# ***Архитектура***

## 

## **1. Классификация ЭВМ и ВС**

|  |  |  |
| --- | --- | --- |
| Тип | Цена, долларов | Сфера применения |
| «Одноразовые» компьютеры | 0,5 | Поздравительные открытки |
| Встроенные компьютеры (микроконтроллеры) | 5 | Часы, машины, различные приборы |
| Мобильные и игровые компьютеры | 50 | Домашние компьютерные игры, смартфоны |
| Персональные компьютеры | 500 | Настольные и портативные компьютеры |
| Серверы | 5000 | Сетевые серверы |
| Мэйнфреймы | 5 000 000 | Пакетная обработка данных в банке |

**ЭВМ нулевого поколения – механические компьютеры(1642-1945)**

Вычислительная техника развивалась со времен изобретения счет до нашей эры. Нулевое поколение можно назвать механической эрой вычислительной техники. Для этого поколения характерны механические вычислительные устройства - арифмометры. С развитием появляются устройства на основе электро-механических реле, набором ламп, вводом данных с помощью перфокарт и программы с помощью перфоленты. Возможность за короткое время(менее 10с, умножение 3c) выполнять основные арифметические операции(сложение, вычитание, умножение, деление) над числами до 23 разрядов (вычислитель Mark1). ZUSE

**ЭВМ первого поколения – электронные лампы(1945-1955)**

Онибыли ламповыми машинами 50-х годов. Их элементной базой были электровакуумные лампы. Эти ЭВМ были весьма громоздкими сооружениями, содержавшими в себе тысячи ламп, занимавшими иногда сотни квадратных метров территории, потреблявшими электроэнергию в сотни киловатт.

Например, одна из первых ЭВМ – [ENIAC](https://www.compgramotnost.ru/istoria-computera/poyavilas-pervaya-evm) представляла собой огромный по объему агрегат длиной более 30 метров, содержала 18 тысяч электровакуумных ламп и потребляла около 150 киловатт электроэнергии.

Для ввода программ и данных применялись перфоленты и перфокарты. Не было монитора, клавиатуры и мышки. Использовались эти машины, главным образом, для инженерных и научных расчетов, не связанных с переработкой больших объемов данных. В 1949 году в США был создан первый полупроводниковый прибор, заменяющий электронную лампу. Он получил название **транзистор**.

**ЭВМ второго поколения – транзисторы (1955-1965)**

В 60-х годах транзисторы стали элементной базой для ЭВМ второго поколения. Машины стали компактнее, надежнее, менее энергоемкими. Возросло быстродействие и объем внутренней памяти. Большое развитие получили устройства внешней (магнитной) памяти: магнитные барабаны, накопители на магнитных лентах.

В этот период стали развиваться языки программирования высокого уровня: ФОРТРАН, АЛГОЛ, КОБОЛ. Составление программы перестало зависеть от конкретной модели машины, сделалось проще, понятнее, доступнее.

В 1959 г. был изобретен метод, позволивший создавать на одной пластине и транзисторы, и все необходимые соединения между ними. Полученные таким образом схемы стали называться интегральными схемами или чипами. Изобретение интегральных схем послужило основой для дальнейшей миниатюризации компьютеров.

В дальнейшем количество транзисторов, которое удавалось разместить на единицу площади интегральной схемы, увеличивалось приблизительно вдвое каждый год.

**ЭВМ третьего поколения – интегральные схемы (1965 – 1980)**

Это поколение ЭВМ создавалось на новой элементной базе – **интегральных схемах (ИС)**.

ЭВМ третьего поколения начали производиться во второй половине 60-х годов, когда американская фирма IBM приступила к выпуску системы машин IBM-360. Немного позднее появились машины серии IBM-370.

В Советском Союзе в 70-х годах начался выпуск машин серии ЕС ЭВМ (Единая система ЭВМ) по образцу IBM 360/370. Скорость работы наиболее мощных моделей ЭВМ достигла уже нескольких миллионов операций в секунду. На машинах третьего поколения появился новый тип внешних запоминающих устройств – магнитные диски.

Успехи в развитии электроники привели к созданию больших интегральных схем (БИС), где в одном кристалле размещалось несколько десятков тысяч электрических элементов.

В 1971 году американская фирма Intel объявила о создании микропроцессора. Это событие стало революционным в электронике.

Соединив микропроцессор с устройствами ввода-вывода и внешней памяти,  получили новый тип компьютера: микро-ЭВМ.

**ЭВМ четвертого поколения – сверхбольшие интегральные схемы (1980 - ?)**

Микро-ЭВМ относится к машинам четвертого поколения. Наибольшее распространение получили персональные компьютеры (ПК). Их появление связано с именами двух американских специалистов: [Стива Джобса](https://www.compgramotnost.ru/it-specialistu/stiv-dzhobs-on-prosto-vzyal-i-izmenil-mir) и Стива Возняка. В 1976 году на свет появился их первый серийный ПК Apple-1, а в 1977 году – Apple-2.

Однако с 1980 года «законодателем мод» на рынке ПК становится американская фирма IBM. Ее архитектура стала фактически международным стандартом на профессиональные ПК. Машины этой серии получили название IBM PC (Personal Computer). Появление и распространение ПК по своему значению для общественного развития сопоставимо с появлением книгопечатания.

С развитием этого типа машин появилось понятие «информационные технологии», без которых невозможно обойтись в большинстве областей деятельности человека. Появилась новая дисциплина – информатика.

**ЭВМ пятого поколения – компьютеры небольшой мощности и невидимые компьютеры**

Они будут основаны на принципиально новой элементной базе. Основным их качеством должен быть высокий интеллектуальный уровень, в частности, распознавание речи, образов. Это требует перехода от традиционной фон-неймановской [архитектуры компьютера](https://www.compgramotnost.ru/sostav-computera/princip-otkrytoj-arxitektury) к архитектурам, учитывающим требования задач создания искусственного интеллекта.

Пятое поколение ЭВМ строится по принципу человеческого мозга, управляется голосом. Соответственно, предполагается применение принципиально новых технологий. Огромные усилия были предприняты Японией в разработке компьютера 5-го поколения с искусственным интеллектом, но успеха они пока не добились.

Фирма IBM тоже не намерена сдавать свои позиции мирового лидера, например, Японии. Мировая гонка за создание компьютера пятого поколения началась еще в 1981 году.

