**1 ВВЕДЕНИЕ**

**1.1 Архитектура, организация и конфигурация ЭВМ**

*Архитектура ЭВМ*— абстрактное определение ЭВМ в терминах основных функциональных модулей, языков программирования и структур данных. Архитектура не определяет особенности реализации аппаратной части ЭВМ, времени выполнения команд, степени параллелизма, ширины шин и других аналогичных характеристик. Архитектура отображает аспекты структуры ЭВМ, которые являются видимыми для пользователя-программиста: систему команд, режимы адресации, форматы данных, набор программно-доступных регистров. Одним словом, термин «архитектура» используется для описания возможностей, предоставляемых ЭВМ.  
  
Термин *организация ЭВМ*определяет описание того, как реализованы возможности ЭВМ.  
  
Весьма часто употребляется термин *конфигурация ЭВМ,*под которым понимается компоновка вычислительного устройства с четким определением характера, количества, взаимосвязей и основных характеристик его функциональных элементов.  
  
Таким образом, по цепочке *архитектура → организация → конфигурация* мы переходим от абстрактного определения к описанию конкретной ЭВМ.  
  
Термин «архитектура системы» часто употребляется как в узком, так и в широком смысле этого слова. В узком смысле под архитектурой понимается архитектура набора команд. Архитектура набора команд служит границей между аппаратурой и программным обеспечением и представляет ту часть системы, которая видна программисту или разработчику компиляторов. Следует отметить, что это наиболее частое употребление этого термина. В широком смысле архитектура охватывает понятие организации системы, включающее такие высокоуровневые аспекты разработки компьютера как систему памяти, структуру системной шины, организацию ввода/вывода и т.п.   
  
Применительно к вычислительным системам термин «архитектура» может быть определен как распределение функций, реализуемых системой, между ее *уровнями*, точнее как определение границ между этими уровнями. Таким образом, архитектура вычислительной системы *предполагает многоуровневую организацию*.  
  
Архитектура первого уровня определяет, какие функции по обработке данных выполняются системой в целом, а какие возлагаются на внешний мир (пользователей, операторов, администраторов баз данных и т.д.). Система взаимодействует с внешним миром через *набор интерфейсов*: языки (язык оператора, языки программирования, языки описания и манипулирования базой данных, язык управления заданиями) и *системные программы* (программы-утилиты, программы редактирования, сортировки, сохранения и восстановления информации).   
  
Интерфейсы следующих уровней могут разграничивать определенные уровни внутри программного обеспечения. Например, *уровень управления логическими ресурсами* может включать реализацию таких функций, как управление базой данных, файлами, виртуальной памятью, сетевой телеобработкой. К *уровню управления физическими ресурсами* относятся функции управления внешней и оперативной памятью, управления процессами, выполняющимися в системе.   
  
Следующий уровень отражает основную линию разграничения системы, а именно *границу между системным программным обеспечением и аппаратурой*. Эту идею можно развить и дальше и говорить о распределении функций между отдельными частями физической системы. Например, некоторый интерфейс определяет, какие функции реализуют центральные процессоры, а какие - процессоры ввода/вывода.  
  
Архитектура следующего уровня определяет разграничение функций между процессорами ввода/вывода и контроллерами внешних устройств. В свою очередь можно разграничить функции, реализуемые контроллерами и самими устройствами ввода/вывода (терминалами, модемами, накопителями на магнитных дисках и лентах). Архитектура таких уровней часто называется *архитектурой физического ввода/вывода*.   
^

**1.2 Общие требования к современным компьютерам**

**1.2.1 Отношение стоимость/производительность**

У разработчиков компьютеров нет одной единственной цели. Для достижения поставленных целей при проектировании высокопроизводительных конструкций приходится игнорировать стоимостные характеристики. Суперкомпьютеры фирмы Cray Research и высокопроизводительные мейнфреймы компании IBM относятся именно к этой категории компьютеров.  
  
Другим крайним примером может служить низкостоимостная конструкция, где производительность принесена в жертву для достижения низкой стоимости. К этому направлению относятся персональные компьютеры различных клонов IBM PC.  
  
Между этими двумя крайними направлениями находятся конструкции, основанные на отношении *стоимость/производительность*, в которых разработчики находят баланс между стоимостными параметрами и производительностью. Типичными примерами такого рода компьютеров являются миникомпьютеры и рабочие станции.   
  
Для сравнения различных компьютеров между собой обычно используются стандартные методики измерения производительности. Эти методики позволяют разработчикам и пользователям использовать полученные в результате испытаний количественные показатели для оценки тех или иных технических решений, и в конце концов именно производительность и стоимость дают пользователю рациональную основу для решения вопроса, какой компьютер выбрать.   
^

**1.2.2 Надежность и отказоустойчивость**

Важнейшей характеристикой вычислительных систем является *надежность*. Повышение надежности основано на принципе предотвращения неисправностей путем снижения интенсивности отказов и сбоев за счет применения электронных схем и компонентов с высокой и сверхвысокой степенью интеграции, снижения уровня помех, облегченных режимов работы схем, обеспечение тепловых режимов их работы, а также за счет совершенствования методов сборки аппаратуры.   
  
*Отказоустойчивость* - это такое свойство вычислительной системы, которое обеспечивает ей, как логической машине, возможность продолжения действий, заданных программой, после возникновения неисправностей.  
  
Введение отказоустойчивости требует *избыточного аппаратного и программного обеспечения*. Направления, связанные с предотвращением неисправностей и с отказоустойчивостью, - основные в проблеме надежности. Концепции параллельности и отказоустойчивости вычислительных систем естественным образом связаны между собой, поскольку в обоих случаях требуются дополнительные функциональные компоненты. Поэтому, собственно, на параллельных вычислительных системах достигается как наиболее высокая производительность, так и очень высокая надежность. Имеющиеся ресурсы избыточности в параллельных системах могут гибко использоваться как для повышения производительности, так и для повышения надежности. Структура многопроцессорных и многомашинных систем приспособлена к *автоматической реконфигурации* и обеспечивает возможность продолжения работы системы после возникновения неисправностей.   
  
Следует помнить, что понятие надежности включает не только аппаратные средства, но и программное обеспечение. Главной целью повышения надежности систем является целостность хранимых в них данных. 

**1.2.3 Масштабируемость**

*Масштабируемость* представляет собой возможность *наращивания* числа и мощности процессоров, объемов оперативной и внешней памяти и других ресурсов вычислительной системы.  
  
Масштабируемость должна обеспечиваться *архитектурой и конструкцией* компьютера, а также соответствующими *средствами программного обеспечения*. Добавление каждого нового процессора в действительно масштабируемой системе должно давать прогнозируемое увеличение производительности и пропускной способности при приемлемых затратах.  
  
Одной из основных задач при построении масштабируемых систем является минимизация стоимости расширения компьютера и упрощение планирования. В идеале добавление процессоров к системе должно приводить к линейному росту ее производительности. Однако это не всегда так. Потери производительности могут возникать, например, при недостаточной пропускной способности шин из-за возрастания трафика между процессорами и основной памятью, а также между памятью и устройствами ввода/вывода. В действительности реальное увеличение производительности трудно оценить заранее, поскольку оно в значительной степени зависит от динамики поведения прикладных задач.   
  
Масштабируемость программного обеспечения затрагивает все его уровни от простых механизмов передачи сообщений до работы с такими сложными объектами как мониторы транзакций и вся среда прикладной системы. В частности, программное обеспечение должно минимизировать трафик межпроцессорного обмена, который может препятствовать линейному росту производительности системы.  
  
Аппаратные средства (процессоры, шины и устройства ввода/вывода) являются только частью масштабируемой архитектуры, на которой программное обеспечение может обеспечить предсказуемый рост производительности. Важно понимать, что простой переход, например, на более мощный процессор может привести к перегрузке других компонентов системы. Это означает, что действительно масштабируемая система должна быть *сбалансирована по всем параметрам*.   
^

**1.2.4 Совместимость и мобильность программного обеспечения**

Концепция программной *совместимости* впервые в широких масштабах была применена разработчиками системы IBM/360. Основная задача при проектировании совместимых систем заключалась в создании такой архитектуры, которая была бы одинаковой с точки зрения пользователя для всех моделей системы независимо от цены и производительности каждой из них.  
  
Совместимость позволяет сохранять существующий задел программного обеспечения при переходе на новые (как правило, более производительные) модели. Следует заметить однако, что со временем даже самая передовая архитектура неизбежно устаревает и возникает потребность внесения радикальных изменений архитектуру и способы организации вычислительных систем.   
  
В настоящее время одним из наиболее важных факторов, определяющих современные тенденции в развитии информационных технологий, является ориентация компаний-поставщиков компьютерного оборудования на рынок прикладных программных средств. Это объясняется прежде всего тем, что для конечного пользователя в конце концов важно программное обеспечение, позволяющее решить его задачи, а не выбор той или иной аппаратной платформы.   
  
Этот переход выдвинул ряд новых требований. Прежде всего такая вычислительная среда должна позволять гибко менять количество и состав аппаратных средств и программного обеспечения в соответствии с меняющимися требованиями решаемых задач. Во-вторых, она должна обеспечивать возможность запуска одних и тех же программных систем на различных аппаратных платформах, т.е. обеспечивать мобильность программного обеспечения. В третьих, эта среда должна гарантировать возможность применения одних и тех же человеко-машинных интерфейсов на всех компьютерах, входящих в неоднородную сеть.  
  
В условиях жесткой конкуренции производителей аппаратных платформ и программного обеспечения сформировалась *концепция открытых систем*, представляющая собой совокупность стандартов на различные компоненты вычислительной среды, предназначенных для обеспечения мобильности программных средств в рамках неоднородной, распределенной вычислительной системы.   
  
Одним из вариантов моделей открытой среды является модель OSE (Open System Environment), предложенная комитетом IEEE POSIX. На основе этой модели национальный институт стандартов и технологии США выпустил документ "Application Portability Profile (APP). The U.S. Government's Open System Environment Profile OSE/1 Version 2.0", который определяет рекомендуемые для федеральных учреждений США спецификации в области информационных технологий, обеспечивающие мобильность системного и прикладного программного обеспечения. Все ведущие производители компьютеров и программного обеспечения в США в настоящее время придерживаются требований этого документа.

## 1.3 Классификация компьютеров по областям применения

### 1.3.1 Персональные компьютеры и рабочие станции

*Персональные компьютеры (ПК)* появились в результате эволюции миникомпьютеров при переходе элементной базы машин с малой и средней степенью интеграции на большие и *сверхбольшие интегральные схемы*. ПК, благодаря своей низкой стоимости, очень быстро завоевали хорошие позиции на компьютерном рынке и создали предпосылки для разработки новых программных средств, ориентированных на самого широкого *потребителя непрофессионала*. Это прежде всего - *дружественные пользовательские интерфейсы*, а также *проблемно-ориентированные среды и инструментальные средства для автоматизации разработки прикладных программ*.   
  
Миникомпьютеры стали прародителями и другого направления развития современных систем - 32-разрядных машин. Создание RISC-процессоров и микросхем памяти емкостью более 1 Мбит привело к окончательному оформлению настольных систем высокой производительности, которые сегодня известны как *рабочие станции*.  
  
Первоначальная ориентация рабочих станций на *профессиональных пользователей* привела к тому, что рабочие станции - это хорошо сбалансированные системы, в которых высокое быстродействие сочетается с большим объемом оперативной и внешней памяти, высокопроизводительными внутренними магистралями, высококачественной и быстродействующей графической подсистемой и разнообразными устройствами ввода/вывода. Это свойство выгодно отличает рабочие станции среднего и высокого класса от ПК и сегодня.  
  
Тем не менее быстрый рост производительности ПК на базе новейших микропроцессоров Intel в сочетании с резким снижением цен на эти изделия и развитием технологии локальных шин (VESA и PCI), позволяющей устранить многие "узкие места" в архитектуре ПК, делают современные персональные компьютеры весьма привлекательной альтернативой рабочим станциям.  
  
В свою очередь производители рабочих станций создали изделия так называемого "начального уровня", которые по стоимостным характеристикам близки к высокопроизводительным ПК, но все еще сохраняют лидерство по производительности и возможностям наращивания.  
  
Поэтому в настоящее время появилось понятие персональной *рабочей станции,*которое объединяет оба направления.   
  
Среди других факторов, способствующих этому процессу, следует выделить: 

* Применение ПК стало более разнообразным. Помимо обычных для этого класса систем текстовых процессоров, даже средний пользователь ПК может теперь работать сразу с несколькими прикладными пакетами, включая электронные таблицы, базы данных и высококачественную графику.
* Адаптация графических пользовательских интерфейсов существенно увеличила требования пользователей ПК к соотношению производительность/стоимость.
* Широкое распространение систем мультимедиа прямо зависит от возможности использования высокопроизводительных ПК и рабочих станций с адекватными аудио- и графическими средствами, и объемами оперативной и внешней памяти.

### 1.3.2 X-терминалы

*X-терминалы* представляют собой комбинацию бездисковых рабочих станций и стандартных ASCII-терминалов.  
  
Бездисковые рабочие станции часто применялись в качестве дорогих дисплеев и в этом случае не полностью использовали локальную вычислительную мощь. Одновременно многие пользователи ASCII-терминалов хотели улучшить их характеристики, чтобы получить возможность работы в многооконной системе и графические возможности. Совсем недавно, как только стали доступными очень мощные графические рабочие станции, появилась тенденция применения "подчиненных" X-терминалов, которые используют рабочую станцию в качестве локального сервера.   
  
На компьютерном рынке X-терминалы занимают промежуточное положение между персональными компьютерами и рабочими станциями. Как правило, стоимость X-терминалов составляет около половины стоимости сравнимой по конфигурации бездисковой машины и примерно четверть стоимости полностью оснащенной рабочей станции.  
  
