираются случайно. Во втором случае, чтобы уменьшить вероятность перемещения информации, которая в ближайшее время может потребоваться, все обращения к блокам фиксируются.

Очень часто организация кэш-памяти в компьютерах отличается именно стратегией выполнения записи. Когда выполняется запись в кэш-память, имеются две базовые возможности:

• сквозная запись (write through, store through): информация записывается в две области – в блок кэш-памяти и в блок более низкого уровня памяти;

• запись с обратным копированием (write back, copy back, store in): информация записывается только в блок кэш-памяти. Модифицированный блок кэш-памяти записывается в основную память, только когда он замещается.

1. USB – flash.

**USB-флеш-накопитель** подключается к компьютеру или иному считывающему устройству по интерфейсу USB. Основное назначение USB-накопителей – хранение, перенос и обмен данными, резервное копирование, загрузка операционных систем и др. USB-флеши обычно съемные и перезаписываемые (рис. 3.10). Достоинства:

• небольшой вес, бесшумность работы и портативность;

• универсальность в применении;

• низкое энергопотребление;

• работоспособность в широком диапазоне температур;

• высокая устойчивость к механическим воздействиям;

• длительное автономное хранение информации.

Недостатки:

• ограниченное число циклов записи-стирания перед выходом из строя; чипы памяти, сделанные по технологии MLC (большинство), чаще всего выдерживают не более 5000 циклов перезаписи; ограниченный ресурс USB-коннектора (около 1500 подключений);

• скорость записи и чтения ограничена пропускной способностью USB;

• в отличие от компакт-дисков имеют недостатки, свойственные любой электронике: чувствительны к радиации и электростатическому разряду.

1. Дополнительная память: floppy, MD и т.д. Технологии записи.

Доступ к данным, расположенным в разноуровневой памяти, существенно различается по времени. Доступ к регистровой памяти измеряется несколькими наносекундами, к оперативной памяти (ОП) – десятками наносекунд, к оптическим дискам – секундами. Магнитные диски представляют собой алюминиевые диски с магнитным покрытием, диаметр которых колеблется от 12 до 3 см и менее. Головка диска, поддерживаемая воздушной подушкой, записывает или считывает с соответствующего участка движущегося диска биты информации. Последовательность битов, записанная по окружности диска за его полный оборот, называют дорожкой. Дискета (floppy disk) имеет то же назначение, что и магнитные диски. При чтении информации с дискеты головка непосредственно касается ее поверхности, в то время как головка магнитного диска перемещается над его поверхностью на воздушной подушке. Дискеты различаются физическими размерами, объемом хранимой информации и плотностью записи: низкой (LD – Low Density) и высокой (HD – High Density). Оптические диски имеют более высокую плотность записи, чем магнитные, и обеспечивают хранение огромных объемов информации. Впервые оптические диски были разработаны фирмой «Philips». В 1980 г. фирмы «Philips» и «Sony» совместно разработали Compact Disk (CD). Универсальная память, разработанная компанией «Nantero» на кремниевой основе размером с компакт-диск, способна хранить 10 млрд бит цифровой информации. Кодировка осуществляется через физическую ориентацию наномерных структур. Здесь каждая ячейка памяти изготовлена из углеродных нанотрубок размером менее 0,1 · 10–3 человеческого волоса. В перспективе такая память позволит хранить триллионы битов информации на одном квадратном сантиметре.

1. Виртуальная память.

Виртуальный адрес (ВА) – адрес, по которому ссылаются на ячейку виртуальной памяти. Область виртуальных адресов – это множество всех виртуальных адресов. Использование виртуальной адресации обусловливается рядом обстоятельств. Однородность области адресов. Защита памяти. Изменение структуры памяти. Для удовлетворения вышеприведенных обстоятельств требуется аппарат страничной организации памяти. Виртуальную память пользователя можно разделить на три типа: • «активные» блоки, которые содержат программу и данные, используемые в текущий момент; • «пассивные» блоки, содержащие программу и данные, которые будут использоваться при выполнении программы; • «мнимые» блоки, к которым не обращаются на протяжении выполнения программы. Аппарат виртуальной адресации должен отображать виртуальную среду на реальную, причем так, чтобы «активные» блоки находились в оперативной памяти, «пассивные» – по возможности в оперативной или вспомогательной, а «мнимые» – нигде. Преобразование виртуального адреса в реальный осуществляется с помощью регистров адресов страниц (РАС). Структура виртуального адреса: [N блока, N строки]. Каждой странице ОП соответствует свой РАС [N блока, занимающего в данный момент страницу, Разряд использования, Разряд записи, Разряд активности]. Аппарат виртуальной адресации отображает виртуальный адрес в реальный следующим образом: виртуальный адрес сравнивается одновременно с содержимым всех РАС. Единственным сравнимым с ним РАС будет регистр, содержащий тот же номер блока и «1» в разряде активности. Регистр определяет номер страницы, с которой он связан. Для получения реального адреса памяти к номеру страницы данного РАС присоединяется номер строки из виртуального адреса.

1. Топологии компьютерных сетей.

Топология – это конфигурация соединения элементов Локальной Вычислительной Сети. Основные топологические решения делят, как правило, на два типа:

• широковещательные, где каждый источник PS (Physical Signalling) передает сигналы, которые могут восприниматься всеми приемниками (остальными PS): топологии «шина», «дерево», «звезда с пассивным центром», «ячейка», «решетка»;

• последовательные, где каждый физический подуровень передает информацию только одному из PS: топологии «кольцо», «цепочка», «звезда с интеллектуальным центром», «сетка», «снежинка».

Основная конфигурация широковещательного типа – «шина». Коммуникации между территориально распределенными устройствами в ЛВС в чем-то похожи на коммуникации между модулями в компьютерах: высокая скорость передачи данных, частая смена структуры потока, неравномерная загрузка. Основное их отличие состоит в том, что скорость передачи данных в ЛВС может быть ниже, а длительность «взрывной интенсивности» – больше. Данные передаются в форме пакетов: [АДРЕС ПОЛУЧАТЕЛЯ, АДРЕС ОТПРАВИТЕЛЯ, ДАННЫЕ, КОНТРОЛЬНАЯ СУММА].

ШИНА – (+) Экономный расход кабеля. Относительно простая в использовании среда передачи данных. Легкая расширяемость и надежность. (-) При больших объемах трафика уменьшается пропускная способность сети. Выход кабеля из строя прерывает работу многих пользователей. КОЛЬЦО – (+) Увеличение количества пользователей не оказывает значительного влияния на производительность сети. (-) Выход из строя какого-нибудь компьютера может остановить работу всей сети. В динамике невозможно реконфигурировать сеть. ЗВЕЗДА – (+) Выход из строя какого-нибудь компьютера не влияет на работоспособность сети. Сеть легко модифицируема (-) Выход из строя центрального узла приводит сеть в нерабочее состояние. ЯЧЕЙКА – (+) Относительно просто локализовать и диагностировать отказы. Высокая надежность. (-) Высокая избыточность конфигурации требует больших затрат кабеля

1. Эталонная модель компьютерных сетей.