## **1.2. Основные параметры и общие принципы построения ЭВМ**

Основные параметры ЭВМ:

Производительность – характеризуется числом команд, выполняемых ЭВМ за одну секунду.

Емкость запоминающих устройств - определяет, какой набор программ и данных может быть одновременно размещен в памяти.

Надежность - это способность ЭВМ при определенных условиях выполнять требуемые функции в течение заданного периода времени.

Точность - Точность получения результатов обработки в основном определяется разрядностью ЭВМ, а также используемыми структурными еди­ницами представления информации (байтом, словом, двойным словом)

Также к основным параметрам относят потребляемую мошность (в ваттах) и габаритные размеры.

1945г. - Джон Фон Нейман сформулировал принципы построения компьютера:

1)Состав основных устройств ЭВМ: универсальная вычислительная машина должна включать в себя устройство памяти, арифметико-логическое устройство (АЛУ), устройство управления (УУ), устройство ввода/вывода.

2)Принцип двоичного кодирования: данные и программы хранятся в памяти в форме двоичного кода.

3)Принцип хранимой программы: во время обработки данные и программы находятся в общей памяти ЭВМ.

4)Принцип адресуемой памяти: память состоит из двоичных разрядов – битов, которые объединяются в ячейки, каждая из которых имеет адрес. Адрес ячейки памяти – это ее порядковый номер.

5)Принцип программного управления: работа машины происходит автоматически под управлением программы, которая помещается в оперативную память.

## **2. Построение классической ЭВМ. Особенности развития структур ЭВМ**

Основным принципом построения всех современных компьютеров является программное управление. В его основе лежит представление алгоритма решения любой задачи в виде программы вычислений.

Классическая схема компьютера, отвечающая программному принципу управления, логично вытекает из последовательного характера преобразований, выполняемых человеком по некоторому алгоритму (программе). В любом компьютере имеются устройства ввода информации (УВв), с помощью которых пользователи вводят программы решаемых задач и данные. Введенная информация сначала полностью или частично запоминается в оперативном запоминающем устройстве (ОЗУ), а затем переносится во внешнее запоминающее устройство (ВЗУ), предназначенное для длительного хранения информации, где преобразуется в специальный информационный объект - файл. При использовании файла в вычислительном процессе его содержимое переносится в ОЗУ. Затем программная информация команда за командой считывается в устройство управления.

Устройство управления (УУ) предназначается для автоматического выполнения программ путем принудительной координации всех остальных устройств. Вызываемые из ОЗУ команды дешифрируются устройством управления: определяют код операции, которую необходимо выполнить следующей, и адреса операндов, принимающих участие в данной операции. Арифметико-логическое устройство (АЛУ) выполняет арифметические и логические операции над данными. Основной частью АЛУ является операционный автомат, в состав которого входят сумматоры, счетчики, регистры, логические преобразователи и др. Он каждый раз перестраивается на выполнение очередной операции. Результаты выполнения отдельных операций сохраняются для последующего использования на одном из регистров АЛУ или записываются в память. В персональных компьютерах, относящихся к компьютерам четвертого поколения, произошло дальнейшее изменение структуры. Соединение всех устройств в единую машину обеспечивается с помощью общей шины, представляющей собой линии передачи данных, адресов, сигналов управления и питания. Ядро ПК образуют процессор, основная память (ОП), состоящая из оперативной памяти и постоянного запоминающего устройства (ПЗУ), и видеопамять. Подключение всех внешних устройств (ВнУ), дисплея, клавиатуры, внешних ЗУ и др. обеспечивается через соответствующие адаптеры - согласователи скоростей работы сопрягаемых устройств, или контроллеры - специальные устройства управления периферийной аппаратурой. Организацию согласованной работы шин и устройств выполняют микросхемы системной логики, называемые чипсетом. Уровень высокоскоростных устройств образуют процессоры, видеопамять, оперативная память; уровень низко-скоростных устройств образуют любые внешние устройства.

Классическая ЭВМ:

Полувековая история развития вычислительной техники была связана с совершенствованием классической структуры компьютера, имеющей следующие отличительные признаки:

• ядро компьютера - процессор - единственный вычислитель в структуре, дополненный каналами обмена информацией и памятью;

• линейная организация ячеек всех видов памяти фиксированного размера;

• одноуровневая адресация ячеек памяти, стирающая различия между всеми типами информации;

• внутренний машинный язык низкого уровня, при котором команды содержат элементарные операции преобразования простых операндов;

• последовательное централизованное управление вычислениями;

• достаточно примитивные возможности устройства ввода-вывода.

Развитие структур ЭВМ.

Дальнейшее поступательное развитие вычислительной техники напрямую связано с переходом к параллельным вычислениям, с идеями построения многопроцессорных систем и сетей, объединяющих большое количество отдельных процессоров и (или) ЭВМ. С появлением в начале нового тысячелетия многоядерных микропроцессоров эра компьютеров классической структуры и связанных с ними последовательных вычислений заканчивается. На смену идут новые параллельные структуры с новыми принципами их построения. Они становятся экономически более выгодными. Будущее вычислительной техники - именно за этими системами. Появление третьего поколения ЭВМ было обусловлено переходом от транзисторов к интегральным микросхемам. Значительные успехи в миниатюризации электронных схем не просто способствовали уменьшению размеров базовых функциональных узлов ЭВМ, но и создали предпосылки для существенного роста быстродействия процессора. Возникло существенное противоречие между высокой скоростью обработки информации внутри машины и медленной работой устройств ввода-вывода, в большинстве своем содержащих механически движущиеся части. Процессор, руководивший работой внешних устройств, значительную часть времени был бы вынужден простаивать в ожидании информации “из внешнего мира”, что существенно снижало бы эффективность работы всей ЭВМ в целом. Для решения этой проблемы возникла тенденция к освобождению центрального процессора от функций обмена и к передаче их специальным электронным схемам управления работой внешних устройств. Такие схемы имели различные названия: каналы обмена, процессоры ввода-вывода, периферийные процессоры. Последнее время все чаще используется термин “контроллер внешнего устройства” (или просто контроллер). Таким образом, наличие интеллектуальных внешних устройств может существенно изменять идеологию обмена. Центральный процессор при необходимости произвести обмен выдает задание на его осуществление контроллеру. Дальнейший обмен информацией может протекать под руководством контроллера без участия центрального процессора. Для связи между отдельными функциональными узлами ЭВМ используется общая шина (часто ее называют магистралью). Шина состоит из трех частей: шина данных, по которой передается информация; шина адреса, определяющая, куда передаются данные; шина управления, регулирующая процесс обмена информацией. Описанную схему легко пополнять