Типовой X-терминал включает следующие элементы: 

* Экран высокого разрешения - обычно размером от 14 до 21 дюйма по диагонали;
* Микропроцессор на базе Motorola 68xxx или RISC-процессор типа Intel i960, MIPS R3000 или AMD29000;
* Отдельный графический сопроцессор в дополнение к основному процессору, поддерживающий двухпроцессорную архитектуру, которая обеспечивает более быстрое рисование на экране и прокручивание экрана;
* Базовые системные программы, на которых работает система X-Windows и выполняются сетевые протоколы;
* Программное обеспечение сервера X11;
* Переменный объем локальной памяти (от 2 до 8 Мбайт) для дисплея, сетевого интерфейса, поддерживающего TCP/IP и другие сетевые протоколы.
* Порты для подключения клавиатуры и мыши.

X-терминалы отличаются от ПК и рабочих станций не только тем, что не выполняет функции обычной локальной обработки. Работа X-терминалов зависит от главной (*хост*) системы, к которой они подключены посредством сети. Для того, чтобы X-терминал мог работать, пользователи должны инсталлировать программное обеспечение многооконного сервера X11 на главном процессоре, выполняющим прикладную задачу (наиболее известная версия X11 Release 5).  
  
Х-терминалы отличаются также от стандартных алфавитно-цифровых ASCII и традиционных графических дисплейных терминалов тем, что они могут быть подключены к любой главной системе, которая поддерживает стандарт *X-Windows*. Более того, локальная вычислительная мощь X-терминала обычно используется для *обработки отображения*, а не обработки приложений, которые выполняются удаленно на главном компьютере (сервере). Вывод такого удаленного приложения просто отображается на экране X-терминала.   
  
Оснащенный стандартной системой X-Windows, X-терминал может отображать на одном и том же экране множество приложений одновременно. Каждое приложение может выполняться в своем окне и пользователь может изменять размеры окон, их месторасположение и манипулировать ими в любом месте экрана.   
  
X-Windows - результат совместной работы Масачусетского технологического института (MIT) и корпорации DEC. Система X-Windows (известная также под именем X) в настоящее время является открытым де-факто стандартом для доступа к множеству одновременно выполняющихся приложений с возможностями многооконного режима и графикой высокого разрешения на интеллектуальных терминалах, персональных компьютерах, рабочих станциях и X-терминалах.  
  
Она стала стандартом для обеспечения интероперабельности (переносимости) продуктов многих поставщиков и для организации доступа к множеству приложений.  
  
В настоящее время X-Windows является также стандартом для разработки пользовательского интерфейса. Более 90% поставщиков UNIX-рабочих станций и многие поставщики персональных компьютеров адаптировали систему X-Windows и применяют в качестве стандарта.

### 1.3.3 Серверы

В распределенной *модели клиент-сервер* часть работы выполняет компьютер – *сервер*, а часть пользовательский компьютер – *клиент*.  
  
Существует несколько типов серверов, ориентированных на разные применения:

* файл-сервер;
* сервер базы данных;
* принт-сервер;
* вычислительный сервер;
* сервер приложений.

Таким образом, тип сервера определяется *видом ресурса*, которым он владеет (файловая система, база данных, принтеры, процессоры или прикладные пакеты программ).   
  
С другой стороны существует классификация серверов, определяющаяся масштабом сети, в которой они используются:

* сервер рабочей группы;
* сервер отдела;
* сервер масштаба предприятия (корпоративный сервер).

Эта классификация весьма условна. Например, размер группы может меняться в диапазоне от нескольких человек до нескольких сотен человек, а сервер отдела обслуживать от 20 до 150 пользователей. Очевидно в зависимости от числа пользователей и характера решаемых ими задач требования к составу оборудования и программного обеспечения сервера, к его надежности и производительности сильно варьируются.   
  
Файловые серверы небольших рабочих групп (не более 20-30 человек) проще всего реализуются на платформе персональных компьютеров и программном обеспечении Novell NetWare. Файл-сервер, в данном случае, выполняет роль центрального хранилища данных.  
  
Серверы прикладных систем и высокопроизводительные машины для среды клиент-сервер значительно отличаются требованиями к аппаратным и программным средствам. Графика для большинства серверов несущественна, поэтому достаточно иметь обычный монохромный монитор с разрешением VGA.   
  
Скорость процессора для серверов с интенсивным вводом/выводом некритична. Они должны быть оснащены достаточно мощными блоками питания для возможности установки дополнительных плат расширения и дисковых накопителей. Желательно применение устройства бесперебойного питания. Как правило, для работы с многозадачными операционными системами такие серверы оснащаются интерфейсом SCSI (или Fast SCSI). Распределение данных по нескольким жестким дискам может значительно повысить производительность.   
  
При наличии одного сегмента сети и 10-20 рабочих станций пиковая пропускная способность сервера ограничивается максимальной пропускной способностью сети. Поэтому важно использовать хорошую плату сетевого интерфейса.   
  
Однако для файл-серверов общего доступа, с которыми одновременно могут работать несколько десятков, а то и сотен человек, простой однопроцессорной платформы и программного обеспечения Novell может оказаться недостаточно. В этом случае используются мощные многопроцессорные серверы с возможностями наращивания оперативной памяти до нескольких гигабайт, дискового пространства до сотен гигабайт, быстрыми интерфейсами дискового обмена (типа Fast SCSI-2, Fast&Wide SCSI-2 и Fiber Channel) и несколькими сетевыми интерфейсами. Эти серверы используют операционную систему UNIX, сетевые протоколы TCP/IP и NFS. Подобного рода серверы получили название *суперсерверов*.   
  
По уровню общесистемной производительности, функциональным возможностям отдельных компонентов, отказоустойчивости, а также в поддержке многопроцессорной обработки, системного администрирования и дисковых массивов большой емкости суперсерверы вышли в настоящее время на один уровень с мейнфреймами и мощными миникомпьютерами. Современные суперсерверы характеризуются: 

* наличием двух или более центральных процессоров RISC, либо Pentium, либо Intel 486;
* многоуровневой шинной архитектурой, в которой запатентованная высокоскоростная системная шина связывает между собой несколько процессоров и оперативную память, а также множество стандартных шин ввода/вывода, размещенных в том же корпусе;
* поддержкой технологии дисковых массивов RAID;
* поддержкой режима симметричной многопроцессорной обработки, которая позволяет распределять задания по нескольким центральным процессорам или режима асимметричной многопроцессорной обработки, которая допускает выделение процессоров для выполнения конкретных задач.

Как правило, суперсерверы работают под управлением операционных систем UNIX, а в последнее время и Windows NT, которые обеспечивают *многопотоковую многопроцессорную и многозадачную обработку*. Суперсерверы должны иметь достаточные возможности наращивания дискового пространства и вычислительной мощности, средства обеспечения надежности хранения данных и защиты от несанкционированного доступа. Кроме того, в условиях быстро растущей организации, важным условием является возможность наращивания и расширения уже существующей системы.  
^

### 1.3.4 Мейнфреймы (Main Frame)

*Мейнфрейм* - это синоним понятия *большая универсальная ЭВМ*. Мейнфреймы и до сегодняшнего дня остаются наиболее мощными (не считая суперкомпьютеров) *вычислительными системами общего назначения*, обеспечивающими непрерывный круглосуточный режим эксплуатации. Они могут включать один или несколько процессоров, каждый из которых, в свою очередь, может оснащаться *векторными сопроцессорами* (ускорителями операций с суперкомпьютерной производительностью).  
  
Прогресс в области элементно-конструкторской базы позволил существенно сократить габариты основных устройств. Наряду со сверхмощными мейнфреймами, требующими организации двухконтурной водяной системы охлаждения, имеются менее мощные модели, для охлаждения которых достаточно принудительной воздушной вентиляции, и модели, построенные по блочно-модульному принципу и не требующие специальных помещений и кондиционеров.   
  
Основными поставщиками мейнфреймов являются известные компьютерные компании IBM, Amdahl, ICL, Siemens Nixdorf и некоторые другие, но ведущая роль принадлежит безусловно компании IBM. Именно архитектура системы IBM/360, выпущенной в 1964 году, и ее последующие поколения стали образцом для подражания. В нашей стране в течение многих лет выпускались машины ряда ЕС ЭВМ, являвшиеся отечественным аналогом этой системы.   
  
В архитектурном плане мейнфреймы представляют собой многопроцессорные системы, содержащие один или несколько центральных и периферийных процессоров с общей памятью, связанных между собой высокоскоростными магистралями передачи данных. При этом основная вычислительная нагрузка ложится на центральные процессоры, а периферийные процессоры (в терминологии IBM - селекторные, блок-мультиплексные, мультиплексные каналы и процессоры телеобработки) обеспечивают работу с широкой номенклатурой периферийных устройств.   
  
Первоначально мейнфреймы ориентировались на централизованную модель вычислений, работали под управлением патентованных операционных систем и имели ограниченные возможности для объединения в единую систему оборудования различных фирм-поставщиков. Однако повышенный интерес потребителей к открытым системам заставил поставщиков мейнфреймов существенно расширить возможности своих операционных систем в направлении совместимости. В настоящее время они демонстрирует свою открытость, обеспечивая соответствие со спецификациями POSIX 1003.3, возможность использования протоколов межсоединений OSI и TCP/IP или предоставляя возможность работы на своих компьютерах под управлением операционной системы UNIX собственной разработки.   
  
Главным недостатком мейнфреймов в настоящее время остается относительно низкое соотношение производительность/стоимость. Однако фирмами-поставщиками мейнфреймов предпринимаются значительные усилия по улучшению этого показателя.   
^

### 1.3.5 Системы высокой готовности и кластерные архитектуры

Двумя основными проблемами построения вычислительных систем для критически важных приложений, связанных с обработкой транзакций, управлением базами данных и обслуживанием телекоммуникаций, являются обеспечение *высокой производительности и продолжительного функционирования систем*. Наиболее эффективный способ достижения заданного уровня производительности - применение *параллельных масштабируемых архитектур*.  
  
Задача обеспечения продолжительного функционирования системы имеет три составляющих:

* надежность;
* готовность;
* удобство обслуживания.

Все эти три составляющих предполагают, в первую очередь, борьбу с неисправностями системы, порождаемыми отказами и сбоями в ее работе. Эта борьба ведется по всем трем направлениям, которые взаимосвязаны и применяются совместно.   
  
Повышение уровня готовности предполагает подавление в определенных пределах влияния отказов и сбоев на работу системы с помощью средств контроля и коррекции ошибок, а также средств *автоматического восстановления вычислительного процесса* после проявления неисправности, включая аппаратурную и программную избыточность, на основе которой реализуются различные варианты отказоустойчивых архитектур. Повышение готовности есть способ борьбы за снижение времени простоя системы.  
  
Основные эксплуатационные характеристики системы существенно зависят от удобства ее обслуживания, в частности от ремонтопригодности, контролепригодности и т.д.   
  
В последние годы в литературе по вычислительной технике все чаще употребляется термин «*системы высокой готовности»(High Availability Systems)*. Все типы систем высокой готовности имеют общую цель - *минимизацию времени простоя*.  
  
Имеется два типа времени простоя компьютера: *плановое и неплановое*. Минимизация каждого из них требует различной стратегии и технологии. Плановое время простоя обычно включает время, принятое руководством, для проведения работ по модернизации системы и для ее обслуживания. Неплановое время простоя является результатом отказа системы или компонента. Хотя системы высокой готовности возможно больше ассоциируются с минимизацией неплановых простоев, они оказываются также полезными для уменьшения планового времени простоя.   
  
Существует несколько типов систем высокой готовности, отличающиеся своими функциональными возможностями и стоимостью. Стоимость систем высокой готовности на много превышает стоимость обычных систем. Вероятно поэтому наибольшее распространение в мире получили *кластерные системы*, благодаря тому, что они обеспечивают достаточно высокий уровень готовности систем при *относительно низких затратах*.  
  
Термин *кластеризация* на сегодня в компьютерной промышленности имеет много различных значений. Строгое определение могло бы звучать так: реализация объединения машин, представляющегося единым целым для операционной системы, системного программного обеспечения, прикладных программ и пользователей.  
  
Машины, кластеризованные вместе таким способом могут при отказе одного процессора очень быстро перераспределить работу на другие процессоры внутри кластера. Это, возможно, наиболее важная задача многих поставщиков систем высокой готовности.   
  
Первой концепцию кластерной системы анонсировала компания DEC, определив ее как группу объединенных между собой вычислительных машин, представляющих собой единый узел обработки информации. По существу VAX-кластер представляет собой слабосвязанную многомашинную систему с общей внешней памятью, обеспечивающую единый механизм управления и администрирования. В настоящее время на смену VAX-кластерам приходят UNIX-кластеры. При этом VAX-кластеры предлагают проверенный набор решений, который устанавливает критерии для оценки подобных систем.   
  
VAX-кластер обладает следующими свойствами: 

* *^ Разделение ресурсов.* Компьютеры VAX в кластере могут разделять доступ к общим ленточным и дисковым накопителям. Все компьютеры VAX в кластере могут обращаться к отдельным файлам данных как к локальным.
* *^ Высокая готовность.* Если происходит отказ одного из VAX-компьютеров, задания его пользователей автоматически могут быть перенесены на другой компьютер кластера. Если в системе имеется несколько контроллеров внешних накопителей и один из них отказывает, другие контроллеры автоматически подхватывают его работу.
* *^ Высокая пропускная способность.* Ряд прикладных систем могут пользоваться возможностью параллельного выполнения заданий на нескольких компьютерах кластера.
* *^ Удобство обслуживания системы.* Общие базы данных могут обслуживаться с единственного места. Прикладные программы могут инсталлироваться только однажды на общих дисках кластера и разделяться между всеми компьютерами кластера.
* *Расширяемость.* Увеличение вычислительной мощности кластера достигается подключением к нему дополнительных VAX-компьютеров. Дополнительные накопители на магнитных дисках и магнитных лентах становятся доступными для всех компьютеров, входящих в кластер.