В 1978 г была создана эталонная модель взаимодействия обычных систем, называемая “семиуровневой моделью”. Идея состоит в том, что разделяются всевозможные просецессы, происходящие в сети во время сеанса связи, на семь функциональных уровней, строго организованных в соответствии с естественным следованием событий, происходящих на протяжении всего сеанса обмена информацией. Все уровни модели взаимодействуют строго иерархической системе - каждый уровень обслуживает уровни, находящиеся выше его , и пользуется услугами уровне, расположеных ниже. Иерархия следующая: физический, канальный, сетевой, транспортный, сеансовый, представления данных и прикладной.

Правила взаимодейстивия объектов одноименных уровней различных систем называют протоколами, правила взаимодействия объектов смежных уровней одной и той же системы - междууровневым интерфейсом.Все протоколы стандартизированны.

На верхем уровне взаимодействия осуществляются прикладные процессы и в модель включаются те ихчасти, которые связаны с передачей информации между системами. Основные части прикладных уровней связаны с хранением и обработкой информации. На нижнем уровне функции взаимодействия связаны с физической средой соединения.

В информационно - вычислительной среде используются 3 типа соединений: одностороннее, поочередно двухстороннее, одновременное двухстороннее.

****

1. Позиционные и непозиционные системы. Кодирование текстовых данных.

Позиционными называются системы счисления, в которых значение цифры зависит от ее места (позиции) в записи числа. Непозиционными называются системы счисления, в которых значение цифры не зависит от ее места (позиции) в записи числа. Позиционной является привычная для нас в повседневной жизни десятичная система счисления, в которой значение (вес) цифры зависит от ее позиции в записи числа. В числе 1111 одна и та же цифра 1 означает последовательно единицу, десяток, сотню, тысячу. Непозиционной системой счисления является, например, римская. Правила выполнения арифметических операций в непозиционных системах счисления совсем иные.

В машинно-ориентированной форме двоичные числа в компьютерах представляются в двух видах – с фиксированной и плавающей точкой. В форме с фиксированной точкой последняя устанавливается за определенным разрядом числа, отделяя его целую часть от дробной. Числа с фиксированной точкой масштабируются во избежание переполнения разрядной сетки во время их обработки. В форме с плавающей точкой числа представляются в виде N = m · g^p , где m – мантисса числа (правильная дробь (m < 1)); p – порядок числа (целое число); g – основание системы счисления. Порядок числа указывает положение точки в числе. При изменении порядка положение точки изменяется (она «плавает»). Если абсолютное значение мантиссы удовлетворяет условию 1 > |m| ≥ 1/g, то число называют нормализованным. В двоичной системе это условие запишется так: 1 > |m| ≥ 0,1, т.е. мантисса числа меньше единицы и первая значащая цифра непосредственно следует после запятой. 129 Если же в первой позиции после запятой записан нуль, то число называют ненормализованным. Представление чисел в нормализованном виде позволяет иметь в разрядной сетке большее число значащих цифр и, следовательно, повышает точность вычислений. Поэтому в разрядной сетке компьютеров хранятся нормализованные числа с плавающей точкой. Если в процессе вычислений получается ненормализованное число, то оно автоматически нормализуется путем сдвига мантиссы с соответствующим изменением ее порядка.

1. Система в коде вычетов.

Поиск новых систем кодирования данных преследует цель постоянно оптимизировать ресурсы, необходимые для обработки данных: память, процессорное время, средства надежности, точность вычислений и т.д. Не явилась исключением и система кодирования данных с применением вычетов, призванная ускорить арифметические операции, повысить точность и надежность вычислений.

Пусть P1, P2, ..., Pn – целые числа; Pi > 1, (Pi, Pj ) = 1, i ≠ j; M = (I = 1 to n)∏ Pi ; xi – наименьшие неотрицательные решения системы сравнений A ≡ xi (mod Pi ), i = 1, 2, ..., n; A ⊂ [0, M). Тогда кортеж (x1, x2, ..., xn) будем называть кодом числа A в системе в коде вычетов (СКВ) при заданных основаниях P1 , P2 , ..., Pn . Записывают это обычно так: A ∼ (x1, x2, ..., xn). Компоненты xi называют разрядами числа A в СКВ. Идейными корнями данная система восходит к так называемой китайской теореме об остатках.

Существуют алгоритмы арифметических операций на основе СКВ, алгоритмы перевода из позиционной системы счисления в СКВ и т.д.

• каждый разряд числа несет информацию обо всем числе, откуда следует независимость разрядов друг от друга;

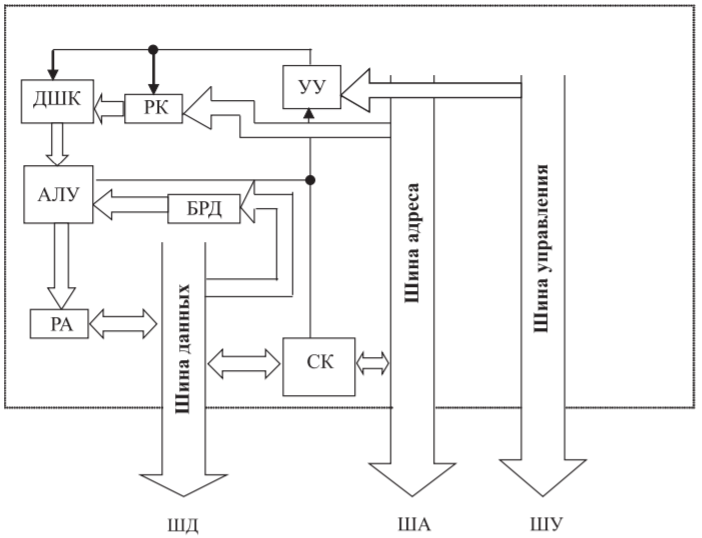
• отсутствие переносов между разрядами при выполнении арифметических операций упрощает их выполнение;

• малоразрядность компонент числа превращает арифметические операции в однотактные, иногда просто в выборку результата из таблицы.

Система в коде вычетов в основном применяется в специализированных вычислительных устройствах из-за высокой сложности операций определения знака числа и сравнения чисел.