новыми устройствами – это свойство называют открытостью архитектуры. Для пользователя открытая архитектура означает возможность свободно выбирать состав внешних устройств для своего компьютера, т.е. конфигурировать его в зависимости от круга решаемых задач. Новый вид памяти – видео-ОЗУ (видеопамять). Его появление связано с разработкой особого устройства вывода – дисплея. Остановимся еще на одной важной особенности структуры современных ЭВМ. Поскольку процессор теперь перестал быть центром конструкции, стало возможным реализовывать прямые связи между устройствами ЭВМ. На практике чаще всего используют передачу данных из внешних устройств в ОЗУ и наоборот. Режим, при котором внешнее устройство обменивается непосредственно с ОЗУ без участия центрального процессора, называется прямым доступом к памяти (ПДП). Для его реализации необходим специальный контроллер. Подчеркнем, что режим ПДП в машинах первого и второго поколений не существовал. При увеличении потоков информации между устройствами ЭВМ единственная магистраль перегружается, что существенно тормозит работу компьютера. Поэтому в состав ЭВМ могут вводиться одна или несколько дополнительных шин. Например, одна шина может использоваться для обмена с памятью, вторая -для связи с “быстрыми”, а третья – с “медленными” внешними устройствами. Отметим, что высокоскоростная шина данных ОЗУ обязательно требуется при наличии режима ПДП. Завершая обсуждение особенностей внутренней структуры современных ЭВМ, укажем несколько характерных тенденций в ее развитии. Во-первых, постоянно расширяется и совершенствуется набор внешних устройств, что приводит, как описывалось выше, к усложнению системы связей между узлами ЭВМ. Во-вторых, вычислительные машины перестают быть однопроцессорными. Помимо центрального, в компьютере могут быть специализированные процессоры для вычисления с плавающей запятой (так называемые математические сопроцессоры), видеопроцессоры для ускорения вывода информации на экран дисплея и т.п. Развитие методов параллельных вычислений также вызывает к жизни вычислительные системы достаточно сложной структуры, в которых одна операция выполняется сразу несколькими процессорами. В-третьих, наметившееся стремление иметь быстродействующие машины не только для вычислений, но и для логического анализа информации, также может привести в ближайшие годы к серьезному пересмотру традиционной фон-неймановской архитектуры.

## 3. Модули памяти на системной плате

Со времен появления полупроводниковой памяти и до начала 90-х годов все микросхемы памяти производились, продавались и устанавливались в виде отдельных микросхем. Эти микросхемы вмещали от 1 Кбит до 1 Мбит информации и выше. В первых персональных компьютерах часто оставлялись пустые разъемы, чтобы покупатель в случае необходимости мог вставить дополнительные микросхемы памяти.

В настоящее время распространен другой подход. Группа микросхем (обычно 8 или 16) монтируется на одну крошечную печатную плату и продается как один блок. Он называется SIMM (Single Inline Memory Module — модуль памяти с односторонним расположением выводов) или DIMM (Dual Inline Memory Module — модуль памяти с двухсторонним расположением выводов). На платах SIMM устанавливается один краевой разъем с 72 контактами; при этом скорость передачи данных за один тактовый цикл составляет 32 бит. Модули DIMM, как правило, снабжаются двумя краевыми разъемами (по одному на каждой стороне платы) с 120 контактами; таким образом, общее количество контактов достигает 240, а скорость передачи данных возрастает до 64 бит за цикл. В настоящее время наиболее распространенными являются DDR3 DIMM — третья версия двухскоростных модулей памяти.

Обычно модули DIMM содержат 8 микросхем по 256 Мбайт каждая. Таким образом, весь модуль вмещает 2 Гбайт информации. Во многих компьютерах предусматривается возможность установки четырех модулей; следовательно, при использовании модулей по 2 Гбайт общий объем памяти достигает 8 Гбайт (и более при использовании модулей большей емкости).

В портативных компьютерах обычно используется модуль DIMM меньшего размера, который называется SO-DIMM (Small Outline DIMM). Модули SIMM и DIMM могут содержать бит четности или код исправления ошибок, однако, поскольку вероятность возникновения ошибок в модуле составляет примерно одну ошибку за 10 лет, в большинстве обычных компьютеров схемы обнаружения и исправления ошибок не применяются.

ОЗУ и ПЗУ

ОЗУ (оперативное запоминающее устройство), или RAM (Random Access Memory — оперативная память) разделяется на 2 типа: статическую и динамическую.

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

В динамическом ОЗУ (Dynamic RAM, DRAM), напротив, триггеры не используются.

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

Поскольку динамическому ОЗУ нужен только один транзистор и один конденсатор на бит (статическому ОЗУ требуется в лучшем случае 6 транзисторов на бит), динамическое ОЗУ имеет очень высокую плотность записи (много битов на одну микросхему). По этой причине основная память почти всегда строится на основе динамических ОЗУ. Однако динамические ОЗУ работают очень медленно

(время доступа занимает десятки наносекунд). Таким образом, сочетание кэшпамяти на основе статического ОЗУ и основной памяти на основе динамического ОЗУ соединяет в себе преимущества обоих устройств.

Существует несколько типов динамических ОЗУ. Самый древний тип, который все еще используется, — FPM (Fast Page Mode — быстрый постраничный режим). Это ОЗУ представляет собой матрицу битов. Аппаратное обеспечение представляет адрес строки, а затем — адреса столбцов. Благодаря явно передаваемым сигналам память работает асинхронно по отношению к главному тактовому генератору системы.

FPM постепенно замещается памятью EDO (Extended Data Output — память с расширенными возможностями вывода), которая позволяет обращаться к памяти еще до того, как закончилось предыдущее обращение. Такой конвейерный режим, хотя и не ускоряет доступ к памяти, повышает пропускную способность, позволяя получить больше слов в секунду.