Работа любой кластерной системы определяется двумя главными компонентами: высокоскоростным механизмом связи процессоров между собой и системным программным обеспечением, которое обеспечивает клиентам прозрачный доступ к системному сервису.

## 1.4 Оценка производительности вычислительных систем

Основу для сравнения различных типов компьютеров между собой дают стандартные методики измерения производительности. В процессе развития вычислительной техники появилось несколько таких стандартных методик. Они позволяют разработчикам и пользователям осуществлять выбор между альтернативами на основе количественных показателей, что дает возможность постоянного прогресса в данной области.   
  
Единицей измерения производительности компьютера является время: компьютер, выполняющий тот же объем работы за меньшее время является более быстрым. Время выполнения любой программы измеряется в секундах. Часто производительность измеряется как скорость появления некоторого числа событий в секунду, так что меньшее время подразумевает большую производительность. Однако в зависимости от того, что мы считаем, время может быть определено различными способами.  
  
Наиболее простой способ определения времени называется астрономическим временем, временем ответа (response time), временем выполнения (execution time) или прошедшим временем (elapsed time). Это задержка выполнения задания, включающая буквально все: работу процессора, обращения к диску, обращения к памяти, ввод/вывод и накладные расходы операционной системы. Однако при работе в мультипрограммном режиме во время ожидания ввода/вывода для одной программы, процессор может выполнять другую программу, и система не обязательно будет минимизировать время выполнения данной конкретной программы.   
  
Для измерения времени работы процессора на данной программе используется специальный параметр - время ЦП (CPU time), которое не включает время ожидания ввода/вывода или время выполнения другой программы. Очевидно, что время ответа, видимое пользователем, является полным временем выполнения программы, а не временем ЦП. Время ЦП может далее делиться на время, потраченное ЦП непосредственно на выполнение программы пользователя и называемое пользовательским временем ЦП, и время ЦП, затраченное операционной системой на выполнение заданий, затребованных программой, и называемое системным временем ЦП.   
  
В большинстве современных процессоров скорость протекания процессов взаимодействия внутренних функциональных устройств определяется не естественными задержками в этих устройствах, а задается единой системой синхросигналов, вырабатываемых некоторым генератором тактовых импульсов, как правило, работающим с постоянной скоростью. Дискретные временные события называются тактами синхронизации (clock ticks), просто тактами (ticks), периодами синхронизации (clock periods), циклами (cycles) или циклами синхронизации (clock cycles). Разработчики компьютеров обычно говорят о периоде синхронизации, который определяется либо своей длительностью (например, 10 наносекунд), либо частотой (например, 100 МГц). Длительность периода синхронизации есть величина, обратная к частоте синхронизации.   
  
Таким образом, время ЦП для некоторой программы может быть выражено двумя способами: количеством тактов синхронизации для данной программы, умноженным на длительность такта синхронизации, либо количеством тактов синхронизации для данной программы, деленным на частоту синхронизации.  
  
Важной характеристикой, часто публикуемой в отчетах по процессорам, является среднее количество тактов синхронизации на одну команду - CPI (clock cycles per instruction). При известном количестве выполняемых команд в программе этот параметр позволяет быстро оценить время ЦП для данной программы.   
  
Таким образом, производительность ЦП зависит от трех параметров: такта (или частоты) синхронизации, среднего количества тактов на команду и количества выполняемых команд. Невозможно изменить ни один из указанных параметров изолированно от другого, поскольку базовые технологии, используемые для изменения каждого из этих параметров, взаимосвязаны: частота синхронизации определяется технологией аппаратных средств и функциональной организацией процессора; среднее количество тактов на команду зависит от функциональной организации и архитектуры системы команд; а количество выполняемых в программе команд определяется архитектурой системы команд и технологией компиляторов. Когда сравниваются две машины, необходимо рассматривать все три компоненты, чтобы понять относительную производительность.  
  
В процессе поиска стандартной единицы измерения производительности компьютеров было принято несколько популярных единиц измерения, вследствие чего несколько безвредных терминов были искусственно вырваны из их хорошо определенного контекста и использованы там, для чего они никогда не предназначались. В действительности единственной подходящей и надежной единицей измерения производительности является время выполнения реальных программ, и все предлагаемые замены этого времени в качестве единицы измерения или замены реальных программ в качестве объектов измерения на синтетические программы только вводят в заблуждение. 

### 1.4.1. MIPS

Одной из альтернативных единиц измерения производительности процессора (по отношению к времени выполнения) является MIPS - (миллион команд в секунду). Имеется несколько различных вариантов интерпретации определения MIPS.   
  
В общем случае MIPS есть скорость операций в единицу времени, т.е. для любой данной программы MIPS есть просто отношение количества команд в программе к времени ее выполнения. Таким образом, производительность может быть определена как обратная к времени выполнения величина, причем более быстрые машины при этом будут иметь более высокий рейтинг MIPS.   
  
Положительными сторонами MIPS является то, что эту характеристику легко понять, особенно покупателю, и что более быстрая машина характеризуется большим числом MIPS, что соответствует нашим интуитивным представлениям.  
  
Однако использование MIPS в качестве метрики для сравнения наталкивается на три проблемы. Во-первых, MIPS зависит от набора команд процессора, что затрудняет сравнение по MIPS компьютеров, имеющих разные системы команд. Во-вторых, MIPS даже на одном и том же компьютере меняется от программы к программе. В-третьих, MIPS может меняться по отношению к производительности в противоположенную сторону.   
  
Классическим примером для последнего случая является рейтинг MIPS для машины, в состав которой входит сопроцессор плавающей точки. Поскольку в общем случае на каждую команду с плавающей точкой требуется большее количество тактов синхронизации, чем на целочисленную команду, то программы, используя сопроцессор плавающей точки вместо соответствующих подпрограмм из состава программного обеспечения, выполняются за меньшее время, но имеют меньший рейтинг MIPS. При отсутствии сопроцессора операции над числами с плавающей точкой реализуются с помощью подпрограмм, использующих более простые команды целочисленной арифметики и, как следствие, такие машины имеют более высокий рейтинг MIPS, но выполняют настолько большее количество команд, что общее время выполнения значительно увеличивается.   
  
Другое определение MIPS связано с очень популярным когда-то компьютером VAX 11/780 компании DEC. Именно этот компьютер был принят в качестве эталона для сравнения производительности различных машин. Считалось, что производительность VAX 11/780 равна 1MIPS (одному миллиону команд в секунду).   
  
Третье определение MIPS связано с IBM RS/6000 MIPS. Дело в том, что ряд производителей и пользователей (последователей фирмы IBM) предпочитают сравнивать производительность своих компьютеров с производительностью современных компьютеров IBM, а не со старой машиной компании DEC. Соотношение между VAX MIPS и RS/6000 MIPS никогда широко не публиковались, но 1 RS/6000 MIPS примерно равен 1.6 VAX 11/780 MIPS. 

### 1.4.2 MFLOP

Измерение производительности компьютеров при решении научно-технических задач, в которых существенно используется арифметика с плавающей точкой, всегда вызывало особый интерес. Именно для таких вычислений впервые встал вопрос об измерении производительности, а по достигнутым показателям часто делались выводы об общем уровне разработок компьютеров. Обычно для научно-технических задач производительность процессора оценивается в MFLOPS (миллионах чисел-результатов вычислений с плавающей точкой в секунду, или миллионах элементарных арифметических операций над числами с плавающей точкой, выполненных в секунду).   
  
Как единица измерения, MFLOPS, предназначена для оценки производительности только операций с плавающей точкой, и поэтому не применима вне этой ограниченной области. Например, программы компиляторов имеют рейтинг MFLOPS близкий к нулю вне зависимости от того, насколько быстра машина, поскольку компиляторы редко используют арифметику с плавающей точкой.   
  
Ясно, что рейтинг MFLOPS зависит от машины и от программы. Этот термин менее безобидный, чем MIPS. Он базируется на количестве выполняемых операций, а не на количестве выполняемых команд. По мнению многих программистов, одна и та же программа, работающая на различных компьютерах, будет выполнять различное количество команд, но одно и то же количество операций с плавающей точкой. Именно поэтому рейтинг MFLOPS предназначался для справедливого сравнения различных машин между собой.   
  
Однако и с MFLOPS не все обстоит так безоблачно. Прежде всего, это связано с тем, что наборы операций с плавающей точкой не совместимы на различных компьютерах. Например, в суперкомпьютерах фирмы Cray Research отсутствует команда деления (имеется, правда, операция вычисления обратной величины числа с плавающей точкой, а операция деления может быть реализована с помощью умножения делимого на обратную величину делителя). В то же время многие современные микропроцессоры имеют команды деления, вычисления квадратного корня, синуса и косинуса.   
  
Другая, осознаваемая всеми, проблема заключается в том, что рейтинг MFLOPS меняется не только на смеси целочисленных операций и операций с плавающей точкой, но и на смеси быстрых и медленных операций с плавающей точкой. Например, программа со 100% операций сложения будет иметь более высокий рейтинг, чем программа со 100% операций деления.   
  
Решение обеих проблем заключается в том, чтобы взять каноническое или нормализованное число операций с плавающей точкой из исходного текста программы и затем поделить его на время выполнения.

### 1.4.3 LINPACK.

LINPACK - это пакет фортран-программ для решения систем линейных алгебраических уравнений. Целью создания LINPACK отнюдь не было измерение производительности. Алгоритмы линейной алгебры весьма широко используются в самых разных задачах, и поэтому измерение производительности на LINPACK представляют интерес для многих пользователей. Сведения о производительности различных машин на пакете LINPACK публикуются Аргоннской национальной лабораторией (США) и периодически обновляются.   
  
В основе алгоритмов действующего варианта LINPACK лежит метод декомпозиции. Исходная матрица размером 100х100 элементов (в последнем варианте размером 1000х1000) сначала представляется в виде произведения двух матриц стандартной структуры, над которыми затем выполняется собственно алгоритм нахождения решения. Подпрограммы, входящие в LINPACK, структурированы. В стандартном варианте LINPACK выделен внутренний уровень базовых подпрограмм, каждая из которых выполняет элементарную операцию над векторами. Набор базовых подпрограмм называется BLAS (Basic Linear Algebra Subprograms). Например, в BLAS входят две простые подпрограммы SAXPY (умножение вектора на скаляр и сложение векторов) и SDOT (скалярное произведение векторов). Все операции выполняются над числами с плавающей точкой, представленными с двойной точностью. Результат измеряется в MFLOPS.   
  
Использование результатов работы тестового пакета LINPACK с двойной точностью как основы для демонстрации рейтинга MFLOPS стало общепринятой практикой в компьютерной промышленности. При этом следует помнить, что при использовании исходной матрицы размером 100х100, она полностью может размещаться в кэш-памяти емкостью, например, 1 Мбайт. Если при проведении испытаний используется матрица размером 1000х1000, то емкости такого кэша уже недостаточно и некоторые обращения к памяти будут ускоряться благодаря наличию такого кэша, другие же будут приводить к промахам и потребуют большего времени на обработку обращений к памяти. Для многопроцессорных систем также имеются параллельные версии LINPACK и такие системы часто показывают линейное увеличение производительности с ростом числа процессоров.   
^

### 1.4.4 SPECint92, SPECfp92

Важность создания пакетов тестов, базирующихся на реальных прикладных программах широкого круга пользователей и обеспечивающих эффективную оценку производительности процессоров, была осознана большинством крупнейших производителей компьютерного оборудования, которые в 1988 году учредили бесприбыльную корпорацию SPEC (Standard Performance Evaluation Corporation). Основной целью этой организации является разработка и поддержка стандартизованного набора специально подобранных тестовых программ для оценки производительности новейших поколений высокопроизводительных компьютеров.   
  
В настоящее время имеется два базовых набора тестов SPEC, ориентированных на интенсивные расчеты и измеряющих производительность процессора, системы памяти, а также эффективность генерации кода компилятором. Как правило, эти тесты ориентированы на операционную систему UNIX, но они также импортированы и на другие платформы. Процент времени, расходуемого на работу операционной системы и функции ввода/вывода, в общем случае ничтожно мал.   
  
Набор тестов CINT92, измеряющий производительность процессора при обработке целых чисел, состоит из шести программ, написанных на языке Си и выбранных из различных прикладных областей: теория цепей, интерпретатор языка Лисп, разработка логических схем, упаковка текстовых файлов, электронные таблицы и компиляция программ.   
  
Набор тестов CFP92, измеряющий производительность процессора при обработке чисел с плавающей точкой, состоит из 14 программ, также выбранных из различных прикладных областей: разработка аналоговых схем, моделирование методом Монте-Карло, квантовая химия, оптика, робототехника, квантовая физика, астрофизика, прогноз погоды и другие научные и инженерные задачи. Две программы из этого набора написаны на языке Си, а остальные 12 - на Фортране. В пяти программах используется одинарная, а в остальных - двойная точность.   
  
Результаты прогона каждого индивидуального теста из этих двух наборов выражаются отношением времени выполнения одной копии теста на тестируемой машине к времени ее выполнения на эталонной машине. В качестве эталонной машины используется VAX 11/780. SPEC публикует результаты прогона каждого отдельного теста, а также две составные оценки: SPECint92 - среднее геометрическое 6 результатов индивидуальных тестов из набора CINT92 и SPECfp92 - среднее геометрическое 14 результатов индивидуальных тестов из набора CFP92.   
^

### 1.4.5 SP*ECrate\_int92, SPECrate\_fp92*

Составные оценки SPECint92 и SPECfp92 достаточно хорошо характеризуют производительность процессора и системы памяти при работе в однозадачном режиме, но они совершенно не подходят для оценки производительности многопроцессорных и однопроцессорных систем, работающих в многозадачном режиме. Для этого нужна оценка пропускной способности системы или ее емкости, показывающая количество заданий, которое система может выполнить в течение заданного интервала времени. Пропускная способность системы определяется прежде всего количеством ресурсов (числом процессоров, емкостью оперативной и кэш-памяти, пропускной способностью шины), которые система может предоставить в распоряжение пользователя в каждый момент времени. Именно такую оценку, названную SPECrate и заменившую ранее применявшуюся оценку SPECthruput89, SPEC предложила в качестве единицы измерения производительности многопроцессорных систем.   
  