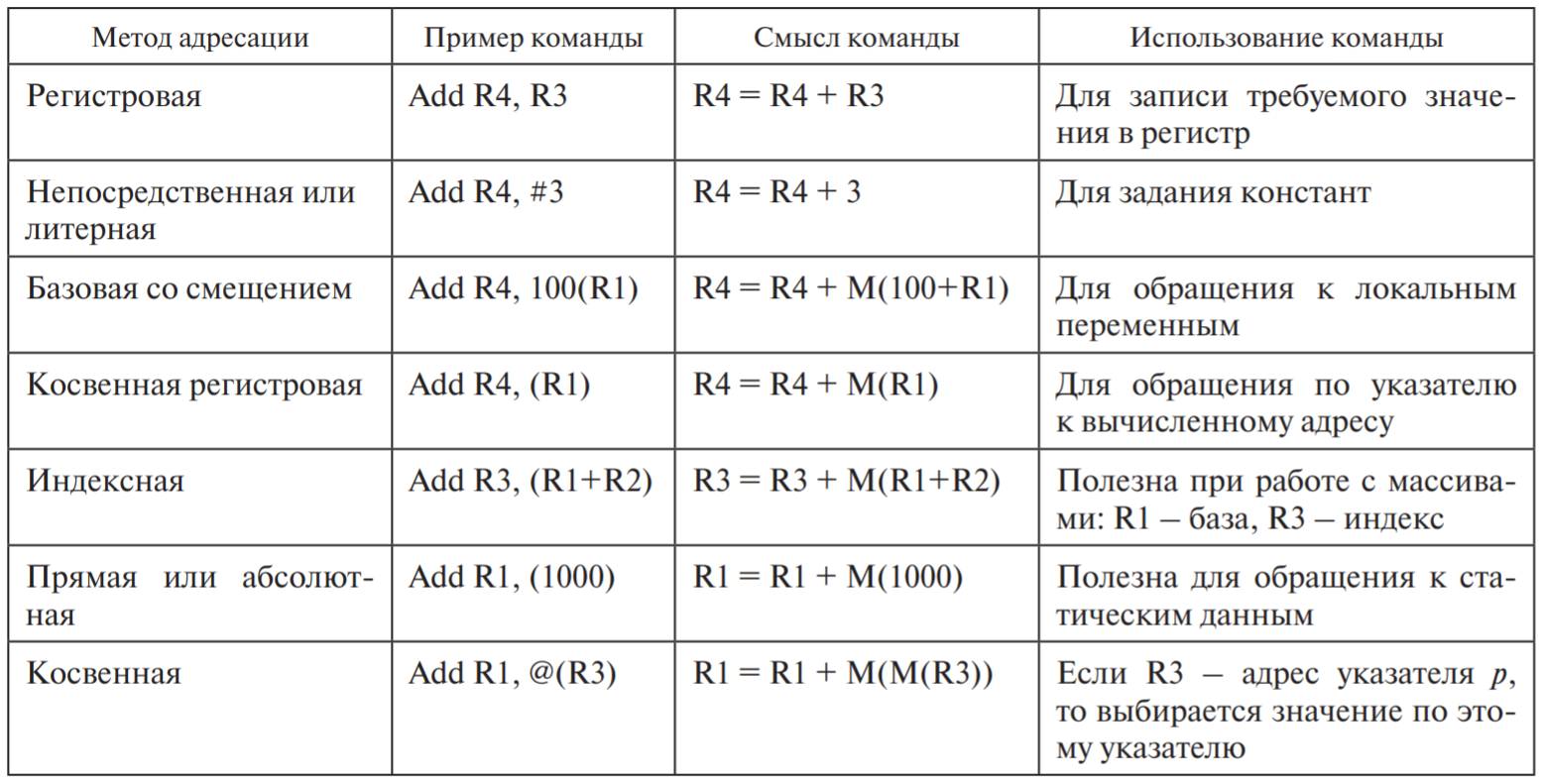
1. Структура микропроцессора.

Каждый микропроцессор может реализовать определенную систему команд (СК), которые хранятся в специальной области памяти. Считывание и выполнение команд осуществляется в соответствии с тактовыми импульсами от генератора сигналов. Управляющая программа задает адрес очередной команды для выполнения. Считанный из памяти код этой команды по шине адреса (ША) пересылается в регистр команд (РК). Дешифратор кода команд (ДШК) расшифровывает ее и посылает в АЛУ для исполнения. Обрабатываемые данные поступают на вход арифметико-логическое устройству через буферный регистр данных (БРД), используемый для временного хранения введенной информации. Результат обработки из арифметико-логическое устройства поступает в устройство оперативного хранения, в качестве которого используется регистр-аккумулятор (РА). Затем по шине данных (ШД) результат поступает на выход микропроцессор. В реальных микропроцессор оперативный регистр-аккумулятор является составной частью регистров общего назначения (РОН), являющихся внутренней сверхоперативной памятью микропроцессор.

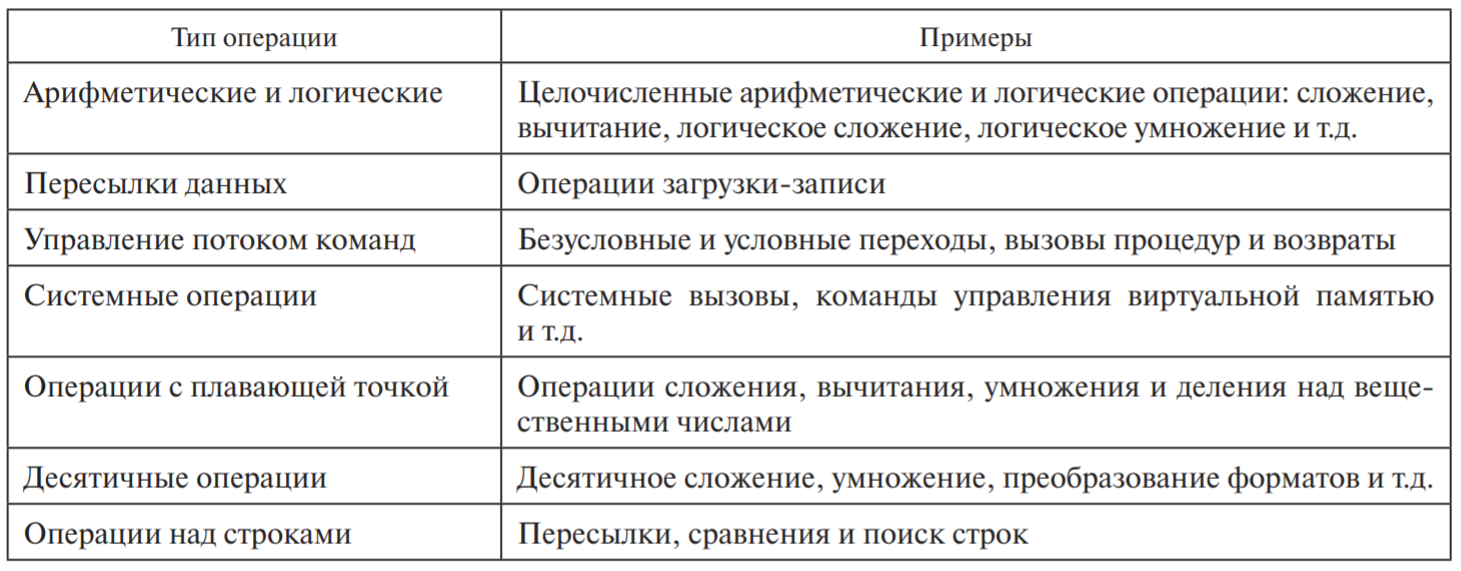


1. Типы команд и методы адресации.

В компьютерах с регистрами общего назначения метод (или режим) адресации объектов, с которыми манипулирует команда, может задавать константу, регистр или ячейку памяти. Система адресации определяет максимальный размер памяти, используемой компьютером. Использование сложных методов адресации позволяет существенно сократить количество команд в программе, но при этом значительно увеличивается сложность аппаратуры



Тип операнда может задаваться либо кодом операции в команде, либо с помощью тега, который хранится вместе с данными и интерпретируется аппаратурой во время обработки данных. Обычно тип операнда (целый, вещественный, символ) определяет и его размер.



1. Процессоры с микропрограммным управлением.