Память типа FPM и EDO сохраняла актуальность в те времена, когда продолжительность цикла работы микросхем памяти не превышала 12 нс. Впоследствии, с увеличением быстродействия процессоров, сформировалась потребность в более быстрых микросхемах памяти, и тогда на смену асинхронным режимам FPM и EDO пришли синхронные динамические ОЗУ (Synchronous DRAM, SDRAM). Синхронное динамическое ОЗУ управляется от главного системного тактового генератора. Данное устройство представляет собой гибрид статического и динамического ОЗУ. Основное преимущество синхронного динамического ОЗУ состоит в том, что оно исключает зависимость микросхемы памяти от управляющих сигналов. ЦП сообщает памяти, сколько циклов следует выполнить, а затем запускает ее. Каждый цикл на выходе дает 4, 8 или 16 бит в зависимости от количества выходных строк. Устранение зависимости от управляющих сигналов приводит к ускорению передачи данных между ЦП и памятью.

Следующим этапом в развитии памяти SDRAM стала память DDR (Double Data Rate — передача данных

с двойной скоростью). Эта технология предусматривает вывод данных как на фронте, так и на спаде импульса, вследствие чего скорость передачи увеличивается вдвое. Например, 8-разрядная микросхема такого типа, работающая с частотой 200 МГц, дает на выходе два 8-разрядных значения 200 миллионов раз в секунду (разумеется, такая скорость удерживается в течение небольшого периода времени); таким образом, теоретически, кратковременная скорость может достигать 3,2 Гбайт/с. Интерфейсы памяти DDR2 и DDR3 обеспечивают дополнительный прирост производительности по сравнению с DDR за счет повышения скорости шины памяти до 533 МГц и 1067 МГц соответственно. На момент издания книги самые быстрые микросхемы DDR3 могли выдавать данные на скорости 17,067 Гбайт/с.

ПЗУ:

Во многих случаях данные должны сохраняться даже при отключенном питании. Более того, после установки ни программы, ни данные не должны изменяться. Эти требования привели к появлению ПЗУ (постоянных запоминающих устройств), или ROM (Read-Only Memory — постоянная память). ПЗУ не позволяют изменять и стирать хранящуюся в них информацию (ни умышленно, ни случайно). Данные записываются в ПЗУ в процессе производства. Для этого изготавливается трафарет с определенным набором битов, который накладывается на фоточувствительный материал, а затем открытые (или закрытые) части поверхности вытравливаются. Единственный способ изменить программу в ПЗУ — поменять всю микросхему.

Чтобы компаниям было проще разрабатывать новые устройства, основанные на ПЗУ, были выпущены программируемые ПЗУ (Programmable ROM, PROM). В отличие от обычных ПЗУ, их можно программировать в условиях эксплуатации, что позволяет сократить время исполнения заказа. Многие программируемые ПЗУ содержат массив крошечных плавких перемычек. Чтобы пережечь

определенную перемычку, нужно выбрать требуемые строку и столбец, а затем приложить высокое напряжение к определенному выводу микросхемы.

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

Следующий этап — электронно перепрограммируемое ПЗУ (Electronically EPROM, EEPROM), которое не нужно для этого помещать в специальную камеру, чтобы подвергнуть воздействию ультрафиолетовых лучей — для стирания информации достаточно подать соответствующие импульсы. Кроме того, чтобы перепрограммировать данное устройство, его не нужно вставлять в специальный аппарат для программирования, в отличие от стираемого программируемого ПЗУ. В то же время самые большие электронно перепрограммируемые ПЗУ в 64 раза меньше обычных стираемых ПЗУ, и работают они в два раза медленнее. Электронно перепрограммируемые ПЗУ не могут конкурировать с динамическими и статическими ОЗУ, поскольку работают в 10 раз медленнее, их емкость в 100 раз меньше, и они стоят гораздо дороже. Они используются только в тех ситуациях, когда необходимо сохранять информацию при выключении питания.

Более современный тип электронно перепрограммируемого ПЗУ — флэш-память. В отличие от стираемого ПЗУ, которое стирается под воздействием ультрафиолетовых лучей, и от электронно перепрограммируемого ПЗУ, которое стирается по байтам, флэш-память стирается и записывается блоками. Многие изготовители производят небольшие печатные платы, содержащие до 64 Гбайт флэш-памяти. Они используются для хранения изображений в цифровых камерах и для других целей. Флэш-память постепенно начинает вытеснять диски, что будет грандиозным шагом вперед, учитывая время доступа в 50 нс. Флэш-память обеспечивает лучшее время доступа при более низком энергопотреблении; с другой стороны, стоимость одного бита флэш-памяти существенно выше, чем у дисков.

## **4. Структура основной памяти ЭВМ**

**Память** — это тот компонент компьютера, в котором хранятся программы и данные.

**Бит**

Основной единицей хранения данных в памяти является двоичный разряд, который называется битом. Бит может содержать 0 или 1. Эта самая маленькая единица памяти. Некоторые компьютеры, например мэйнфреймы IBM используют двоично-десятичный код.

**Адреса памяти**

Память состоит из ячеек, каждая из которых может хранить некоторую порцию информации. Каждая ячейка имеет номер, который называется адресом. По адресу программы могут ссылаться на определенную ячейку. Если память содержит n ячеек, они будут иметь адреса от 0 до n — 1. Все ячейки памяти содержат одинаковое число битов. Если ячейка состоит из k бит, она может содержать любую из 2k комбинаций.

В компьютерах, в которых используется двоичная система счисления адреса памяти также выражаются в двоичных числах. Если адрес состоит из m бит, максимальное число адресуемых ячеек составит 2^m.

**Ячейка** — минимальная адресуемая единица памяти. В последние годы практически все производители выпускают компьютеры с 8-разрядными ячейками, которые называются байтами. Байты группируются в слова. В компьютере с 32-разрядными словами на каждое слово приходится 4 байт, а в компьютере с 64-разрядными словами — 8 байт. Такая единица, как слово, необходима, поскольку большинство команд производят операции над целыми словами (например, складывают два слова).

**Упорядочение байтов**

Байты в слове могут нумероваться слева направо или справа налево. И если на одной ЭВМ стоит **прямой порядок следования байтов**, а на **другой обратный порядок следования байтов**, то при передачи информации между этими машинами слова будут инвертироваться и искажаться. Исправить это простым преобразованием прямого порядка в обратный не выйдет, тк слова все еще будут искажены. Простого решения не существует. Есть один способ, но он неэффективен. (Нужно перед каждой единицей данных помещать заголовок, информирующий, какой тип данных следует за ним — строка, целое и т. д. Это позволит компьютеру-получателю производить только необходимые преобразования.)