При этом для измерения выбран метод "однородной нагрузки" (homogenous capacity metod), заключающийся в том, что одновременно выполняются несколько копий одной и той же тестовой программы. Результаты этих тестов показывают, как много задач конкретного типа могут быть выполнены в указанное время, а их средние геометрические значения (SPECrate\_int92 - на наборе тестов, измеряющих производительность целочисленных операций и SPECrate\_fp92 - на наборе тестов, измеряющих производительность на операциях с плавающей точкой) наглядно отражают пропускную способность однопроцессорных и многопроцессорных конфигураций при работе в многозадачном режиме в системах коллективного пользования. В качестве тестовых программ для проведения испытаний на пропускную способность выбраны те же наборы CINT92 и CFT92.   
^

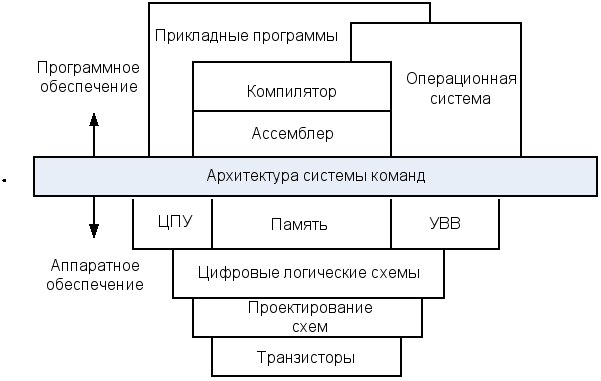
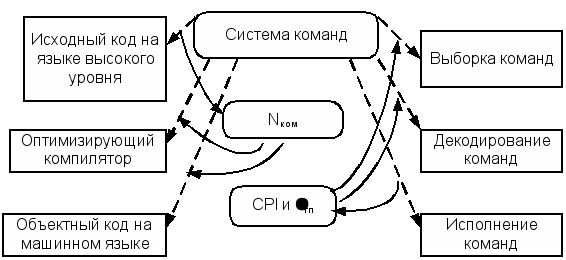
### 1.4.6 TPC-A, TPC-B, TPC-C

По мере расширения использования компьютеров при обработке транзакций в сфере бизнеса все более важной становится возможность справедливого сравнения систем между собой. С этой целью в 1988 году был создан Совет по оценке производительности обработки транзакций (TPC - Transaction Processing Performance Council). К настоящему времени TPC создал три тестовых пакета для обеспечения объективного сравнения различных систем обработки транзакций и планирует создать новые оценочные тесты.   
  
Выпущенный в ноябре 1989 года, тест TCP-A предназначался для оценки производительности систем, работающих в среде интенсивно обновляемых баз данных, типичной для приложений интерактивной обработки данных (OLDP - on-line data processing). Практически при выполнении теста эмулируется типичная вычислительная среда банка, включающая сервер базы данных, терминалы и линии связи. Этот тест использует одиночные, простые транзакции, интенсивно обновляющие базу данных. Одиночная транзакция (подобная обычной операции обновления счета клиента) обеспечивает простую, повторяемую единицу работы, которая проверяет ключевые компоненты системы OLTP. Тест TPC-A определяет пропускную способность системы, измеряемую количеством транзакций в секунду (tps A), которые система может выполнить при работе с множеством терминалов.   
  
В августе 1990 года TPC одобрил TPC-B, интенсивный тест базы данных. TPC-B измеряет пропускную способность системы в транзакциях в секунду (tpsB). Поскольку имеются существенные различия между двумя тестами TPC-A и TPC-B (в частности, в TPC-B не выполняется эмуляция терминалов и линий связи), их нельзя прямо сравнивать.  
  
Тестовый пакет TPC-C моделирует прикладную задачу обработки заказов. Он моделирует достаточно сложную систему OLTP, которая должна управлять приемом заказов, управлением учетом товаров и распространением товаров и услуг. Тест TPC-C осуществляет тестирование всех основных компонентов системы: терминалов, линий связи, ЦП, дискового в/в и базы данных. База данных TPC-C основана на модели оптового поставщика с удаленными районами и товарными складами. База данных содержит девять таблиц: товарные склады, район, покупатель, заказ, порядок заказов, новый заказ, статья счета, складские запасы и история.   
  
Обычно публикуются два результата. Один из них, tpm-C, представляет пиковую скорость выполнения транзакций (выражается в количестве транзакций в минуту). Второй результат, $/tpm-C, представляет собой нормализованную стоимость системы. Стоимость системы включает все аппаратные средства и программное обеспечение, используемые в тесте, плюс стоимость обслуживания в течение пяти лет. 

### 1.4.7 AIM

Одной из независимых организаций, осуществляющей оценку производительности вычислительных систем, является частная компания AIM Technology, которая была основана в 1981 году.   
  
За время своего существования компания разработала специальное программное обеспечение, позволяющее легко создавать различные рабочие нагрузки, соответствующие уровню тестируемой системы и требованиям по ее использованию. Это программное обеспечение состоит из двух основных частей: генератора тестовых пакетов (Benchmark Generator) и нагрузочных смесей (Load Mixes) прикладных задач.   
  
Генератор тестовых пакетов представляет собой программную систему, которая обеспечивает одновременное выполнение множества программ. Он содержит большое число отдельных тестов, которые потребляют определенные ресурсы системы, и тем самым акцентируют внимание на определенных компонентах, из которых складывается ее общая производительность. При каждом запуске генератора могут выполняться любые отдельные или все доступные тесты в любом порядке и при любом количестве проходов, позволяя тем самым создавать для системы практически любую необходимую рабочую нагрузку. Все это дает возможность тестовому пакету моделировать любой тип смеси при постоянной смене акцентов (для лучшего представления реальной окружающей обстановки) и при обеспечении высокой степени конфигурирования.   
  
Каждая нагрузочная смесь представляют собой формулу, которая определяет компоненты требуемой нагрузки. Эта формула задается в терминах количества различных доступных тестов, которые должны выполняться одновременно для моделирования рабочей нагрузки.   
  
Используя эти две части программного обеспечения AIM, можно действительно создать для тестируемой системы любую рабочую нагрузку, определяя компоненты нагрузки в терминах тестов, которые должны выполняться генератором тестовых пакетов. Если некоторые требуемые тесты отсутствуют в составе генератора тестовых пакетов, то они могут быть легко туда добавлены.   
  
К настоящему времени AIM создала восемь стандартных смесей, которые представляют собой обычную среду прикладных задач. В состав стандартных смесей входят следующие модули.  
  
Универсальная смесь для рабочих станций (General Workstation Mix) - моделирует работу рабочей станции в среде разработки программного обеспечения.   
  
Смесь для механического САПР (Mechanical CAD Mix) моделирует рабочую станцию, используемую для трехмерного моделирования и среды системы автоматизации проектирования в механике.   
  
Смесь для геоинформационных систем (GIS Mix) - моделирует рабочую станцию, используемую для обработки изображений и в приложениях геоинформацинных систем.   
  
Смесь универсальных деловых приложений (General Business) - моделирует рабочую станцию, используемую для выполнения таких стандартных инструментальных средств, как электронная почта, электронные таблицы, база данных, текстовый процессор и т.д.   
  
Многопользовательская смесь (Shared/Multiuser Mix) моделирует многопользовательскую систему, обеспечивающую обслуживание приложений для множества работающих в ней пользователей.   
  
Смесь для вычислительного (счетного) сервера (ComputeServer Mix) - моделирует систему, используемую для выполнения заданий с большим объемом вычислений, таких как маршрутизация PCB, гидростатическое моделирование, вычислительная химия, взламывание кодов и т.д.   
  
Смесь для файл-сервера (File Server Mix) - моделирует запросы, поступающие в систему, используемую в качестве централизованного файлового сервера, включая ввод/вывод и вычислительные мощности для других услуг по запросу.   
  
Смесь СУБД (RBMS Mix) - моделирует систему, выполняющую ответственные приложения управления базой данных.   
  
Рейтинг производительности по AIM - стандартная единица измерения пиковой производительности, установленная AIM Technology. Этот рейтинг определяет наивысший уровень производительности системы, который достигается при оптимальном использовании ЦП, операций с плавающей точкой и кэширования диска. Рейтинг машины VAX 11/780 обычно составляет 1 AIM.

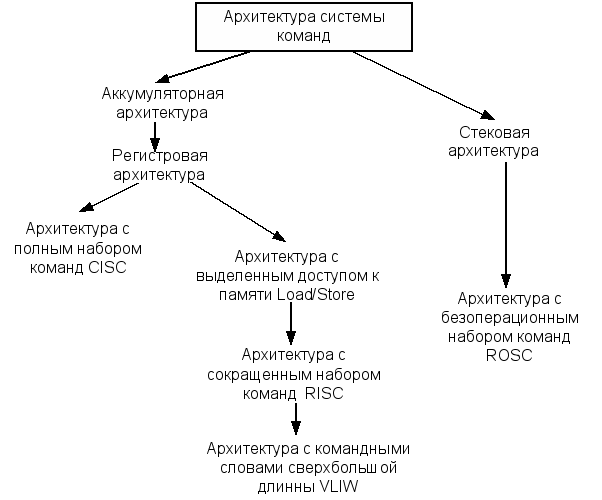
**2 Архитектура системы команд**

*Системой команд*вычислительной машины называют полный перечень команд, которые способна выполнять данная ЭВМ. В свою очередь, под *архитектурой системы команд*(АСК) принято определять те средства вычислительной машины, которые видны и доступны программисту.  
  
Компьютер представляет собой неделимую совокупность материального и идеального: аппаратных и программных компонентов. АСК можно рассматривать как линию согласования нужд разработчиков программного обеспечения с возможностями создателей аппаратуры вычислительной машины. Таким образом, АСК служит интерфейсом между программной (идеальной) и аппаратной (материальной) частями компьютера (рисунок 2.1).  
  
Рисунок 2.1 – Основные компоненты компьютера  
В конечном итоге, цель тех и других — реализация вычислений наиболее эффективным образом, то есть за минимальное время, и здесь важнейшую роль играет правильный выбор архитектуры системы команд.  
  
В упрощенной трактовке время выполнения программы Tвыч можно определить через число команд в программе Nком, среднее количество тактов процессора, приходящихся на одну команду CPI и длительность тактового периода τтп:  
Tвыч = Nком\*CPI\*τтп.  
Каждая из составляющих выражения зависит от одних аспектов архитектуры системы команд и, в свою очередь, влияет на другие (рисунок 2.2), что свидетельствует о необходимости чрезвычайно ответственного подхода к выбору АСК.  
  
Рисунок 2.2 – АСК и эффективность вычислений  
Общая характеристика архитектуры системы команд вычислительной машины складывается из ответов на следующие вопросы:  
  
1.Какого вида данные будут представлены в вычислительной машине и в какой форме?  
  
2.Где эти данные могут храниться помимо основной памяти?  
  
3.Каким образом будет осуществляться доступ к данным?  
  
4.Какие операции могут быть выполнены над данными?  
  
5.Сколько операндов может присутствовать в команде?  
  
6.Как будет определяться адрес очередной команды?  
  
7.Каким образом будут закодированы команды?  
  
В данном разделе обсудим наиболее распространенные архитектуры системы команд.  
^

**2.1 Классификация архитектур системы команд**

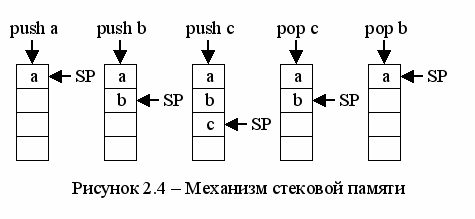
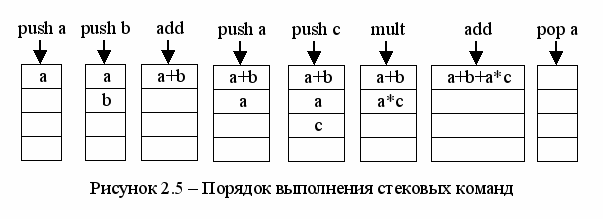
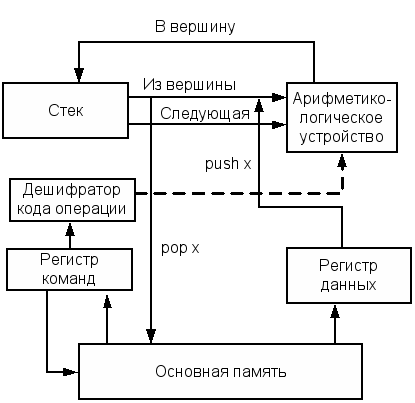
В истории развития вычислительной техники отражаются изменения, происходившие во взглядах разработчиков на перспективность той или иной архитектуры системы команд. Сложившуюся на настоящий момент ситуацию в области АСК иллюстрирует рисунком 2.3.  
  
Среди мотивов, чаще всего предопределяющих переход *к*новому типу АСК, остановимся на двух наиболее существенных. Первый — *место хранения операндов*, что влияет на количество и длину адресов, указываемых в адресной части команд обработки данных. Второй— это *состав операций*, выполняемых вычислительной машиной, и их сложность. Именно эти моменты взяты в качестве критериев излагаемых ниже вариантов классификации архитектур системы команд.  
  
Важную роль при выборе АСК играет ответ на вопрос о том, где могут храниться операнды и каким образом к ним осуществляется доступ. С этих позиции различают следующие виды архитектур системы команд:

* стековую;
* аккумуляторную;
* регистровую;
* с выделенным доступом к памяти.