Для программиста язык ассемблерного типа является языком самого низкого уровня, а выполнение команды такого языка – элементарным шагом работы процессора. На самом деле команда состоит из ряда более элементарных подкоманд, последовательное выполнение которых и реализует соответствующую команду языка ассемблера. Операция, выполняющая каждую подкоманду, управляется отдельным функциональным импульсом. Известны два подхода к построению логики формирования функциональных импульсов: Каждой операции процессора соответствует набор логических схем, определяющих, какой функциональный импульс (ФИ) и в каком такте должен быть возбужден. Такой принцип управления операциями получил название «жесткой (запаянной) логики» и широко применяется во многих компьютерах. Каждой микрооперации (МИО) ставится в соответствие слово (или часть слова), называемое микрокомандой и хранимое в памяти подобно тому, как хранятся в памяти команды обычного компьютера. Здесь команде соответствует микропрограмма, т.е. набор микрокоманд (МИК), указывающих, какие функциональные импульсы и в какой последовательности необходимо возбуждать для выполнения данной операции. Такой подход получил название микропрограммирования или «хранимой логики».

Идея микропрограммирования, высказанная в 1951 г. М. Уилксом, до недавнего времени не находила широкого применения, ибо:

Формирование функционального импульса:

• не было надежных и быстродействующих запоминающих устройств для хранения микропрограммы;

• неправильно понимались задачи и выгоды микропрограммирования.

В последнее время интерес к микропрограммному принципу возродился, так как;

• созданы односторонние (читающие) быстродействующие ЗУ с малым циклом памяти;

• микропрограммирование рассматривается не как средство повышения гибкости программирования, а как метод построения системы управления процессором, удобный для инженера-разработчика компьютера. Существуют два вида микропрограммного управления – гориз. и вертик.

1. Проектирование архитектуры компьютера.

(Исследование концепций) В зависимости от задач выбираем: -CISC/RISC полный/сокращенный набор команд; -нейронную архитектуру, если задача плохо формализована№ -вантовую элем. Базу, если нужна высокая скорость;

(Анализ требований) Нужно определить сложность, функционал, разработать предварительный план, оценить стоимость, трудозатраты, время. Если функционал определен не точно можно потерять время и деньги на изменение проекта. Определить поддерживаемые языки программирования, требования к ОС, совместимость с другими архитектурами. (Составление спецификаций) Разработать критерии (с весовыми коэффициентами) определяющие стоимость, надежность, защиту, выбрать базу, если требование противоречивы, то найти компромисс (изменение известных решений) (создание новых решений) надо учесть затраты и желания заказчика(реинжиниринг) (разработанные структуры и функциональные схемы) решить, что реализовать программного, а что требует аппаратной поддержки, решить вопросы об иерархии памяти, типах и форматах данных, способах кодирования (отладка и оценка проекта) соединить части, оценить быстродействие и время разработки программ, проверить, уложились ли в отведенное время и реализуют ли нужный функционал. (Приемные испытание) (Сопровождение и эксплуатация) выявить и доработать некорректность и ошибки в работе (обучение персонала) (серийное производство) (вывод и эксплуатации)

Коллектив: при взятии в команду оценить: прежняя работа, участие в проектах, сможет ли уч в разных других, профессионализм, ответственность, стрессоустойчивость, ин. Языки, способность помочь, желание самосовершенствования. Невыполнение проекта – чтобы избегать нужно:

Оценивать ход выполнения, вовремя перераспределять работу, контролировать процесс.

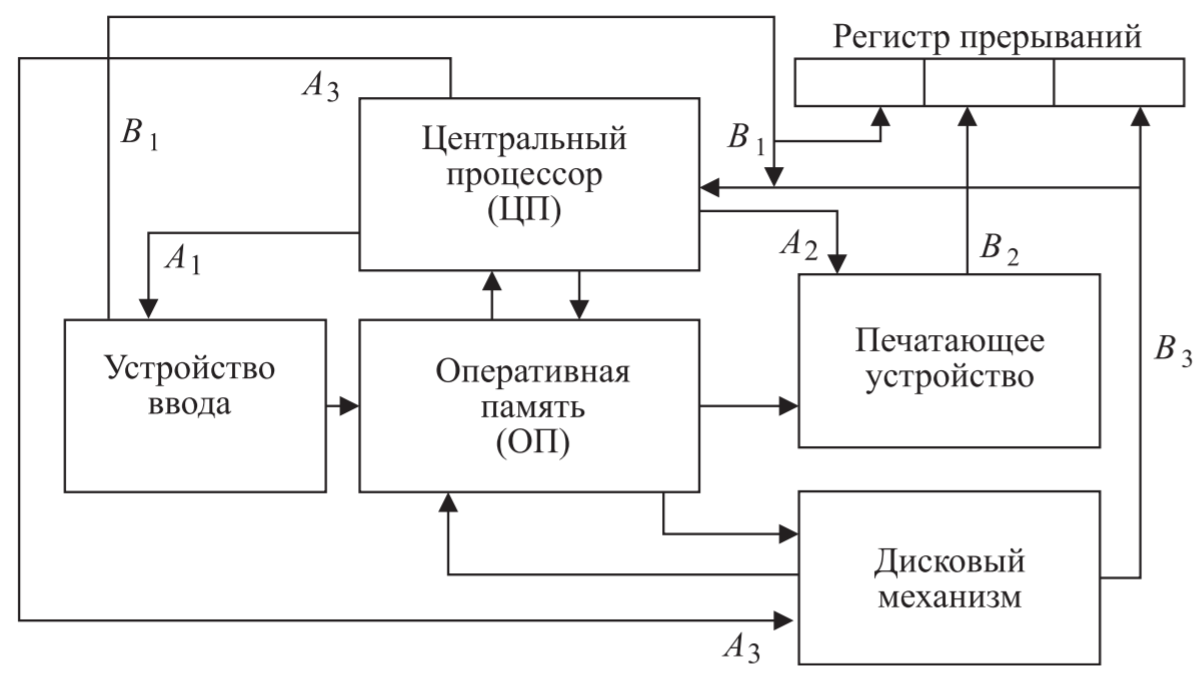
1. Графические процессоры.

Графический процессор (GPU – Graphics Processing Unit) – отдельное устройство персонального компьютера или игровой приставки, которое выполняет графический рендеринг и может рассматриваться в определенном смысле как многоядерная система. Задачи графики предполагают независимую параллельную обработку данных и изначально мультипоточны. Микроархитектура спроектирована таким образом, чтобы была возможность эксплуатировать имеющееся в наличии большое количество нитей, требующих исполнения. GPU состоит из нескольких десятков процессорных ядер (30 – для NVIDIA GT200, 20 – для «Evergreen», 16 – для «Fermi»), которые в терминологии NVIDIA называют Streaming Multiprocessor, а в терминологии ATI – SIMD Engine. Они исполняют несколько сотен программных нитей и реализуют почти те же процессы, что и обычный CPU. Современные графические процессоры эффективно обрабатывают и отображают компьютерную графику. Благодаря специализированной конвейерной архитектуре они намного эффективнее в обработке графической информации, чем типичный центральный процессор. Графический процессор в современных видеоадаптерах применяется в качестве ускорителя трехмерной графики, однако его можно использовать в некоторых случаях и для общих вычислений (GPGPU – General Purpose Graphics Processing Units – GPU общего назначения).