**Код исправления ошибок**

В памяти компьютера из-за всплесков напряжения и по другим причинам время от времени могут возникать ошибки. Для борьбы с ними используются специальные коды, умеющие обнаруживать и исправлять ошибки. В этом случае к каждому слову в памяти добавляются дополнительные биты. Когда слово считывается из памяти, эти доп. биты проверяются, что и позволяет обнаруживать ошибки. Если в памяти обнаруживается недопустимое кодовое слово, компьютер знает, что произошла ошибка.

Если слово = m бит, к которым мы дополнительно прибавляем r бит (контрольных разрядов). Тогда общая длина слова составит n бит (n = m + r). n часто называют **кодовым словом.**

Число битовых позиций, по которым отличаются два слова, называется **интервалом Хэмминга**. Если интервал Хэмминга для двух слов равен d, значит, достаточно d одноразрядных ошибок, чтобы превратить одно слово в другое.

**Кэш-память**

Существует проблема разницы в скорости процессоров и памяти. Память всегда была медленнее и из-за этого процессору так же приходилось ждать. Эту проблему решает **кэш-память** - Память небольшого объема с высокой скоростью работы.

**Основная идея кэш-памяти проста**: в ней находятся слова, которые чаще всего используются. Если процессору нужно какое-нибудь слово, сначала он обращается к кэш-памяти, и только если слова нет, обращается к основной памяти.

Ситуация, когда при последовательных обращениях к памяти в течение некоторого промежутка времени используется только небольшая ее область, называется **принципом локальности**.

Есть **объединенная кэш-память** (unified cache), в которой хранятся и данные, и команды. В этом случае вызов команд и данных автоматически уравновешивается. Однако в настоящее время существует тенденция к использованию **разделенной кэш-памяти** (split cache), когда команды хранятся в одной кэш-памяти, а данные — в другой. Такая архитектура также называется **гарвардской.** Сейчас использую **разделенную кэш-память**,тк она позволяет осуществлять параллельный доступ к данным и командам

**Сборка модулей памяти и их типы**

Группа микросхем (обычно 8 или 16) монтируется на одну крошечную печатную плату и продается как один блок. Он называется **SIMM** (Single Inline Memory Module — модуль памяти с односторонним расположением выводов) или **DIMM** (Dual Inline Memory Module — модуль памяти с двухсторонним расположением выводов). На платах SIMM устанавливается один краевой разъем с 72 контактами; при этом скорость передачи данных за один тактовый цикл составляет 32 бит. Модули DIMM, как правило, снабжаются двумя краевыми разъемами (по одному на каждой стороне платы) с 120 контактами; таким образом, общее количество контактов достигает 240, а скорость передачи данных возрастает до 64 бит за цикл. В настоящее время наиболее распространенными являются DDR4 DIMM — четвертая версия двухскоростных модулей памяти.(по учебнику DDR3)

В портативных компьютерах обычно используется модуль DIMM меньшего размера, который называется **SO-DIMM** (Small Outline DIMM). Модули SIMM и DIMM могут содержать бит четности или код исправления ошибок, однако, поскольку вероятность возникновения ошибок в модуле составляет примерно одну ошибку за 10 лет, в большинстве обычных компьютеров схемы обнаружения и исправления ошибок не применяются.

## **5. Внешняя кэш-память процессора**

Кэш-память процессора изготавливают в виде микросхем статической памяти (англ. Static Random Access Memory, сокращенно - SRAM). По сравнению с другими типами памяти, статическая память обладает очень высокой скоростью работы.

Однако, эта скорость зависит также от объема конкретной микросхемы. Чем значительней объем микросхемы, тем сложнее обеспечить высокую скорость ее работы.

Учитывая указанную особенность, кэш-память процессора изготовляют в виде нескольких небольших блоков, называемых уровнями. В большинстве процессоров используется трехуровневая система кэша:

• Кэш-память первого уровня или L1 (от англ. Level - уровень) – очень маленькая, но самая быстрая и наиболее важная микросхема памяти. Ни в одном процессоре ее объем не превышает нескольких десятков килобайт. Работает она без каких-либо задержек. В ней содержатся данные, которые чаще всего используются процессором.

Количество микросхем памяти L1 в процессоре, как правило, равно количеству его ядер. Каждое ядро имеет доступ только к своей микросхеме L1.

• Кэш-память второго уровня (L2) немного медленнее кэш-памяти L1, но и объем ее более существенный (несколько сотен килобайт). Служит она для временного хранения важной информации, вероятность запроса которой ниже, чем у информации, находящейся в L1.

• Кэш-память третьего уровня (L3) – еще более объемная, но и более медленная схема памяти. Тем не менее, она значительно быстрее оперативной памяти. Ее размер может достигать нескольких десятков мегабайт. В отличие от L1 и L2, она является общей для всех ядер процессора.

Уровень L3 служит для временного хранения важных данных с относительно низкой вероятностью запроса, а также для обеспечения взаимодействия ядер процессора между собой.

Встречаются также процессоры с двухуровневой кэш-памятью. В них L2 совмещает в себе функции L2 и L3.

## 6. **Арифметические операции над двоичными числами с плавающей точкой**

## В современных ЭВМ числа с плавающей точкой хранятся в памяти машин, имея мантиссу и порядок (характеристику) в прямом коде и нормализованном виде. Все арифметические действия над этими числами выполняются так же, как это делается с ними, если они представлены в полулогарифмической форме (мантисса и десятичный порядок) в десятичной системе счисления. Порядки и мантиссы обрабатываются раздельно.