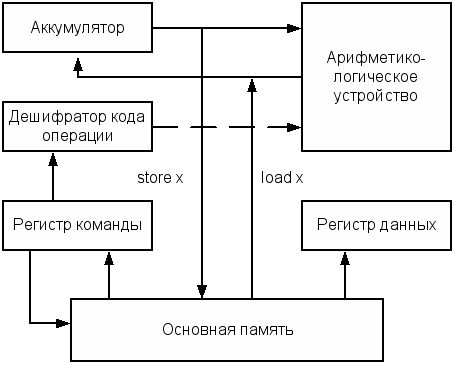
  
Рисунок 2.3 – Классификация АСК  
Выбор той или иной архитектуры влияет на принципиальные моменты: сколько адресов будет содержать адресная часть команд, какова будет длина этих адресов, насколько просто будет происходить доступ к операндам и какой, в конечном итоге, будет общая длина команд.  
^

**2.2 Классификация по месту хранения операндов**

**2.2.1 Стековая архитектура**

*Стеком*называется память, по своей структурной организации отличная от основной памяти ЭВМ, Принципы построения стековой памяти детально рассматриваются позже, здесь же выделим только те аспекты, которые требуются для пояснения особенностей АСК на базе стека.  
  
Стек образует множество логически взаимосвязанных ячеек (рисунок 2.4), взаимодействующих по принципу «последним вошел, первым вышел» (LIFO, Last In First Out).  
  
  
  
Верхнюю ячейку называют *вершиной стека (SP).*Для работы со стеком предусмотрены две операции: push (проталкивание данных в стек) и pop (выталкивание данных из стека). Запись возможна только в ячейку стека с адресом SP-1, при этом указатель стека смещается на одну позицию вниз. Чтение допустимо только из вершины стека (ячейка с адресом SP). Извлеченная информация удаляется из стека, а указатель продвигается вверх – принимает значение SP+1.  
  
В вычислительных машинах, где реализована АСК на базе стека (их обычно называют стековыми), операнды выбираются из двух верхних ячеек стековой памяти. Результат операции заносится в вершину стека. Принцип действия стековой машины поясним на примере вычисления выражения:  
*а = а + b + а\*с.*  
При описании вычислений с использованием стека обычно используется иная форма записи математических выражений, известная как обратная польская нотация, которую предложил польский математик Я. Лукашевич. Особенность ее в том, что в выражении отсутствуют скобки, а знак операции располагается не между операндами, а следует за ними (постфиксная форма). Последовательность операций определяется их приоритетами. Рассмотренное выше выражение в польской нотации примет вид:  
*а = ab+ ac\*+.*  
Данная форма записи однозначно определяет порядок загрузки операндов и выполнения операций в стековой архитектуре. Порядок выполнения команд в стековой АСК представлен на рисунке 2.5.  
  
  
  
Основные узлы и информационные тракты одного из возможных вариантов ЭВМ на основе стековой АСК показаны на рисунке 2.6.  
  
Информация может быть занесена в вершину стека из памяти или из АЛУ. Для записи в стек содержимого ячейки памяти с адресом *х*выполняется команда *push x*, по которой информация считывается из ячейки памяти, заносится в регистр данных, а затем проталкивается в стек. Результат операции из АЛУ заносится в вершину стека автоматически.  
  
Сохранение содержимого вершины стека в ячейке памяти с адресом *x*производится командой *pop x.*По этой команде содержимое верхней ячейки стека подается на шину, с которой и производится запись в ячейку *х,*после чего производится инкремент указателя стека: SP+1.  
  
Для выполнения арифметической или логической операции на вход АЛУ по дается информация, считанная из двух верхних ячеек стека (при этом содержимое стека продвигается на две позиции вверх, то есть операнды из стека удаляются). Результат операции заталкивается в вершину стека. Возможен вариант, когда результат сразу же переписывается в память с помощью автоматически выполняемой операции *pop х.*  
  
Верхние ячейки стековой памяти, где хранятся операнды и куда заносится результат операции, как правило, делаются более быстродействующими и размещаются в процессоре, в то время как остальная часть стека может располагаться в основной памяти и частично даже на магнитном диске.  
  
К достоинствам АСК на базе стека следует отнести возможность сокращения адресной части команд, поскольку все операции производятся через вершину стека, то есть адреса операндов и результата в командах арифметической и логической обработки информации указывать не нужно. Код программы получается компактным. Достаточно просто реализуется декодирование команд.  
  
  
Рисунок 2.6 – [Архитектура ЭВМ](https://gendocs.ru/v33228/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%AD%D0%92%D0%9C) на основе стековой АСК  
С другой стороны, стековая АСК по определению не предполагает произвольного доступа к памяти, из-за чего компилятору трудно создать эффективный программный код, хотя создание самих компиляторов упрощается. Кроме того, стек становится «узким местом» ЭВМ в плане повышения производительности.   
^

**2.2.2 Аккумуляторная архитектура**

Архитектура на базе аккумулятора исторически возникла одной из первых. В ней для хранения одного из операндов арифметической или логической операции в процессоре имеется выделенный регистр — *аккумулятор.*В этот же регистр заносится и результат операции. Поскольку адрес одного из операндов предопределен, в командах обработки достаточно явно указать местоположение только второго операнда.   
  
Типичная архитектура ЭВМ на базе аккумулятора показана на риунке 2.7.  
  
Для загрузки в аккумулятор содержимого ячейки *х*предусмотрена команда загрузки *load x.*По этой команде информация считывается из ячейки памяти *х,*выход памяти подключается к входам аккумулятора и происходит занесение считанных данных в аккумулятор.  
  
Запись содержимого аккумулятора в ячейку *х*осуществляется командой сохранения *store х,*при выполнении которой выходы аккумулятора подключаются к шине, после чего информация с шины записывается в память.  
  
Рисунок 2.7 – Архитектура ЭВМ на основе аккумуляторной АСК  
Для выполнения операции в АЛУ производится считывание одного из операндов из памяти в регистр данных. Второй операнд находится в аккумуляторе. Выходы регистра данных и аккумулятора подключаются к соответствующим входам АЛУ. По окончании предписанной операции результат с выхода АЛУ заносится в аккумулятор.  
  
Достоинствами аккумуляторной АСК можно считать короткие команды и простоту декодирования команд. Однако наличие всего одного регистра порождает многократные обращения к основной памяти.  
^

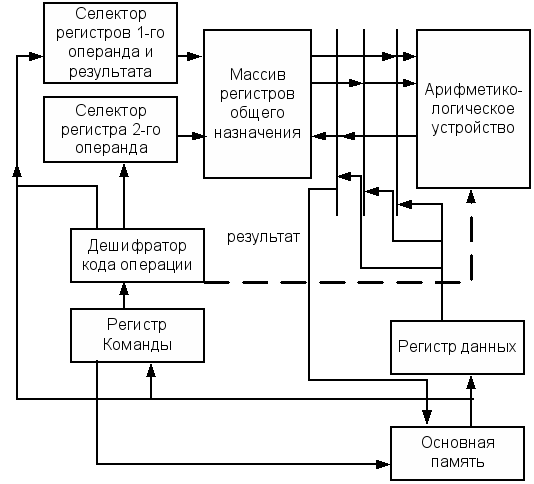
**2.2.3 Регистровая архитектура**

В машинах данного типа процессор включает в себя массив регистров (регистровый файл), известных как регистры общего назначения (РОН), Эти регистры, в каком-то смысле, можно рассматривать как явно управляемый кэш для хранения недавно использовавшихся данных.  
  
Размер регистров обычно фиксирован и совпадает с размером машинного слова, К любому регистру можно обратиться, указав его номер. Количество РОН в архитектурах типа CISC обычно невелико (от 8 до 32), и для представления номера конкретного регистра необходимо не более пяти разрядов, благодаря чему в адресной части команд обработки допустимо одновременно указать номера двух, а зачастую и трех регистров (двух регистров операндов и регистра результата). RISC-архитектура предполагает использование существенно большего числа РОН (до нескольких сотен), однако типичная для таких ЭВМ длина команды (обычно 32 разряда) позволяет определить в команде до трех регистров.  
Т а б л и ц а 2.1 – Сравнение вариантов размещения операндов

|  |  |  |
| --- | --- | --- |
| Тип команд | Достоинства | Недостатки |
| Регистр-регистр  (0, 3) | Простота реализации; фиксированная длина команды; простые алгоритмы компиляции; одинаковый CPI для всех команд. | Большой объем объектного кода; не всегда используются все поля команды. |
| Регистр-память  (1, 2) | Компактный объектный код; простота создания исходного кода. | Длинное поле адреса в команде; потеря одного из операндов. |
| Память-память  (3,3) | Компактный объектный код; малая потребность в РОН. | Низкое быстродействие; разнообразие форматов команд. |

Регистровая архитектура допускает расположение операндов в одной из двух запоминающих сред: основной памяти или регистрах. С учетом возможного размещения операндов в рамках регистровых АСК выделяют три подвида команд обработки:

* регистр-регистр;
* регистр-память;
* память-память.

В варианте «регистр-регистр» операнды могут находиться только в регистрах. В них же засылается и результат. Подтип «регистр-память» предполагает, что один из операндов размещается в регистре, а второй в основной памяти. Результат обычно замешает один из операндов. В командах типа «память-память» оба операнда хранятся в основной памяти. Результат заносится в память. Каждому из вариантов свойственны свои достоинства и недостатки (таблица 2.1).  
  
В выражениях вида (m, n) в первом столбце таблицы m означает количество операндов, хранящихся в основной памяти, a n – общее число операндов в команде арифметической или логической обработки.  
  
Вариант «регистр-регистр» является основным в вычислительных машинах типа RISC, Команды типа «регистр-память» характерны для CISC-машин. Наконец, вариант «память-память» считается неэффективным, хотя и остается в наиболее сложных моделях машин класса CISC.  
  
Возможную структуру и информационные тракты вычислительной машины с регистровой архитектурой системы команд иллюстрирует рисунок 2.8.  
  
Рисунок 2.8 – Архитектура ЭВМ на основе регистровой АСК  
Операции загрузки регистров из памяти и сохранения содержимого регистров в памяти идентичны таким же операциям с аккумулятором. Отличие состоит в этапе выбора нужного регистра, обеспечиваемого соответствующими селекторами.  
  
Выполнение операции в АЛУ включает в себя:  
  
выбор регистра первого операнда;  
  
определение расположения второго операнда (память или регистр);  
  
подачу на вход АЛУ операндов и выполнение операции;  
  
выбор регистра результата и занесение в него результата операции из АЛУ.  
  
Обратим внимание на то, что между АЛУ и регистровым файлом должны быть по крайней мере три шины (рисунок 2.8).  
  
К достоинствам регистровых АСК следует отнести: компактность получаемого кода, высокую скорость вычислений за счет замены обращений к основной памяти на обращения к быстрым регистрам. С другой стороны, данная архитектура требует более длинных инструкций по сравнению с аккумуляторной архитектурой.  
^

**2.2.4 Архитектура с выделенным доступом к памяти**

В архитектуре с выделенным доступом к памяти обращение к основной памяти возможно только с помощью двух специальных команд: *load*и *store.*В английской транскрипции данную архитектуру называют Load/Store architecture. Команда *load*(загрузка) обеспечивает считывание значения из основной памяти и занесение его в регистр процессора (в команде обычно указывается адрес ячейки памяти и номер регистра). Пересылка информации в противоположном направлении производится командой *store*(сохранение). Операнды во всех командах обработки информации могут находиться только в регистрах процессора (чаще всего в регистрах общего назначения). Результат операции также заносится в регистр. В архитектуре отсутствуют команды обработки, допускающие прямое обращение к основной памяти. Допускается наличие в АСК ограниченного числа команд, где операнд является частью кода команды.  
  
Состав и информационные тракты ЭВМ с выделенным доступом к памяти соответствуют рисунку 2.8. Две из трех шин, расположенных между массивом РОН и АЛУ, обеспечивают передачу в арифметико-логическое устройство операндов, хранящихся в двух регистрах общего назначения. Третья служит для занесения результата в выделенный для этого регистр. Эти же шины позволяют загрузить в регистры содержимое ячеек основной памяти и сохранить в ОП информацию, находящуюся в РОН.  
  
АСК с выделенным доступом к памяти характерна для всех вычислительных машин с RISC-архитектурой. Команды в таких ЭВМ, как правило, имеют длину 32 бита и трехадресный формат. К достоинствам АСК следует отнести простоту декодирования и исполнения команды.

**2.3 CISC и RISC архитектуры**

Другой подход к классификации системы команд ЭВМ основан на составе и степени сложности команд.  
  
Современная технология программирования ориентирована на языки высокого уровня (ЯВУ), главная цель которых — облегчить процесс программирования. Переход к ЯВУ, однако, породил серьезную проблему: сложные операторы, характерные для ЯВУ, существенно отличаются от простых машинных операций, реализуемых в большинстве вычислительных машин. Проблема получила название *семантического разрыва,*а ее следствием становится недостаточно эффективное выполнение программ на ЭВМ. Пытаясь преодолеть семантический разрыв, разработчики вычислительных машин в настоящее время выбирают один из трех подходов и, соответственно, один из трех типов АСК:  
  
архитектуру с полным набором команд: CISC (Complex Instruction Set Computer);  
  
архитектуру с сокращенным набором команд: RISC (Reduced Instruction Set Computer);  
  
архитектуру с командными словами сверхбольшой длины: VLIW (Very Long Instruction Word).  
  
В вычислительных машинах типа CISC проблема семантического разрыва решается за счет расширения системы команд, дополнения ее сложными командами, семантически аналогичными операторам ЯВУ. Основоположником CISC архитектуры считается компания IBM, которая начала применять данный подход с семейства машин IBM 360 и продолжает его в своих мощных современных универсальных ЭВМ. таких как IBM ES/9000. Аналогичный подход характерен и для компании Intel в ее микропроцессорах серии 8086 и Pentium. Для CISC-архитектуры типичны:  
  
наличие в процессоре сравнительно небольшого числа регистров общего назначения;  
  
большое количество машинных команд, некоторые из них аппаратно реализуют сложные операторы ЯВУ;  
  
разнообразие способов адресации операндов;  
  
множество форматов команд различной разрядности;  
  
наличие команд, где обработка совещается с обращением к памяти (регистр-память и память-память).  
  