Ведущие производители видеокарт предложили свои реализации GPGPU. Однако разработки в области GPGPU не слишком популярны: во-первых, потому что программировать под GPU довольно сложно (графические процессоры не поддерживают стек, поэтому в программах не должно быть рекурсии, есть проблемы и с вызовами функций), а во-вторых, существует не так много задач, которые эффективно решаются с помощью GPU.

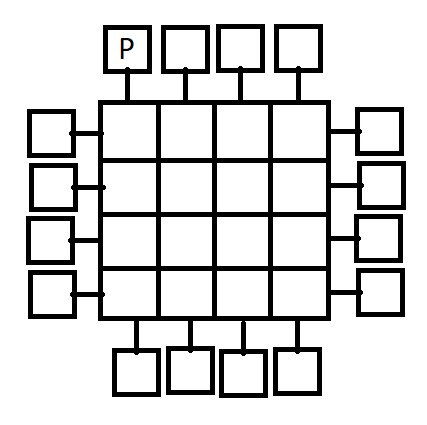
1. Управление процессами в многопроцессорных системах.

Рассмотрим абстрактную вычислительную систему с оперативной памятью, одним центральным процессом, устройством ввода и устройством печати. Пусть в исходном состоянии все периферийные устройства (ПУ) находятся в состоянии покоя, а центральный процессор выполняет процесс Pi(итое), управляемый последовательностью команд. Если в момент ti(итое) этот процесс выдает код, возбуждающий процесс в периферийном устройстве, то центральный процесс переходит в управляющий режим и посылает сигнал по линии связи A1, A2 или A3. Затем центральный процесс возвращается к выполнению своего процесса. По окончании работы одно из периферийного устройства посылает сигнал прерывания по линии Bi(итое), устанавливая тем самым i-й разряд регистра прерываний в соответствующее состояние. Центральный процесс переходит в управляющий режим, анализирует состояние регистра прерываний для опознания периферийного устройства, вызвавшего прерывания, и выполняет соответствующие действия. До перехода в режим управления центрального процесса запоминает состояние прерываемого процесса. Если во время обработки прерывания приходит новое прерывание, обработка предыдущего прерывания будет доведена до конца.



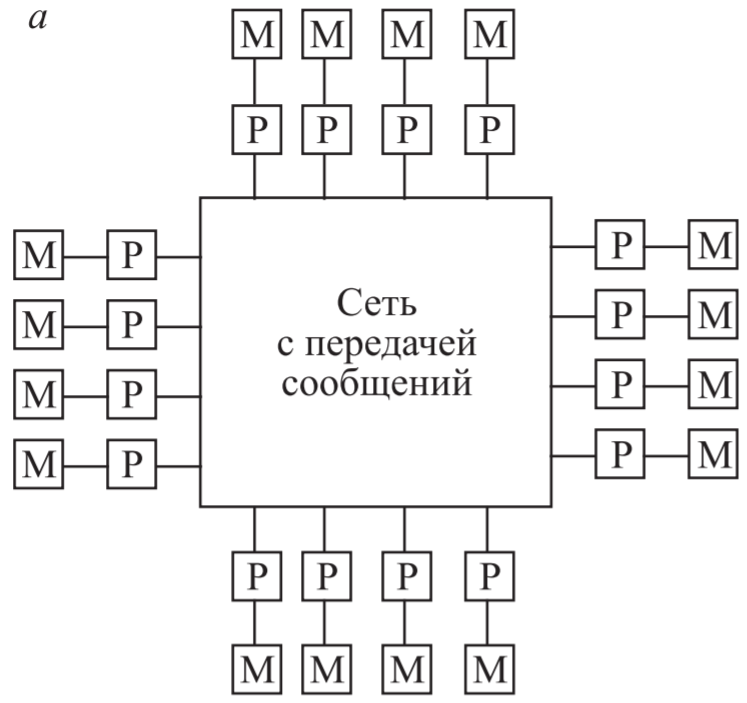
1. Мультипроцессоры.

Если все процессоры разделяют общую физическую память, то такая система называется мультипроцессором или системой с совместно используемой памятью. Мультипроцессорная модель распространяется и на программное обеспечение. Все процессы, работающие вместе на мультипроцессоре, могут разделять одно виртуальное адресное пространство, отображенное в общую память. Любой процесс может считывать слово из памяти или записывать слово в память с помощью команд Load и Store. Обычно выделяют три вида мультипроцессоров в зависимости от механизма реализации памяти совместного использования: архитектуры UMA (Uniform Memory Access – с однородным доступом к памяти), NUMA (Non Uniform Memory Access – с неоднородным доступом к памяти) и COMA (Cache Only Memory Access – с доступом только к кэш-памяти). Архитектура UMA обеспечивает одно и то же время доступа к слову вне зависимости от расположения его внутри модуля и расстояния модуля от процессора. Эта архитектура с предсказуемой производительностью. В архитектуре NUMA доступ к данным существенно зависит от расстояния процессора до модуля с необходимыми данными. Следовательно, расположение данных существенно влияет на производительность мультипроцессора. В обоих случаях обеспечивается единое адресное пространство, видимое для всех процессоров, однако программы, подготовленные для архитектуры UMA, на архитектуре NUMA выполняются медленнее. Архитектура NUMA может быть как с кэш-памятью (CC-NUMA – Coherent Cache NUMA), так и без нее (NC-NUMA – No Caching NUMA). Наличие кэш-памяти немного сглаживает разницу во времени доступа к информации в разных модулях. Однако, если объем кэш-памяти существенно меньше объема требуемых данных, производительность мультипроцессора снижается. В архитектуре COMA память каждого процессора используется как кэш-память. В ней все физическое адресное пространство делится на строки, которые мигрируют по системе по мере необходимости (attraction memory). Такая организация повышает частоту обращений к кэш-памяти, повышая тем самым производительность системы.



1. Мультикомпьютеры.

Параллельную архитектуру, при которой каждый процессор имеет свою собственную память, доступную только этому процессору, называют мультикомпьютером или системой с распределенной памятью **(СХЕМА)**. Мультикомпьютеры обычно являются системами со слабой связью. Ключевое отличие мультикомпьютера от мультипроцессора состоит в том, что мультипроцессоры имеют одно физическое адресное пространство, а мультикомпьютеры содержат отдельное физическое адресное пространство для каждого центрального процессора. Так как процессоры в мультикомпьютере не могут взаимодействовать друг с другом путем чтения из общей памяти и записи в общую память, здесь необходим другой механизм взаимодействия. Они посылают друг другу сообщения, используя сеть межсоединений. В мультикомпьютере для взаимодействия между процессорами часто используются примитивы Send и Receive. Поэтому программное обеспечение мультикомпьютера имеет более сложную структуру, чем у мультипроцессора, следовательно его сложнее программировать. Стали предприниматься попытки найти компромисс между мультипроцессорами и мультикомпьютерами, здесь важно получить такую систему, которая была бы расширяема. Первый подход основан на том, что современные компьютерные системы не монолитны, а состоят из ряда уровней. Это дает возможность реализовать общую память на любом из нескольких уровней. Второй подход – DSM (Distributed Shared Memory − распределенная совместно используемая память) предлагает использовать аппаратное обеспечение мультикомпьютера и операционную систему, которая моделирует разделенную память, создавая единое виртуальное адресное пространство, разбитое на страницы. Третий подход − реализовать общую разделенную память на уровне программного обеспечения.