## Сложение (вычитание). Операция сложения (вычитания) производится в следующей последовательности.

## 1. Сравниваются порядки (характеристики) исходных чисел путем их вычитания р=р1-р2. При выполнении этой операции

## определяется, одинаковый ли порядок имеют исходные слагаемые.

## 2. Если разность порядков равна нулю, то это значит, что одноименные разряды мантисс имеют одинаковые веса

## (двоичный порядок). В противном случае должно проводиться выравнивание порядков.

## 3. Для выравнивания порядков число с меньшим порядком сдвигается вправо на разницу порядков р. Младшие

## выталкиваемые разряды при этом теряются.

## 4.После выравнивания порядков мантиссы чисел можно складывать (вычитать) в зависимости от требуемой операции. Операция вычитания заменяется операцией сложения в соответствии с данными табл. 2.3. Действия над слагаемыми производятся

## вОК или ДК по общим правилам.

## 5.Порядок результата берется равным большему порядку pрез =max (p1,p2).

## 6.Если мантисса результата не нормализована, то осуществляются нормализация и коррекция значений порядка

## Умножение (деление). Операция умножения (деления) чисел с плавающей точкой также требует разных действий над порядками и мантиссами. Алгоритмы этих операций выполняются в следующей последовательности.

## 1.При умножении (делении) порядки складываются (вычитаются) так, как это делается над числами с фиксированной

## точкой.

## 2.При умножении (делении) мантиссы перемножаются (делятся).

## 3.Знаки произведения (частного) формируются путем сложения знаковых разрядов сомножителей (делимого и делителя). Возможные переносы из знакового разряда игнорируются.

## Арифметические операции над двоично-десятичными кодами чисел

## При обработке больших массивов экономической информации переводы чисел из десятичной системы в двоичную и обратно могут требовать значительного машинного времени. Некоторые образцы ЭВМ поэтому имеют или встроенные, или подключаемые блоки, которые обрабатывают десятичные целые числа в их двоично-десятичном представлении. Действия над ними также приводятся к операции алгебраического сложения отдельных цифр чисел

## 1.Сложение чисел начинается с младших цифр (тетрад) и производится с учетом возникающих переносов из младших разрядов в старшие.

## 2.Знак суммы формируется специальной логической схемой по знаку большего слагаемого.

## 3.Для того чтобы при сложении двоично-десятичных цифр возникали переносы, аналогичные при сложении чисел в десятичном представлении, необходимо проводить так называемую десятичную коррекцию. Для этого к каждой тетраде первого

## числа прибавляется дополнительно по цифре 610=01102, что позволяет исключить шесть неиспользуемых комбинаций (10101111)2, так как они кодируют шестнадцатеричные цифры A-F (числа 10-1510).

## 4.После операции суммирования осуществляется корректировка суммы. Из тех тетрад суммы, из которых не было переносов, изымаются ранее внесенные избытки 610=01102. Для этого проводится вторая коррекция. Операция вычитания заменяется, как и обычно, операцией сложения с числом -6,представленным дополнительным кодом 10102, но только в тех разрядах, в которых отсутствовали переносы. При этой второй коррекции переносы из тетрад блокируются.

## 5. Операция вычитания реализуется достаточно своеобразно. По общему правилу сложения (п.п.1-4) к тетрадам числа с большим модулем прибавляются дополнительные коды тетрад другого числа. В качестве знаке результата берется знак числа с большим модулем

## 

## 7. **Синхронизация. «Разгон» и «торможение» процессора**

Основной тактовый генератор системной платы вырабатывает высокостабильную опорную частоту, которая используется для синхронизации процессора, системной шины и шин ввода вывода. Стандартные частоты генератора 4,77; 6; 8; 10; 12; 16; 20; 25; 33,3; 40; 50; 60; 66,6; 75; 83; 100; 125 МГц. При наличии переключателя Turbo компьютер с процессором 286, 386 частота выбирается из пары частот. В процессорах 486 и старше частота фиксирована, а переключатель Turbo может отключать кэширование или включать режим прерывистой синхронизации.

Так как быстродействие различных компонентов компьютеров существенно различается, применяется деление опорной частоты для синхронизации шин ввода вывода и внутреннее умножение частоты в процессорах. Различают следующие частоты:

· Host Bus - частота системной шины (внешняя частота шины процессора). Она является опорной для всех других и устанавливается джемперами с перемычками. Микропроцессоры Pentium используют частоты от 50 до 125 МГц. Частоты 75 МГц и выше требуют высокие технологии изготовления системных плат, чипсетов и микросхем обрамления;

· CPU Clock - внутренняя частота процессора на которой работает его вычислительное ядро, для ее получения применяется внутреннее умножение частоты на 1,5; 2; 2,5; 3; 3,5; 4 и некоторые другие значения. Коэффициент умножения задается с помощью джемперов на системной плате, которые заземляют определенные выводы микропроцессора. Трактовка управляющих сигналов при этом зависит от марки модели процессора;

· PCI Bus Clock - это частота шины PCX, которая должна составлять 25; 33,3 МГц. Обычно она обеспечивается делением Host Bus Clock на 2 или на 3. Оптимальная частота для процессоров 5 и 6 поколений 66,6 МГц;

· VLB Bus Clock - это частота шины VBL определяемая аналогично предыдущей;

· ISA Bus Clock - это частота шины ISA, которая должна быть близка к 8 МГЦ. Она обычно задается в BIOS Setup через коэффициент деления системной шины.

Разгон и торможение процессора

Фирмы производители гарантируют стабильную длительную работу на заявляемых частотах при опорном питающем напряжении. При снижении питающего напряжения уменьшается рассеиваемая мощность и допустимая частота синхронизации.

Повышение тактовой частоты увеличивает потребляемую мощность, но для обеспечения стабильной работы внутренней логики требуются некоторые повышения питающего напряжения. Возможность выбора внешней частоты синхронизации и внутреннего коэффициента умножения процессора создает иллюзию возможности разгона процессора. Здесь есть следующие варианты:

· подъем входной частоты с 60 до 66,66 МГц для которого конфигурирование самого процессора не изменяется, однако процессоры с частотой ядра 100; 133; 166 Мгц таким способом не разгонятся;

· увеличение коэффициента умножения который воспринимается даже процессорами с частотой ядра 75 МГц, однако коэффициент 2,5 воспринимают только процессоры начиная со стопинга С;

· комбинированные изменения коэффициента умножения и внешней частоты. Эти изменения могут производиться в пределах штатной спецификации.