К типу CISC можно отнести практически все ЭВМ, выпускавшиеся до середины 1980-х годов, и значительную часть производящихся в настоящее время. Рассмотренный способ решения проблемы семантического разрыва вместе с тем ведет к усложнению аппаратуры ЭВМ, главным образом устройства управления, что, в свою очередь, негативно сказывается на производительности ЭВМ в целом. Этозаставило более внимательно проанализировать программы, получаемые после компиляции с ЯВУ. Был предпринят комплекс исследований, в результате которых обнаружилось, что доля дополнительных команд, эквивалентных операторам ЯВУ, в общем объеме программ не превышает 10-20%, а для некоторых наиболее сложных команд даже 0,2%. В то же время объем аппаратных средств, требуемых для реализации дополнительных команд, возрастает весьма существенно. Так, емкость микропрограммной памяти при поддержании сложных команд может увеличиваться на 60%.  
  
Детальный анализ результатов упомянутых исследований привел к серьезному пересмотру традиционных решений, следствием чего стало появление *RISC-архитектуры.*Термин RISC впервые был использован Д. Паттерсоном и Д. Дитцелем в 1980 году. Идея заключается в ограничении списка команд ЭВМ наиболее часто используемыми простейшими командами, оперирующими данными, размещенными только в регистрах процессорах. Обращение к памяти допускается лишь с помощью специальных команд чтения и записи – выделенный доступ кпамяти. Резко уменьшено количество форматов команд и способов указания адресов операндов. Сокращение числа форматов команд и их простота, использование ограниченного количества способов адресации, отделение операций обработки данных от операций обращения к памяти позволяет существенно упростить аппаратные средства ЭВМ и повысить их быстродействие.  
  
RISC-архитектура разрабатывалась таким образом, чтобы уменьшить Tвыч за счет сокращения CPI и τтп*.* Как следствие, реализация сложных команд за счет последовательности из простых, но быстрых RISC-команд оказывается не менее эффективной, чем аппаратный вариант сложных команд в CISC-архитектуре.  
  
Элементы RISC-архитектуры впервые появились в вычислительных машинах CDC 6600 и суперЭВМ компании Cray Research. Достаточно успешно реализуется RISC-архитектура и в современных ЭВМ, например, в процессорах Alpha фирмы DEC, серии РА фирмы Hewlett-Packard, семействе PowerPC и т.п. Отметим, что в последних микропроцессорах фирмы Intel и AMD широко используются идеи, свойственные RISC-архитектуре, так что многие различия между CISC и RISC постепенно стираются.  
  
Помимо CISC- и RISC-архитектур в общей классификации был упомянут еще один тип АСК — архитектура с командными словами сверхбольшой длины (VL1W). Концепция VLIW базируется на RISC-архитектуре, где несколько простых RISC-команд объединяются в одну сверхдлинную команду и выполняются параллельно. В плане АСК архитектура VLIW сравнительно мало отличается от RISC. Появился лишь дополнительный уровень параллелизма вычислений, в силу чего архитектуру VLIW логичнее адресовать не к вычислительным машинам, а к вычислительным системам.  
  
Наиболее существенные различия в архитектурах типа CISC, RISC и VLIW приведены таблице 2.2.  
Т а б л и ц а 2.2 – Сравнительная оценка CISC-, RISC- и VLIW-архитектур

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | CISC | RISC | VLIW |
| Длина команды | Переменная | Постоянная | Постоянная |
| Расположение полей | Варьируется | Неизменно | Неизменно |
| Количество РОН | Несколько | Много | Много |
| Доступ к памяти | Выполняется как часть команды обработки | Выделенный | Выделенный |

^

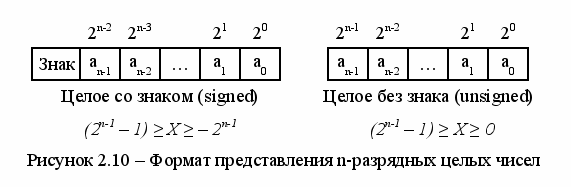
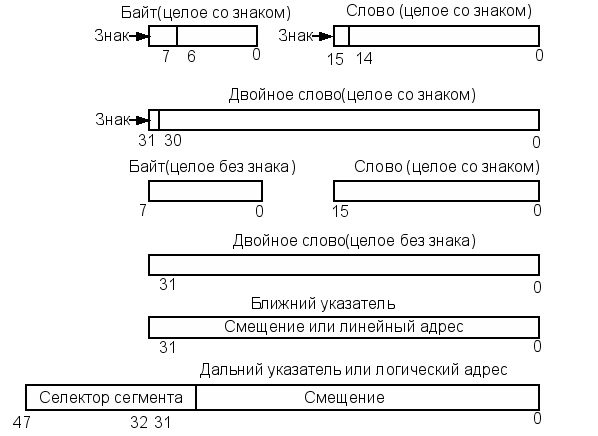
**2.4 Типы и форматы операндов**

Машинные команды оперируют данными, которые в этом случае принято называть *операндами.*К наиболее общим (базовым) типам операндов можно отнести: адреса, числа, символы и логические данные. Помимо них ЭВМ обеспечивает обработку и более сложных информационных единиц: графических изображений, аудио-, видео- и анимационной информации. Такая информация является производной от базовых типов данных и хранится в виде файлов на внешних запоминающих устройствах. Для каждого типа данных в ЭВМ предусмотрены определенные форматы.  
^

**2.4.1 Числовая информация**

Среди цифровых данных можно выделить две группы:  
  
целые типы, используемые для представления целых чисел;  
  
вещественные типы для представления рациональных чисел.  
  
В рамках первой группы имеется несколько форматов представления численной информации, зависящих от ее характера. Для представления вещественных чисел используется форма с плавающей запятой.  
^

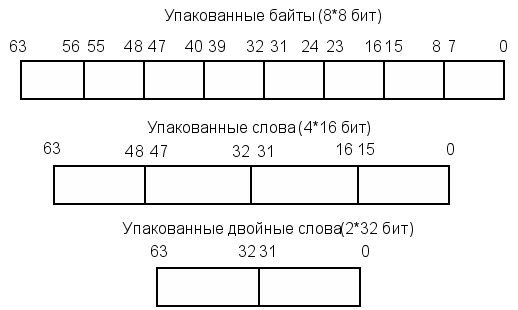
**2.4.2 Числа в форме с фиксированной запятой**

Представление числа *X в форме с фиксированной запятой (ФЗ)*, которую иногда называют также *естественной формой,*включает в себя знак числа и его модуль в q-ичном коде. Здесь *q — основание системы счисления*или *база.*Для современных ЭВМ характерна двоичная система (q=2), но иногда используются также восьмеричная (q*=* 8) или шестнадцатеричная (q=16) системы счисления. Запятую в записи числа называют соответственно двоичной, восьмеричной или шестнадцатеричной. Знак положительного числа кодируется двоичной цифрой 0, а знак отрицательного числа — цифрой 1.  
  
Числам с ФЭ соответствует запись вида  
  
*X = ±an-1an-2…a1a0,a-1a-2…a-r.*  
  
Отрицательные числа обычно представляются в дополнительном коде. Разряд кода числа, в котором размешается знак, называется *знаковым разрядом кода.*Разряды, где располагаются значащие цифры числа, называются *цифровыми разрядами кода.*Знаковый разряд размещается левее старшего цифрового разряда. Положение запятой одинаково для всех чисел и в процессе решения задач не меняется. Хотя запятая и фиксируется, в коде числа она никак не выделяется, а только подразумевается. В общем случае разрядная сетка ЭВМ для размещения чисел в форме с ФЗ имеет вид, представленный на рисунке 2.9, где n разрядов используются для записи целой части числа и rразрядов — для дробной части.   
  
  
  
При заданных значениях n и г *диапазон изменения модулей* чисел, коды которых могут быть представлены в данной разрядной сетке, определяется соотношением  
*qn > |X| ≥ q-r.*  
Если число является смешанным (содержит целую и дробную части), оно обрабатываются как целое, хотя и не является таковым (в этом случае применяют термин*масштабируемое целое*).Обработкасмешанных чисел в ЭВМ встречается крайне редко.  
  
  
  
  
Рисунок 2.11 – Целочисленные форматы в микропроцессорах фирмы Intel  
Как правило, используются ЭВМ с *целочисленной арифметикой*. Целые числа представимы при фиксации запятой после младшего разряда (r=0). Это наиболее распространенный способ, поэтому фактически понятие ФЗ относится исключительно к целым числам, и операции с числами в форме ФЗ будут характеризоваться как целочисленные. Возможные форматы целых чисел со знаком и без знака показаны на рисунке 2.10.  
  
Отрицательные целые числа в двоичном счислении (q=2) представляются в дополнительном формате, что позволяет избежать двойного представления нуля: ±0. *Дополнительный формат* получается следующим образом:

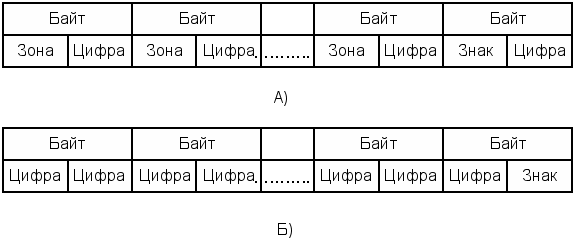
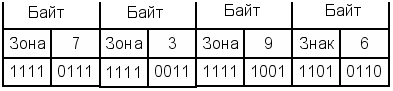
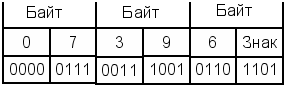
1. производится инверсия всех разрядов в представлении модуля исходного числа;
2. к полученному результату прибавляется единица.

Тогда число «–0» (все цифровые разряды нулевые) фактически будет представлять число «–2n-1».  
  
На рисунке 2.11 приведены целочисленные форматы с фиксированной запятой, принятые в микропроцессорах фирмы Intel.  
  
Целые числа применяются также для работы с адресами. На рисунке 2.11 это 32-разрядный формат ближнего и 48-разрядный формат дальнего указателей.  
  
Представление чисел в формате ФЗ упрощает аппаратурную реализацию ЭВМ и сокращает время выполнения машинных операций, однако при решении задач необходимо постоянно следить за тем, чтобы все исходные данные, промежуточные и окончательные результаты не выходили за допустимый диапазон формата, иначе возможно переполнение разрядной сетки и результат вычислений будет неверным.  
^

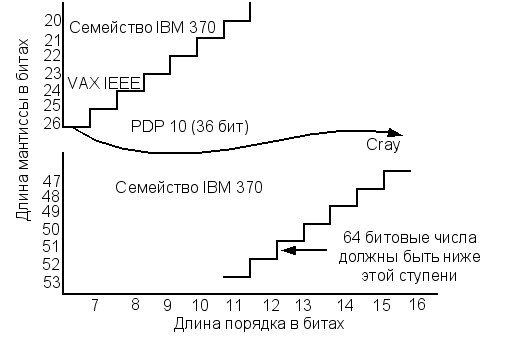
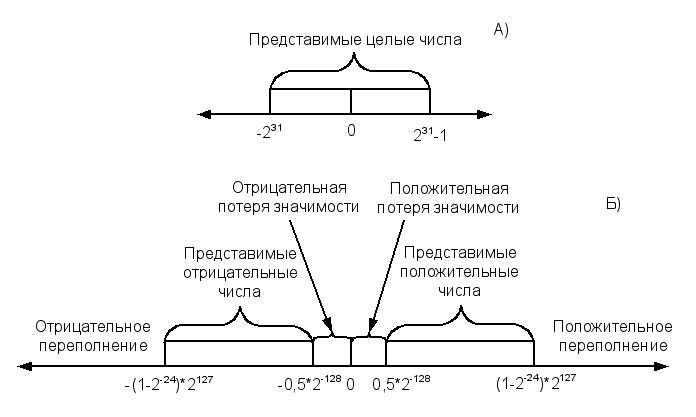
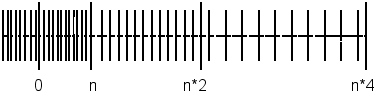
**2.4.3 Упакованные целые числа**

  
Рисунок 2.12 – Форматы упакованных чисел в технологиях MMX и 3D Now!  
В АСК современных микропроцессоров имеются команды, оперирующие целыми числами, представленными в упакованном виде. Связано это с обработкой мультимедийной информации. Формат предполагает упаковку в пределах достаточно длинного слова (обычно 64-разрядного) нескольких небольших целых чисел, а соответствующие команды обрабатывают все эти числа параллельно. Если каждое из чисел состоит из четырех двоичных разрядов, то в 64-разрядное слово можно поместить до 16 таких чисел. Неиспользованные разряды заполняются нулями, В микропроцессорах фирмы Intel, начиная с Pentium ММХ, присутствуют специальные команды для обработки мультимедийной информации (ММХ-команды), оперирующие целыми числами, упакованными в квадрослова (64-разрядные слова). Предусмотрены три формата (рисунок 2.12): упакованные байты (восемь 8-разрядных чисел); упакованные слова (четыре 16-разрядных числа) и упакованные двойные слова (два 32-разрядных числа).  
  
Байты в формате упакованных байтов нумеруются от 0 до 7, причем байт 0 располагается в младших разрядах квадрослова. Аналогичная система нумерации и размещения упакованных чисел применяется для упакованных слов (номера 0-3) и упакованных двойных слов (номера 0-1).  
  