1. Метрики аппаратного и программного обеспечения.

Главным вопросом в метрике СПД в плане аппаратного обеспечения являются параметры функционирования системы межсоединений: пропускная способность и время ожидания. Полное время ожидания – время на отправку пакета и получения ответа. При посылке пакета в память это время чтения и записи в память, при пересылке «процессор – процессору» – время процессорной связи для пакетов минимального размера (одного слова). Для коммутации каналов время ожидания состоит из времени установки и времени передачи. Пакетная коммутация не требует посылки пробного пакета в пункт назначения, но для нее необходимо некоторое время Tk на коммутацию пакета. В случае коммутации без буферизации пакетов «wormhole routing» («червоточина») время передачи пакета в наилучшем случае будет приближаться к Tk+ p/b, ибо отсутствует как посылка пробных пакетов для установки схемы, так и задержка для промежуточного хранения.

Очень важное значение имеет возможность быстро передавать большие объемы информации. Это свойство отражает суммарная пропускная способность, определяемая суммированием пропускной способности всех каналов связи.

Что касается ПО. Литературные источники по СПД утверждают, что n-кратного увеличения быстродействия нельзя достичь на СПД с n процессорами. Наибольшее ускорение возможно при решении N-объектных задач. Например, задачу инвертирование матриц нельзя ускорить более, чем в пять раз, вне зависимости от числа процессоров. И для этого есть ряд причин. Во-первых, все программы содержат последовательную часть: считывание данных, инициализацию, интегрирование результата. Пусть линейная часть есть f, а параллельная 1 – f. Тогда если на однопроцессорном компьютере время решения задачи T, то общее время решения задачи не может быть лучше fT + (1 – f) T/n. Только при f = 0 возможно линейное ускорение. Это явление называют законом Амдала. Во-вторых, невозможность линейного ускорения связана со временем ожидания в коммуникациях, ограниченной пропускной способностью, неэффективными алгоритмами, непроизводительными задержками для запуска большого количества процессоров.

Если увеличение количества процессоров ведет к повышению производительности системы, то ее называют расширяемой.

1. Архитектура суперкомпьютеров.

Особое место среди систем параллельного действия занимают суперкомпьютеры. Их высокие характеристики достигаются благодаря успехам в области микропроцессорных и коммуникационных технологий, расширению сфер применения графических процессоров и появлению новых трудоемких задач.

Архитектура суперкомпьютера СКИФ. Это двухуровневая открытая масштабируемая архитектура, включающая в себя кластерный и потоковый уровни. На кластерном уровне использована технология SCALI. СКИФ –это 32-процессорный кластер, системная сеть которого построена на основе SCI (по топологии 2D тор 4×4 – квадратная матрица 4×4, где каждый узел связан с четырьмя соседними), а вспомогательная сеть, служащая для управления кластером с головного компьютера, базируется на технологии Switched Fast Ethernet. На потоковом уровне используется однородная вычислительная среда с топологией плоской решетки из большого числа микропроцессоров, объединенных потоковыми линиями передачи данных. Открытая и масштабируемая архитектура не накладывает жестких ограничений на программно-аппаратную платформу узлов кластера, топологию вычислительной сети, конфигурацию и диапазон производительности суперкомпьютеров. Такой подход позволяет создавать прикладные суперкомпьютерные системы в широком диапазоне производительности.

1. Организация системы прерывания.

Работу вычислительной системы можно представить как последовательность событий программно-определяемых (порождаемых программой, причем возможные моменты их появления известны) и программно-независимых (вызванных посторонними для программы источниками, причем моменты возникновения их могут быть неизвестны). События, происходящие вне процессора, как правило, программно-независимые и происходят асинхронно. События, происходящие внутри процессора, могут быть двух типов. Последовательность арифметических операций определяется программой, в то время как особые ситуации (переполнение, попытка деления на нуль и т.д.) зависят от сочетания операндов и предусмотреть их во время программирования практически невозможно. Чтобы ВС могла реагировать на программно-независимые события при минимальных усилиях программиста и максимально возможном быстродействии, ее надо снабдить дополнительными аппаратно-логическими средствами, совокупность которых называют системой прерывания программ (СПП). Прерывание программы – это свойство ВС при возникновении особых событий временно прекратить выполнение текущей программы и передать управление программе, специально предусмотренной для обработки данного события. Системы прерывания программ, как правило, выполняют следующие основные функции:

• организуют вход в прерывающую программу;

• осуществляют приоритетный выбор между запросами прерывания;

• обеспечивают возврат к прерванной программе и программное изменение приоритетов программ.

Для сравнения различных СПП используются чаще всего следующие параметры их функционирования:

1. Время реакции (t р ) – время между появлением запроса на прерывание и началом выполнения первой команды прерывающей программы;

2. Время обслуживания прерывания (t обс) – разность между полным временем выполнения прерывающей программы (t пр) и временем выполнения всех полезных команд (t п).

3. Удельный вес прерывающих программ η=tп/tпр.

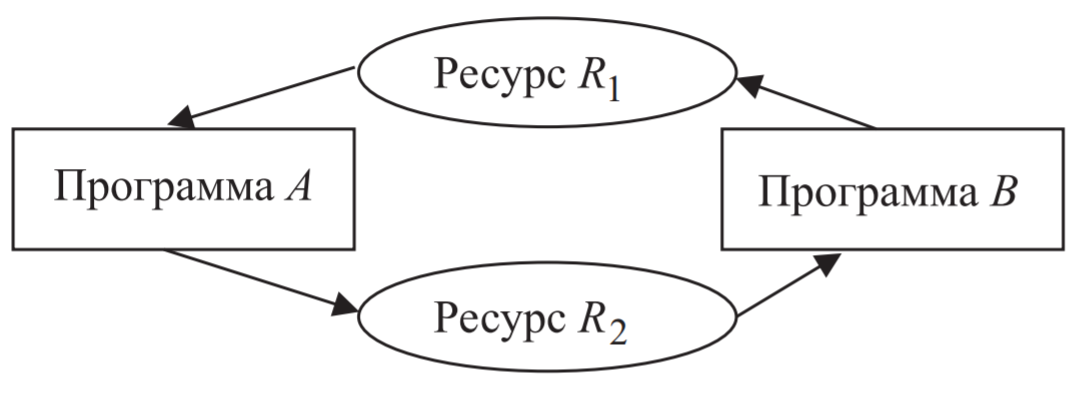
4. Глубина прерывания (n) – максимальное число программ, которые могут прерывать друг друга.

1. Механизмы синхронизации процессов.

Процессы, выполняемые в мультипрограммном режиме, можно рассматривать как набор последовательных слабосвязанных процессов, которые действуют почти независимо друг от друга, лишь изредка используя общие ресурсы. Взаимосвязь между такими процессами устанавливается с помощью различных сообщений и так называемого механизма синхронизации, который позволяет согласовывать и координировать работу процессов.