На системных платах предназначенных не только для процессоров Intel возможны и более радикальные варианты на которые идут для повышения в основном производительности шины PCI. Это дает эффект на конкретных приложениях. Фирма Intel отказалась от повышения внешней частоты до 75 МГц, так как это не дает принципиального роста пропускной способности. Эффект от разгона процессора теряется за счет относительно медленной памяти, так как при переходе на более высокую частоту системной шины BIOS совместно с чипсетом увеличивает число тактов ожидания в циклах памяти

Варианты разгона Pentium

Pentium 100 до 120, 133; 133 до 150, 166; 166 до 180, 200. При экспериментах с разгоном не следует сразу повышать питающее напряжение, необходимо после повышения частоты постоянно следить за степенью нагрева процессора и радиатора стабилизатора напряжения. Проверкой работы системы может служить длительная активная работа в многозадачном режиме с каким-либо тяжелым приложением. Следует иметь в виду, что фирма производитель ставит маркировку частоты исходя из обоснованных критериев качества и надежности, не забывая про получение прибыли.

## 8. **Адресация информации в ЭВМ**

Адресный код - это информация о адресе операнда, содержащаяся в команде. Исполнительный адресок - это номер ячейки памяти, к которой делается фактическое обращение. В современных ЭВМ адресный код, обычно, не совпадает с исполнительным адресом. Выбор методов адресации, формирования исполнительного адреса и преобразования адресов является одним из важных процессов разработки ЭВМ. Разглядим методы адресации, используемые в современных ЭВМ: 1) Подразумеваемый операнд. В команде может не содержаться явных указаний о операнде; в данном случае операнд предполагается и практически задается кодом операции команды. 2) Подразумеваемый адресок. В команде может не содержаться явных указаний о адресе участвующего в операции операнда либо адресе, по которому должен быть расположен итог операции, но этот адресок предполагается. 3) Конкретная адресация. В команде содержится не адресок операнда, а конкретно сам операнд. При конкретной адресации не требуется обращения к памяти для подборки операнда и ячейки памяти для его хранения. Это содействует уменьшению времени выполнения программы и занимаемо го ею размера памяти. Конкретная адресация комфортна для хранения различного рода констант. 4) Ровная адресация. В адресной части команды быть может конкретно указан исполнительный адресок. 5) Относительная (базисная) адресация. Относительная адресация дозволяет при наименьшей длине адресного кода

команды обеспечить доступ к хоть какой ячейке памяти 6) Укороченная адресация. в команде задаются лишь младшие разряды адресов, а старшие разря ды при всем этом предполагаются нулевыми. 7) Косвенная адресация. Адресный код команды в данном случае показывает адресок ячейки памяти, в какой находится адресок операнда либо команды. Косвенная адресация обширно употребляется в малых и микро ЭВМ, имеющих короткое машинное слово, для преодоления ограничений недлинного формата команды (вместе употребляются регистровая и косвенная адресация). 8) Адресация слов переменной длины. традиционно реализуется методом указания в команде местоположения в памяти начала слова и его длины. 9) Стековая адресация. Стек представляет собой группу поочередно пронумерованных регистров либо ячеек памяти, снабженных указателем стека, в каком автоматом при записи и считывании устанавливается номер (адресок) крайней занятой ячейки стека (вершины стека). При операции записи заносимое в стек слово помещается в последующую по порядку вольную ячейку стека, а при считывании из стека извлекается крайнее поступившее в него слово. 10) Автоинкрементная и автодекрементная адресации. При автоинкрементной адресации поначалу содержимое регистра употребляется как адресок операнда, а потом получает приращение, равное числу б в элементе массива. При автодекрементной адресации поначалу содержимое указанного в команде регистра уменьшается на число б в элементе массива, а потом употребляется как адресок операнда. 11) Индексация. Для выполнения индексации вводятся так именуемые индексные регистры. Исполнительный адресок при индексации формируется методом сложения адресного кода команды (смещения) с содержимым индексного регистра

(индексом), а при наличии базирования - и с базисным адресом. Для управления индексацией употребляются команды, задающие операции над содержимым индексных регистров - команды индексной математики. Можно отметить главные виды индексных операций: - засылка в соответственный индексный регистр исходного значения индекса; - изменение индекса; - проверка окончания повторяющихся вычислений.

## 9. **Шина PCI. . Ускоренный графический порт (AGP). Шина USB**

## Предшественницы шины PCI(ISA(2 байта за цикл, частота 8,33 МГц), EISA(4 байта за цикл)) имели низкую пропускную способность 16,7 Мбайт/с и 33,3 Мбайт/с соответственно, но для показа полноэкранных видео, игр и других новых приложений требовалось скорость по меньшей мере 135 Мбайт/с.

## «В 1990 году компания Intel разработала новую шину с гораздо более высокой пропускной способностью» - шину PCI.

## Первая шина PCI передавала 32 бита за цикл, а частота была 33МГц, тем самым пропускная способность составляла 133 Мбайта/с

## Потом была PCI 2.0, PCI 2.1, PSI 2.2 в конце концов PCI выдавала 64 бита за цикл, частоту имела 66 МГц и пропускная способность составляла 528 Мбайт/с (учитывая, что остальные компоненты помимо шины, справлялись с частотой), когда для Full HD требуется 155 Мбайт/с.

## Все же здесь есть некоторые проблемы. Во-первых, этого недостаточно для шины памяти. Во-вторых, шина PCI несовместима со всеми старыми платами ISA. По этой причине компания Intel решила разрабатывать компьютеры с тремя и более шинами

## Ключевыми компонентами данной архитектуры являются мосты между шинами Мост PCI связывает центральный процессор, память и шину PCI. Мост ISA связывает шину PCI с шиной ISA, а также поддерживает один или два IDE-диска

## Шины PCI идут 3 видов они поддерживают разные напряжения, для старых PC 3.3 В, для новых 5 В, есть универсальные для тех и других PC

## Типы плат так же разные 32(120 выводов) и 64(120+64 вывода) разрядные, как мы знаем 64 поддерживают и 32, но не наоборот.

## PCI Express до 16 Гбайт/с

## Шины PCI являются синхронными, как и все шины PC, восходящие к первой модели IBM PC. Все транзакции в шине PCI осуществляются между задающим и подчиненным устройствами.