Идентичные форматы упакованных данных применяются также в другой технологии обработки мультимедийной информации, предложенной фирмой AMD. Эта технология носит название 3DNow!, а реализована в микропроцессорах данной фирмы.  
^

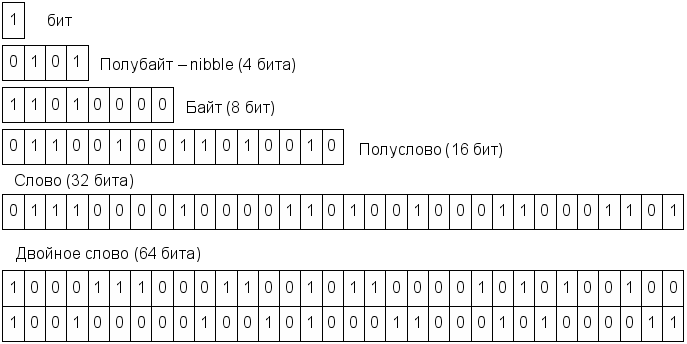
**2.4.4 Десятичные числа**

  
  
А) – зонный; Б) – уплотненный  
  
Рисунок 2.13 – Форматы десятичных чисел  
В ряде задач, главным образом, учетно-статистического характера, приходится иметь дело с хранением, обработкой и пересылкой десятичной информации. Особенность таких задач состоит в том, что обрабатываемые числа могут состоять из различного и весьма большого количества десятичных цифр. Традиционные методы обработки с переводом исходных данных в двоичную систему счисления и обратным преобразованием результата зачастую сопряжены с существенными накладными расходами. По этой причине в ЭВМ применяются иные специальные формы представления десятичных данных. В их основу положен принцип кодирования каждой десятичной цифры эквивалентным двоичным числом из четырех битов (тетрадой), то есть так называемым двоично-десятичным кодом (BCD – Binary Coded Decimal).  
  
Используются два формата представления десятичных чисел (все числа рассматриваются как целые): *зонный (распакованный) уплотненный (упакованный).*В обоих форматах каждая десятичная цифра представляется двоичной тетрадой, то есть заменяется двоично-десятичным кодом. Из оставшихся задействованных шести четырехразрядных двоичных комбинации (24= 16) две служат для кодирования знаков «плюс» и «минус». Например, в ЭВМ семейства IBM 360/370/390 для знака «плюс» выбран код 11002=С16, а для знака «минус» — код 11012=D16.  
  
Зонный формат, приведенный на рисунке 2.13, А), применяется в операциях ввода/вывода. В нем под каждую цифру выделяется один байт, где младшие четыре разряда отводятся под код цифры, а в старшую тетраду (поле зоны) записывается специальный код «зона», не совпадающий с кодами цифр и знаков. В IBM 360/370/390 это код 11112- F16 Исключение составляет байт, содержащий младшую цифру десятичного числа, где в поле зоны хранится знак числа. На рисунке 2.14 показана запись числа «–7396» в зонном формате. В некоторых ЭВМ принят вариант зонного формата, где поле зоны заполняется нулями.  
  
Рисунок 2.14 – Представление числа «–7396» в зонном формате  
При выполнении операций сложения и вычитания над десятичными числами обычно используется упакованный формат и в нем же получается результат. Умножение и деление возможно только в зонном формате.  
  
В упакованном формате, показанном на рисунке 2.13, Б), каждый байт содержит коды двух десятичных цифр. Правая тетрада последнего байта предназначается для записи знака числа. Десятичное число должно занимать целое количество байтов. Если это условие не выполняется, то четыре старших двоичных разряда левого байта заполняется нулями. Так, представление числа -7396 в упакованном формате имеет вид, приведенный на рисунке 2.15.  
  
Рисунок 2.15 – Представление числа «–7396» в упакованном формате  
Размещение знака в младшем байте, как в зонном, так и в упакованном представлениях, позволяет задавать десятичные числа произвольной длины и передавать их в виде цепочки байтов. В этом случае знак указывает, что байт, в котором он содержится, является последним байтом данного числа, а следующий байт последовательности — это старший байт очередного числа.  
^

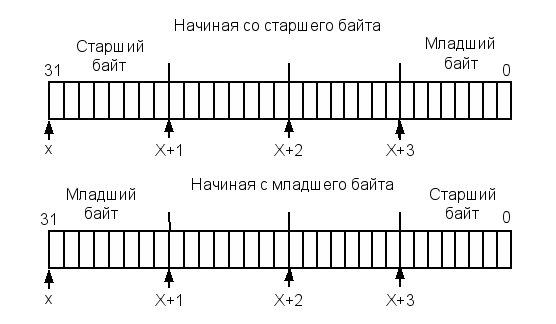
**2.4.5 Числа в форме с плавающей запятой**

От недостатков ФЗ в значительной степени свободна форма представления чисел с *плавающей запятой (ПЗ)*, известная также под названиями *нормальной*или *полулогарифмической*формы. В данном варианте каждое число разбивается на две группы цифр. Первая группа цифр называется *мантиссой,*вторая — *порядком.*Число представляется в виде произведения  
*X*= *± mq±p,*  
где *т –*мантисса числа *X, р*– порядок числа, *q*– основание системы счисления.  
  
  
  
Для представления числа в форме с ПЗ требуется задать знаки мантиссы и порядка, их модули в q-ичном коде, а также основание системы счисления q. Формат представления числа с ПЗ показан на рисунке 2.16. Нормальная форма неоднозначна, так как взаимное изменение m и p приводит к «плаванию» запятой, чем и обусловлено название этой формы.  
  
Диапазон и точность представления чисел с ПТ зависят от числа разрядов, отводимых под порядок и мантиссу. На рисунке 2.17 показаны диапазоны разрядностей порядка и мантиссы, характерные для известных ЭВМ.  
  
В большинстве вычислительных машин для упрощения операций над порядками последние приводят к целым положительным числам, применяя так называемый *смещенный порядок.*Для этого к истинному порядку добавляется целое положительное число – смещение. Например, в системе со смещением 128 порядок «–3» представляется как «125» (–3 + 128). Обычно смещение выбирается равным половине представимого диапазона порядков. Отметим, что смещенный порядок занимает все биты поля порядка, в том числе и тот, который ранее использовался для записи знака порядка.  
  
  
Рисунок 2.17 – Типовые разрядности порядка и мантиссы  
Мантисса в числах с ПЗ обычно представляется в нормализованной форме*.*Это означает, что на мантиссу налагаются такие условия, чтобы она по модулю была меньше единицы, а первая цифра после точки отличалась от нуля. Полученная таким образом мантисса называется *нормализованной.*  
  
Для представления числа с ПЗ в ЭВМ используется база 2 (q = 2). При этом применяют еще один способ повышения точности представления мантиссы, называемый *приемом скрытой единицы.*Суть его в том, что в нормализованной мантиссе старшая цифра всегда равна единице, следовательно, эта цифра может не записываться, а подразумеваться. Запись мантиссы начинают с ее второй цифры, и это позволяет задействовать дополнительный значащий бит для более точного представления числа. Следует отметить, что значение порядка в данном случае не меняется. Скрытая единица перед выполнением арифметических операций восстанавливается, а при записи результата — удаляется. Таким образом, нормализованная мантисса 0,101000(1) при использовании способа «скрытой единицы» будет иметь вид 0,010001 (в скобках указана цифра, не поместившаяся в поле мантиссы при стандартной записи).  
  
Для более существенного увеличения точности вычислений под число отводят несколько машинных слов, например два. Дополнительные биты, как правило, служат для увеличения разрядности мантиссы, однако в ряде случаев часть из них может отводиться и для расширения поля порядка. В процессе вычислений может получаться ненормализованное число. В таком случае ЭВМ, если это предписано командой, автоматически нормализует его.  
  
Примеры. На рисунке 2.18 представлен типичный 32-битовый формат числа с ПЗ. Старший (левый) бит содержит знак числа. Значение смешенного порядка хранится в разрядах с 1-го по 8-й и может находиться в диапазоне от 0 до 255. Для получения фактического значения порядка из содержимого этого поля нужно вычесть фиксированное значение, равное 128. С таким смещением фактические значения порядка могут лежать в диапазоне от -128 до +127. Третье поле слова содержит нормализованную мантиссу со скрытым разрядом (единицей). Благодаря такому приему 23-разрядное поле позволяет хранить 24-разрядную мантиссу в диапазоне от 0,5 до 1,0.  
  
Рисунок 2.18 – 32-битовый формат числа с ПЗ  
На рис. 2.19 приведены диапазоны чисел, которые могут быть записаны с помощью 32-разрядного слова.  
  
  
  
Рисунок 2.19 – Числа, представимые в 32-битовых форматах:  
  
А) – целые числа; Б) – числа с ПЗ  
В варианте для целых чисел в дополнительном коде могут быть представлены все целые числа от -231 до (231 – 1), то есть всего 232 различных чисел (см. рисунок 2.19, А)*.*  
  
Для случая ПЗ представимы два диапазона чисел (см. рисунок 2.19, Б):  
  
отрицательные числа: *–(1-2–24)\*2127 ≤ X ≤ –0,5\*2–128*;  
  
положительные числа: *0,5\*2–128 ≤ X ≤ (1-2–24)\*2127*.  
  
В эту область не включены пять участков:  
  
отрицательные числа, меньшие чем*–(1-2–24)\*2127*– *отрицательное переполнение;*  
  
отрицательные числа, большие чем *–0,5\*2–128* – *отрицательная потеря значимости;*  
  
положительные числа, меньшие чем *0,5\*2–128* – *положительная потеря значимости;*  
  
положительные числа, большие чем *(1-2–24)\*2127* – *положительное переполнение.*  
  
Показанная запись числа с ПЗ не учитывает нулевого значения. Для этой цели используется специальная кодовая комбинация. Переполнения возникают, когда в результате арифметической операции получается значение большее, чем можно представить порядком 127. При переполнении, как правило, выдается соответствующее сообщение и выполнение программы прекращается. Потеря значимости – это когда результат представляет собой слишком маленькое дробное значение. Потеря значимости является менее серьезной проблемой, поскольку такой результат обычно рассматривают как нулевой.  
  
Кроме этого, числа в форме с ПЗ, в отличие от чисел в форме с ФЗ, размешены на числовой оси *неравномерно*. Расстояние между соседними представимыми числами увеличивается с ростом порядка n. Возможные значения в начале числовой оси расположены плотнее, а по мере движения вправо – все реже, что иллюстрирует рисунок 2.20. Это означает, что чаще всего вычисления приводят к результату, который не является точным, а представляет собой округление до ближайшего представимого значения.  
  
  
Рисунок 2.20 – Плотность представимых чисел с ПЗ на числовой оси  
Для формата с ПЗ имеет место противоречие между диапазоном и точностью. Если увеличить число битов, отведенных под порядок, расширяется диапазон представимых чисел. Однако, поскольку может быть представлено только фиксированное число различных значений, уменьшается плотность и тем самым точность. Единственный путь увеличения как диапазона, так и точности – увеличение количества разрядов, поэтому в большинстве ЭВМ предлагается использовать числа в одинарном и двойном форматах. Например, число одинарного формата может занимать 32 бита, а двойного (double) – 64 бита.  
  
Числа с плавающей запятой в разных ЭВМ имеют несколько различных форматов. В настоящее время для всех ЭВМ рекомендован стандарт, разработанный общепризнанным международным центром стандартизации IEEE (Institute of Electrical and Electronics Engineers). Этот стандарт был разработан с целью облегчить перенос программ с одного процессора на другие и нашел широкое применение практически во всех процессорах и арифметических сопроцессорах.  
  
Стандарт определяет 32-битовый (float) и 64-битовый (double) форматы с 8- и 11-разрядным порядком, соответственно. Основанием системы счисления является 2.   
  
Не все кодовые комбинации в форматах ШЕЕ интерпретируются обычным путем – некоторые комбинации используются для представления специальных значений. Предельные значения порядка, содержащие все нули (0) и все единицы (255 — в одинарном формате и 2047 — в двойном формате), определяют специальные значения. Нулевой порядок совместно с нулевой мантиссой представляют положительный или отрицательный 0, в зависимости от состояния бита знака мантиссы. Порядок, содержащий единицы во всех разрядах, совокупно с нулевой мантиссой представляют положительную или отрицательную бесконечность, в зависимости от состояния бита знака, что позволяет пользователю самому решить, считать ли это ошибкой или продолжать вычисления со значением, равным бесконечности. Кодовая комбинация, в которой порядок содержит все единицы, а мантисса не равна 0, используется как признак «не числа» (NAN — Not a Number) и служит для предупреждения о различных исключительных ситуациях.  
^

**2.4.7 Единицы разрядности типов данных**

Операционные устройства вычислительных машин рассчитаны на обработку кодов фиксированной длины. Общепринятые величины разрядности кодов чисел показаны на рисунке 2.21.  
  
Наименьшей единицей данных в ЭВМ служит *бит (BIT, Binary digiT — двоичная цифра)*. В большинстве случаев эта единица информации слишком мала. Однобитовые операционные устройства использовались в ЭВМ с последовательной обработкой информации, а в современных машинах с параллельной обработкой разрядов они практически не применяются. Побитовую работу с данными скорее можно встретить в многопроцессорных вычислительных системах, построенных из одноразрядных процессоров.  
  
Следующая по величине единица состоит из четырех битов и называется *полубайтом или тетрадой*, или реже «ниблом» (nibble — огрызок). Она также редко имеет самостоятельное значение и заслуживает упоминания как единица представления отдельных десятичных цифр при их двоично-десятичной записи.  
  
Реально наименьшей обрабатываемой единицей считается *байт*, состоящий из восьми битов. На практике эта единица информации также оказывается недостаточной, и значительно чаще применяются числа, представленные двумя (*полуслово*), четырьмя (*слово*), восемью (*двойное слово*) или шестнадцатью (*счетверенное слово*) байтами.  
  