П р и м е р. Пусть программе А нужен ресурс R1. Она запрашивает его и получает. Программе В нужен ресурс R2. Она запрашивает его и получает. Далее пусть программа А, не отпуская R1, запрашивает R2, а программа В, не отпуская R2, запрашивает R1. Налицо типичный клинч, если только один из ресурсов R1 или R2 не может быть освобожден до момента завершения обработки соответствующего процесса.

Для разрешения подобных проблем наиболее часто используют простейшие приемы синхронизации процессов, тесно связанные с аппаратным оборудованием. К таким приемам относятся стандартные операции типа Wait («Ждать») и Signal («Оповестить»): Wait позволяет временно заблокировать процесс, а Signal информирует систему о необходимости разблокирования процесса, задержанного из-за невыполнения условия. Следует отметить такие приемы, как «Блокировка памяти» (для реализации взаимного исключения одному процессу разрешается выполнить операцию над памятью, а другому ждать, пока первый не завершит работу), «Проверка» и «Установка». Наиболее эффективным и простым средством синхронизации процессов, исключающим состояние «активного» ожидания, является семафор. Среди базовых элементов синхронизации можно назвать семафоры, блокировки, мьютексы и критические секции. Данные примитивы действуют по одинаковому принципу: разрешают использовать одному процессу какой-то ресурс, при этом никакие другие процессы к этому ресурсу доступа не имеют.



1. Планирование процессов.

Все задания (процессы), находящиеся в системе с мультипрограммированием, конкурируют из-за процессорного времени. Кроме процессов пользователя имеются и системные процессы, для выполнения которых нужно процессорное время. Существует, как правило, две очереди процессов:

• очередь готовых процессов – активные процессы, ожидающие кванта времени центрального процессора (Ог);

• очередь заблокированных процессов, ожидающих выполнения какого-нибудь события (Об). Пусть P – процессор. Каждый процесс pi может находиться в одном из состояний: готовом, заблокированном или выполняемом. Если pi ∈ Ог и ему выделен квант t pi процессорного времени, то он будет выполняться, пока не произойдет одно из событий:

• истечет его квант времени;

• процесс будет заблокирован;

• процесс будет вытеснен более приоритетным процессом;

• процесс будет завершен. Каждому pi ∈ Ог поставлен в соответствие приоритет – целое число, учитывающее важность процесса pi , занимаемый им объем памяти, срочность выполнения и объем I/О, а также внешний приоритет, назначаемый пользователем. Такие системные процессы, как учет, управление стандартными программами, выполняются так же, как и задания пользователей. Все алгоритмы планирования должны учитывать привилегированные процессы, а также тот факт, что ни один процесс в системе не должен ждать бесконечно долго. Система должна планировать процессы так, чтобы свести к минимуму средние потери или ограничить максимальные потери по всем заданиям, что не всегда согласуется с эффективным использованием всех компонент системы.

1. Нейрокомпьютеры.

При проектировании любого вычислительного устройства учитывается класс задач, который наиболее эффективно будет на нем решен.

Недостаточная производительность современных вычислительных машин является фактором, сдерживающим развитие многих научных и прикладных исследований. Сегодня одним из важнейших свойств компьютерной системы считается ее масштабируемость, которая охватывает не только возможность наращивания числа и мощности процессоров, объемов оперативной и внешней памяти, других ресурсов системы и способов их объединения, обеспечиваемых архитектурой и конструкцией компьютера, но определяется и соответствующими программными средствами. В последние несколько лет наблюдается взрыв интереса к нейронным сетям (НС), которые успешно применяются в самых различных областях – статистике, экономике и финансах, медицине, технике, геологии, физике. Нейронные сети вошли в практику во многих отраслях человеческой деятельности, где нужно решать задачи прогнозирования, классификации или управления. Такой успех определяется, прежде всего, широкими возможностями нейронных сетей – мощного метода моделирования, позволяющего воспроизводить чрезвычайно сложные зависимости. Нейронные сети – нелинейные по своей природе. На протяжении многих лет в большинстве областей применялось линейное моделирование, поскольку для него хорошо разработаны процедуры оптимизации. В задачах, где линейная аппроксимация неудовлетворительна, линейные модели работают неудовлетворительно. Кроме того, нейронные сети позволяют моделировать линейные зависимости в случае большого числа переменных. Применение нейронных сетей отличается также простотой. Нейронные сети обучаются на примерах. Пользователь нейронной сети подбирает представленные данные, а затем запускает алгоритм обучения, который автоматически воспринимает структуру данных. При этом от пользователя, конечно, требуется набор эвристических знаний о том, как следует отбирать и готовить данные, выбирать нужную архитектуру сети и интерпретировать результаты.

1. Производительность вычислительных систем.

Что касается нейрокомпьютеров, то такой класс задач представляет собой не только неформализуемые или плохо формализуемые задачи, в алгоритм решения которых включается процесс обучения на реальном экспериментальном материале (как правило, задачи распознавания образов), но и задачи с естественным параллелизмом (например, обработка изображений). Нейрокомпьютер - вычислительная система шестого поколения, состоящая из большого числа параллельно работающих простых вычислительных элементов (нейронов). Элементы связаны между собой, образуя нейронную сеть. Они выполняют единообразные вычислительные действия и не требуют внешнего управления. Большое число параллельно работающих вычислительных элементов обеспечивают высокое быстродействие.

Нейрокомпьютеры, в отличие от классической ЭВМ предыдущих поколений, не просто имеют большие возможности, но и принципиально по-другому используются. Место программирования занимает обучение посредством корректировки весов связей, в результате чего каждое входное воздействие приводит к формированию соответствующего выходного сигнала. После обучения сеть может применять полученные навыки к новым входным сигналам. При переходе от программирования к обучению повышается эффективность решения плохо формализуемых задач. Таким образом, нейрокомпьютеры используют не заранее разработанные алгоритмы, а специальным образом подобранные примеры, на которых обучаются. В нейронной сети нет локальных областей, в которых запоминается конкретная информация. Она запоминается во всей сети.

Можно сформулировать следующие свойства нейрокомпьютеров:

* параллельная работа большого количества простых вычислительных устройств;
* способность к обучению, осуществляемая через настройки параметров сети;
* высокая помехо и отказоустойчивость;
* новые физические принципы обработки информации для аппаратных реализаций нейронных сетей в силу простого строения отдельных нейронов.

Разработки в области нейрокомпьютинга ведутся по следующим направлениям:

* разработка нейроалгоритмов,
* создание специализированного программного обеспечения для моделирования нейронных сетей,
* разработка специализированных процессорных плат для имитации нейросетей,
* электронные и оптоэлектронные реализации нейронных сетей.

1. Квантовые процессоры.

Квантовый компьютер как и обычный работает с нулями и единицами, но его функциональные элементы производят действие в фазовом пространстве некоторой квантовой системы, используя унитарные преобразования этого пространства.

Различные типы квантовых компьютеров строятся используя квантовые явления но разного порядка. Среди них можно указывать квантование магнитного потока на нарушение сверхпроводимости-Джозефсоновских переходах или поддержание когерентности волновых функций используемых кубитами, в течение всего времени вычислений.



В качестве основных технологий, на которых проектируются квантовые компьютеры, выделяют следующие:

1. Твердотельные квантовые точки на полупроводниках: в качестве логических кубитов используются либо зарядовые состояния (нахождение или отсутствие электрона в определенной точке) либо направление электронного и/или ядерного спина в данной квантовой точке. Управление через внешние потенциалы или лазерным импульсом.

2. Сверхпроводящие элементы (джозефсоновские переходы, сквиды и др.). В качестве логических кубитов используются присутствие/отсутствие куперовской пары в определенной пространственной области. Управление: внешний потенциал/магнитный поток.

3. Ионы в вакуумных ловушках Пауля (или атомы в оптических ловушках). В качестве логических кубитов используются основное/возбужденное состояния внешнего электрона в ионе. Управление: классические лазерные импульсы вдоль оси ловушки или направленные на индивидуальные ионы+ колебательные моды ионного ансамбля.

4. Смешанные технологии: использование заранее приготовленных запутанных состояний фотонов для управления атомными ансамблями или как элементы управления классическими вычислительными сетями.

1. Области применения квантовых компьютеров.

Во-первых, на квантовом компьютере можно моделировать любую квантовую систему за полиномиальное число шагов. Это позволит предсказывать свойства молекул и кристаллов, проектировать микроскопические электронные устройства размером в несколько десятков ангстрем. Сейчас такие устройства находятся на пределе технологических возможностей, но в будущем они, вероятно, будут применяться в обычных компьютерах.

Второй пример – разложение на множители и аналогичные теоретико-числовые задачи, связанные с абелевыми группами. Ранее уже упоминалось, что в 1994 г. П.Шор придумал квантовый алгоритм, позволяющий разложить на простые множители число из n знаков за полиномиальное от n число шагов. Этот результат может существенно усложнить работу криптологов: разлагая числа на множители, можно подбирать ключи к шифрам, подделывать электронные подписи и т. д. Применение идей квантовой механики уже открыли новую эпоху в области криптографии, так как методы квантовой криптографии открывают новые возможности в области передачи сообщений. Прототипы систем подобного рода находятся на стадии разработки.

Третий пример применения квантового компьютера – это поиск нужной записи в неупорядоченной базе данных. Здесь ускорение не столь значительно: для нахождения одной записи из N требуется порядка операций на квантовом компьютере вместо N операций на классическом.

Помимо алгоритма Шора, известны и такие именные квантовые алгоритмы.

· Алгоритм Гровера позволяет найти решение уравнения f(x) = 1, 0 < x < N, за время 0().

· Алгоритм Залки – Визнера позволяет моделировать унитарную эволюцию квантовой системы из n частиц за почти линейное время с использованием 0(n) кубит.

· Алгоритм Дойча – Джоза позволяет «за одно вычисление» определить, является ли функция двоичной переменной f(n) постоянной (f1(n) = 0, f2(n) = 1 независимо от n) или «сбалансированной» (f3(0) = 0, f3(1) = 1; f4(0) = 1, f4(1) = 0).

Основные проблемы, связанные с созданием и применением квантовых компьютеров:

· необходимо обеспечить высокую точность измерений;

· внешние воздействия могут разрушить квантовую систему или внести в ее работу существенные искажения.

Физическая реализация квантового компьютера очень сложная задача, так как необходимо удовлетворить нескольким почти несовместимым требованиям:

· элементы квантового компьютера – квантовые биты (спины или что-то подобное) – должны быть изолированы друг от друга и от окружающей среды.

· необходимо иметь возможность избирательного воздействия на пару квантовых битов.

· каждый из унитарных операторов должен быть реализован с достаточной точностью.

1. Коммуникационная среда SCI.

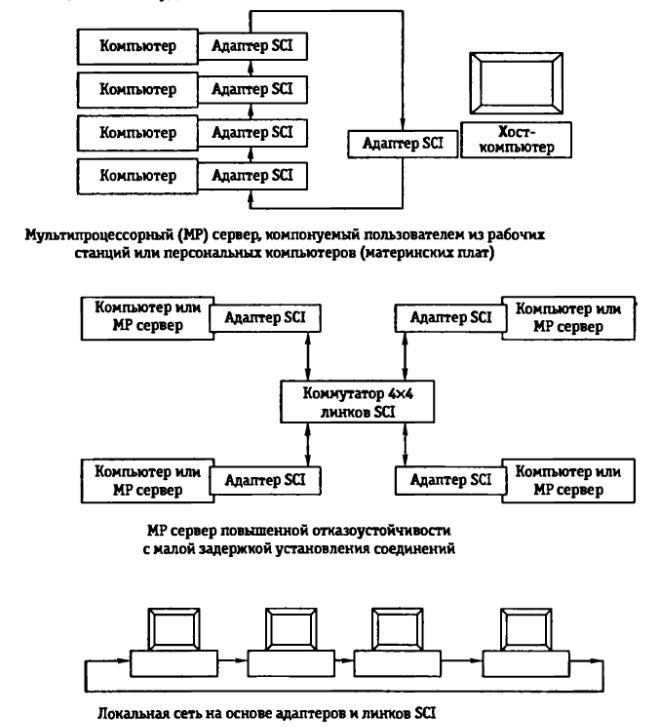
Масштабируемый когерентный интерфейс (Scalable Coherent Interface) принят как стандарт ANSI/IEEE Std 1596-1992. Для обозначения этого стандарта общепринято сокращение SCI.

SCI предусматривает реализацию когерентности посредством стандартно организованной кэш-памяти, размещаемой в узле SCI. Эта кэш-память располагается в интерфейсе между вычислительным модулем (BM) и узлом и вписывается в механизм реализации когерентности ВМ следующим образом:

если на предыдущих уровнях иерархии памяти ВМ обнаруживается отсутствие необходимых данных, то производится их поиск в кэш-памяти узла;

\*при нахождении данных, если последние состоятельны, их копия перемещается внутрь иерархии памяти ВМ. Если же данные находятся в состоянии модификации, запись осуществляется после ее завершения;

\*при отсутствии данных вырабатывается сигнал промаха, который активизирует адаптер интерфейса на выполнение действий по доставке данных с удаленных блоков памяти.



Стандарт SCI обеспечивает построение легкой в реализации, масштабируемой, эффективной в стоимостном аспекте коммуникационной среды для объединения процессоров и памяти, создания распределенной сети рабочих станций, организации ввода/вывода суперЭВМ, высокопроизводительных серверов и рабочих станций на базе микропроцессоров. Стандарт предусматривает наличие пропускной способности не менее 1 Гбайт/с для сосредоточенных систем и не менее 1 Гбайт/с для распределенных систем типа сети рабочих станций. Каждый узел имеет входной и выходной каналы. Узлы связаны однонаправленными каналами «точка-точка» либо с соседним узлом, либо подключены к коммутатору. При объединении узлов должна обязательно формироваться циклическая магистраль (кольцо) из узлов, соединенных каналами «точка-точка», между входным и выходным каналами каждого узла. Один из узлов в кольце, называемый scrubber,

выполняет следующие функции: инициализации узлов кольца с установлением адресов, управления таймерами, уничтожения пакетов, не нашедших адресата. Этот узел отмечает проходящие через него пакеты и уничтожает уже помеченные пакеты. В кольце может быть только один scrubber.