Разрядность целочисленного АЛУ обычно выбирается равной разрядности адреса (для большинства современных ЭВМ это 32 разряда). Следовательно, наиболее выгодными в плане быстродействия являются такие целые числа, длина которых совпадает с разрядностью адреса. Использование более коротких чисел позволяет сэкономить на памяти, но выигрыша в производительности не дает.  
  
Рисунок 2.21 – Стандартные единицы разрядности  
Блоки операций с плавающей запятой обычно согласованы со стандартом IEEE 754 и рассчитаны на обработку чисел в формате двойной длины (64 бита). В большинстве ЭВМ реальная разрядность таких блоков даже больше (80 бит). Таким образом, наилучшим вариантом при проведении вычислений с плавающей запятой можно считать формат двойного слова. При выборе формата меньшей длины (32 разряда) вычисления все равно ведутся с большей точностью, после чего результат округляется. Таким образом, использование короткого формата чисел с плавающей запятой, как и в случае целых чисел с фиксированной запятой, помимо экономии памяти никаких иных преимуществ также не дает.  
^

**2.4.8 Размещение числовых данных в памяти**

В современных ЭВМ разрядность одной ячейки памяти, как правило, равна одному байту (8 бит). В то же время реальная длина кодов чисел составляет 2, 4, 8 или 16 байт. При хранении таких чисел в памяти последовательные байты числа размещают в нескольких ячейках с последовательными адресами, при этом для доступа к числу указывается только *наименьший из адресов*. При разработке архитектуры системы команд необходимо определить порядок размещения байтов в памяти, то есть какому из байтов (старшему или младшему) будет соответствовать этот наименьший адрес. На рисунке 2.22 показаны оба варианта размещения 32-разрядного числа в четырех последовательных ячейках памяти, начиная с адреса *X.*  
  
  
Рисунок 2.22 – Варианты размещения 32-разрядного числа  
В вычислительном плане оба способа записи равноценны. Так, фирмы DEC и Intel отдают предпочтения размещению в первой ячейке младшего байта, a IBM и Motorola ориентируются на противоположный вариант. Выбор обычно связан с некими иными соображениями разработчиков ЭВМ. В настоящее время в большинстве машин предусматривается использование обоих вариантов, причем выбор может быть произведен программным путем за счет соответствующей установки регистра конфигурации.  
  
Помимо порядка размещения байтов, существенным бывает и выбор адреса, с которого может начинаться запись числа. Связано это с физической реализацией полупроводниковых запоминающих устройств, где обычно предусматривается возможность считывания (записи) четырех байтов подряд. Причем данная операция выполняется быстрее, если адрес первого байта *А*отвечает условию A mod S = 0,S= 2, 4, 8, 16. Числа, размешенные в памяти в соответствии с этим правилом, называются *выровненными*. Большинство компиляторов генерируют код, в котором предусмотрено выравнивание чисел в памяти,

**2.5 Символьная информация**

В общем объеме вычислительных действий все большая доля приходится на обработку символьной информации, содержащей буквы, цифры, знаки препинания, математические и другие символы. Каждому символу ставится в соответствие определенная двоичная комбинация. Совокупность возможных символов и назначенных им двоичных кодов образует *таблицу кодировки.*В настоящее время применяется множество различных таблиц кодировки. Объединяет их весовой принцип, при котором коды цифр возрастают по мере увеличения цифры, а коды символов увеличиваются в алфавитном порядке. Так код буквы «Б» на единицу больше кода буквы «А». Это способствует упрощению обработки в ЭВМ.  
  
До недавнего времени наиболее распространенными были кодовые таблицы, в которых символы кодируются с помощью восьмиразрядных двоичных комбинаций (байтов), позволяющих представить 256 различных символов:

* расширенный двоично-кодированный код EBCDIC (Extended Binary Coded Decimal Interchange Code);
* американский стандартный код для обмена информацией ASCII (American ' Standard Code for Information Interchange).
* Код EBCDIC используется в качестве внутреннего кода в универсальных ЭВМ фирмы IBM. Он же известен под названием ДКОИ (двоичный код для обработки информации).

Стандартный код ASCII – 7-разрядный, восьмая позиция отводится для записи бита четности. Это обеспечивает представление 128 символов, включая все латинские буквы, цифры, знаки основных математических операций и знаки пунктуации. Позже появилась европейская модификация ASCII, называемая Latin 1 (стандарт ISO 8859-1). В ней «полезно» используются все 8 разрядов. Дополнительные комбинации (коды 128-255) в новом варианте отводятся для представления специфических букв алфавитов западноевропейских языков, символов псевдографики, некоторых букв греческого алфавита, а также ряда математических и финансовых символов. Именно эта кодовая таблица считается мировым стандартом де-факто, который применяется с различными модификациями во всех странах.  
  
Хотя код ASCII достаточно удобен, он все же слишком тесен и не вмещает множества необходимых символов. По этой причине в 1993 году консорциумом компаний Apple Computer, Microsoft, Hewlett-Packard, DEC и IBM был разработан 16-битовый стандарт ISO 10646, определяющий универсальный набор символов (UCS, Universal Character Set). Новый код, известный под названием Unicode, позволяет задать до 65536 символов, то есть дает возможность одновременно представить символы всех основных «живых» и «мертвых» языков. Для букв русского языка выделены коды 1040-1093.  
  
В «естественном» варианте кодировки Unicode, известном как UCS-2, каждый символ описывается двумя последовательными байтами *т*и *n*, так что номеру символа соответствует численное значение *256****\*****т + п.*Таким образом, кодовый номер представлен 16-разрядным двоичным числом.  
  
Наряду с UCS-2 в рамках Unicode существуют еще несколько вариантов кодировки Unicode (UTF, Unicode Transformation Formats), основные из которых UTF-8 и UTF-7. В кодировке UTF-8 коды символов меньшие, чем 128, представляются одним байтом. Все остальные коды формируются по более сложным правилам. В зависимости от символа его код может занимать от двух до шести байтов, причем старший бит каждого байта всегда имеет единичное значение. Иными словами, значение байта лежит в диапазоне от 128 до 255. Ноль в старшем бите байта означает, что код занимает один байт и совпадает по кодировке с ASCII.  
  
В UTF-7 код символа также может занимать один или более байтов, однако в каждом из байтов значение не превышает 127 (старший бит байта содержит ноль). Многие символы кодируются одним байтом, и *их*кодировка совпадает с ASCII, однако некоторые коды зарезервированы для использования в качестве преамбулы, характеризующей последующие байты многобайтового кода.  
  
Стандарт Unicode обратно совместим с кодировкой ASCII, однако если в ASCII для представления схожих по виду символов (минус, тире, знак переноса) применялся общий код, в Unicode каждый из этих символов имеет уникальную кодировку. Впервые Unicode был использован в операционной системе Windows NT.   
  
Параллельно с развитием Unicode исследовательская группа ISO проводит работы над 32-битовой кодовой таблицей, однако ввиду широкой распространенности кодировки Unicode дальнейшие перспективы новой разработки представляются неопределенными.  
^

**2.6 Логические данные и строки**

Элементом *логических* данных является логическая (булева) переменная, которая может принимать лишь два значения: «истина» или «ложь». Кодирование логического значения принято осуществлять битом информации: единицей кодируют истинное значение, нулем — ложное. Как правило, в ЭВМ оперируют наборами логических переменных длиной в машинное слово. Обрабатываются такие слова с помощью команд логических операций (И, ИЛИ, НЕ и т. д.), при этом все биты обрабатываются одинаково, но независимо друг от друга, то есть никаких переносов между разрядами не возникает.  
  
*^ Строки* — это непрерывная последовательность битов, байтов, слов или двойных слов. *Битовая строка*может начинаться в любой позиции байта и содержать до 232 бит. *Байтовая строка*может состоять из байтов, слов или двойных слов. Длина такой строки варьируется от нуля до (232 – 1) байт (4 Гбайт). Приведенные цифры характерны для превалирующих в настоящее время 32-разрядных ЭВМ.  
  
Если байты байтовой строки представляют собой коды символов, то говорят о *текстовой строке.*Поскольку длина текстовой строки может меняться в очень широких пределах, то для указания конца строки в последний байт заносится код-ограничитель – обычно это нули во всех разрядах байта. Иногда вместо ограничителя длину строки указывают числом, расположенным в первом байте (или двух байтах) строки.  
^

**2.7 Прочие виды информации**

Представляемую в ЭВМ информацию можно разделить на *статическую и динамическую*. Так, числовая, символьная и логическая информация является статической – ее значение не связано со временем. Напротив, аудиоинформация имеет динамический характер – существует только в режиме реального времени и не может быть остановлена для более подробного изучения. Если изменить масштаб времени, аудиоинформация искажается, что используется, например, для создания звуковых эффектов.

**2.7.1 Видеоинформация**

Видеоинформация бывает как статической, так и динамической. *^ Статическая видеоинформация* включает в себя текст, рисунки, графики, чертежи, таблицы и др. Рисунки делятся также на плоские – двумерные и объемные – трехмерные.  
  
*^ Динамическая видеоинформация* – это видеофильмы и мультипликация, использующаяся для передачи движущихся изображений. В их основе лежит последовательное экспонирование на экране в реальном масштабе времени отдельных кадров в соответствии со сценарием.  
  
Демонстрация анимационных и слайд-фильмов опирается на различные принципы. Анимационные фильмы демонстрируются так, чтобы зрительный аппарат человека не мог зафиксировать отдельных кадров. Для получения качественной анимации кадры должны сменяться порядка 30 раз в секунду. При демонстрации слайд-фильмов каждый кадр экспонируется на экране столько времени, сколько необходимо для восприятия его человеком (обычно от 30 с до 1 мин). Слайд-фильмы можно отнести к статической видеоинформации.  
  
В вычислительной технике существует два способа представления графических изображений; *матричный (растровый)*и *векторный.*Матричные (bitmap) форматы хорошо подходят для изображений со сложными гаммами цветов, оттенков и форм, таких как фотографии, рисунки, отсканированные данные. Векторные форматы более приспособлены для чертежей и изображений с простыми формами, тенями и окраской.  
  
В матричных форматах изображение представляется прямоугольной матрицей точек — *пикселов*(picture element), положение которых в матрице соответствует координатам точек на экране. Помимо координат каждый пиксел характеризуется своим цветом, цветом фона или градацией яркости. Количество битов, выделяемых для указания цвета пиксела, изменяется в зависимости от формата. В высококачественных изображениях цвет пиксела описывают 24 битами, что дает около 16 миллионов цветов. Основной недостаток матричной (растровой) графики заключается в большой емкости памяти, требуемой для хранения изображения, из-за чего для описания изображений прибегают к различным методам сжатия данных. В настоящее время существует множество форматов графических файлов, различающихся алгоритмами сжатия и способами представления матричных изображений, а также сферой применения.  
  
Векторное представление, в отличие от матричной графики, определяет описание изображения не пикселями, а кривыми – сплайнами. *Сплайн* - это гладкая кривая, которая проходит через две или более опорные точки, управляющие формой сплайна. В векторной графике наиболее распространены сплайны на основе *кривых Безье*, Суть сплайна: любую элементарную кривую можно построить, зная четыре коэффициента Р0, Р1, Р2 и Р3, соответствующие четырем точкам на плоскости. Перемещение этих точек влечет за собой изменение формы кривой, как показано на рисунке 2.23.

В векторной графике для описания объектов используются математические формулы. Это позволяет при рисовании объектов вычислять, куда необходимо помещать реальные точки изображения. Имеется ряд простейших объектов, или *примитивов,*например эллипс, прямоугольник, линия. Эти примитивы и их комбинации служат основой для создания более сложных изображений. В простейшем случае изображение может быть составлено из отрезков линии, для которых задаются начальные координаты, угол наклона, длина, толщина линии, цвет линии и цвет фона.  
  
Основное достоинство векторной графики состоит в том, что описание объекта является простым и занимает мало памяти. Кроме того, векторная графика в сравнении с матричной имеет следующие преимущества:

* простота масштабирования изображения без ухудшения его качества;
* независимость емкости памяти, требуемой для храпения изображения, от выбранной цветовой модели.

Недостатком векторных изображений является их некоторая искусственность, заключающаяся в том, что любое изображение необходимо разбить на конечное множество составляющих его примитивов. Как и для матричной графики, существует несколько форматов графических векторных файлов.  
  
Матричная и векторная графика существуют не обособленно друг от друга. Так, векторные рисунки могут включать в себя и матричные изображения. Кроме того, векторные и матричные изображения могут быть преобразованы друг в друга. Графические форматы, позволяющие сочетать матричное и векторное описание изображения, называются *метафайлами.*Метафайлы обеспечивают достаточную компактность файлов с сохранением высокого качества изображения.  
  
Рассмотренные формы представления статической видеоинформации используются, в частности, для отдельных кадров, образующих анимационные фильмы. Для хранения анимационных фильмов применяются различные методы сжатия информации, большинство из которых стандартизовано.

**2.7.2 Аудиоинформация**

Понятие *аудио*связано со звуками, которые способно воспринимать человеческое ухо. Частоты аудиосигналов лежат в диапазоне от 15 Гц до 20 КГц, а сигналы по своей природе являются непрерывными (аналоговыми). Прежде чем быть представленной в ЭВМ, аудиоинформация должна быть преобразована в цифровую форму (оцифрована). Для этого значения звуковых сигналов (*выборки, samples*), взятые через малые промежутки времени, с помощью аналого-цифровых преобразователей (АЦП) переводятся в двоичный код. Обратное действие выполняется цифро-аналоговыми преобразователями (ЦАП). Чем чаще производятся выборки, тем выше может быть точность последующего воспроизведения исходного сигнала, но тем большая емкость памяти требуется для хранения оцифрованного звука.  
  
Цифровой эквивалент аудиосигналов обычно хранится в виде файлов, причем широко используются различные методы сжатия такой информации. Как правило, к методам сжатия аудиоинформации предъявляется требование возможности восстановления непрерывного сигнала без заметного ухудшения его качества. В настоящее время распространен целый ряд форматов хранения аудиоинформации.