## Ускоренный графический порт (AGP)

## Предназначена исключительно для обмена данными с графическим адаптером.

## AGP 1.0 Начальная скорость была 264 Мбайта/с (1x)

## AGP 3.0 имела уже 2,1 Гбайт/с (8x), но она уступает сегодняшним PCI Express

## Шина USB

## Шины PCI и PCI Express очень хорошо подходят для соединения высокоскоростных периферийных устройств, но использовать интерфейс PCI для низкоскоростных устройств ввода-вывода (например, мыши и клавиатуры) было бы неэффективно, поэтому была придумана шина USB (Universal Serial Bus — универсальная последовательная шина) 9 большими собравшимися в месте компаниями

## 1. пользователи не должны устанавливать переключатели и перемычки на платах и устройствах;

## 2. пользователи не должны открывать компьютер, чтобы установить новые устройства ввода-вывода;

## 3. должен существовать только один тип кабеля, подходящий для соединения всех устройств;

## 4. устройства ввода-вывода должны получать питание через кабель;

## 5. должна быть возможность подсоединения к одному компьютеру до 127 устройств;

## 6. система должна поддерживать устройства реального времени (например, звуковые устройства, телефон);

## 7. должна быть возможность устанавливать устройства во время работы компьютера;

## 8. установка нового устройства не должна требовать перезагрузки компьютера;

## 9. производство новой шины и устройств ввода-вывода для нее не должны требовать больших затрат.

## Шина USB удовлетворяет всем этим условиям. Она разработана для низкоскоростных устройств (клавиатур, мышей, фотоаппаратов, сканеров, цифровых телефонов и т. д.). Общая пропускная способность первой версии шины (USB 1.0) составляла 1,5 Мбит/с

## USB 1.1 - 12 Мбит/с

## USB 2.0 - 480 Мбит/с

## USB 3.0 - 5 Гбит/с

## Кабель состоит из четырех проводов: два из них предназначены для передачи данных, один — для питания (+5 В) и один — для земли. Система передает 0 изменением напряжения, а 1 — отсутствием изменения напряжения, поэтому длинная последовательность нулевых битов порождает поток регулярных импульсов

## Ровно через каждую миллисекунду (±0,05 мс) корневой хаб передает новый кадр, чтобы синхронизировать все устройства во времени. Кадр состоит из пакетов, первый из которых передается от хаба к устройству. Следующие пакеты кадра могут передаваться в том же направлении, а могут и в противоположном (от устройства к хабу)

## 

## 10. **Структура и система команд ЭВМ**

Решение задач на ЭВМ реализуется программным способом, то есть путем выполнения последовательно во времени отдельных операций,предусмотренных алгоритмом решения задачи, над данными.Программа содержит совокупность различных команд, обеспечивающих требуемую обработку входных данных.

Команда машинной программы – это элементарная инструкция машине, выполняемая ею автоматически без каких-либо дополнительных указаний и пояснений.

КОП Адресная часть

Команда состоит из двух частей: операционной и адресной. Операционная часть команды (код операции -КОП), содержит указание процессору на действия, которые необходимо выполнить над данными. Адресная часть команды –это группа разрядов в команде, в которых записываются коды адреса (адресов) ячеек памяти машины, предназначенных для оперативного хранения данных.

По количеству адресов, записанных в команде, команды делятся на безадресные, одно-, двух- и трехадресные:

- типовая структура трехадресной команды:

КОП а1 а2 а3

а1 и а2 — адреса ячеек (регистров), где расположены соответственно первое и второе числа, участвующие в операции, а3 — адрес ячейки (регистра), куда следует поместить число, полученное в результате выполнения операции;

- типовая структура двухадресной команды:

КОП а1 а2

а1 — это обычно адрес ячейки (регистра), где хранится первое из чисел, участвующих в операции, и куда после завершения операции должен быть записан результат операции, а2 — обычно адрес ячейки (регистра), где хранится второе участвующее в операции число;

- типовая структура одноадресной команды :

КОП а1

а1 в зависимости от модификации команды может обозначать либо адрес ячейки (регистра), где хранится одно из чисел, участвующих в операции, либо адрес ячейки (регистра ), куда следует поместить число — результат операции.

Безадресная команда содержит только код операции, а информация для нее должна быть заранее помещена в определенные регистры машины (безадресные команды могут использоваться только совместно с командами другой адресности).Наибольшее применение в ПК нашли двухадресные команды.

Множество команд, реализованных в конкретной ЭВМ, образует ее систему команд.

Современные ПЭВМ автоматически выполняют несколько сотен различных команд. Например, стандартный набор современных компьютеров содержит более 240 машинных команд.

Все машинные команды можно разделить на группы по видам выполняемых операций:

- операции пересылки данных внутри компьютера;

- арифметические операции над данными;

- логические операции над данными;

- операции над строками (текстовой информацией);

- операции обращения к внешним устройствам компьютера;

- операции передачи управления;

- обслуживающие и вспомогательные операции.

Пояснения требуют операции передачи управления (или, иначе, — ветвления программы), которые служат для изменения естественного порядка выполнения команд.

Существуют операции безусловной передачи управления и операции условной передачи управления.

Операции безусловной передачи управления всегда обусловливают выполнение после данной команды не следующей по порядку, а той, адрес которой в явном или неявном виде указан в адресной части команды.

Операции условной передачи управления вызывают тоже передачу управления по адресу, указанному в адресной части команды, но только в том случае, если выполняется некоторое заранее оговоренное для этой команды условие. Это условие в яв-ном или неявном виде указано в коде операции команды. Команд условной передачи управления насчитывается обычно до нескольких десятков — по числу используемых условий.

Команд безусловных передач управления обычно только три:

- команда передачи управления, которая просто передает управление по заданному адресу и больше никаких действий не выполняет;

- команда передачи управления (ее часто называют командой вызова процедуры или подпрограммы), которая кроме передачи управления процедуре еще и запоминает в специальной стековой памяти адрес следующей команды (адрес возврата из процедуры);

- безадресная команда передачи управления (команда возврата из процедуры), возвращающая управление по запомненному адресу возврата.

Вторая и третья из названных команд безусловных передач управления работают «на пару» — одна передает управление процедуре, другая возвращает из нее. Важную роль в выполнении этих команд передачи управления (да и при многих других ситуациях, отрабатываемых компьютером) играет специальным образом организованная область оперативной памяти — стековая память. Обращение к ячейкам этой памяти выполняется по принципу «последний записанный операнд первым считывается» или, иначе, «первым вошел — последним вышел» (FILO — first input, last output). Стековая память позволяет удобно реализовать процессы иерархического обращения ко многим процедурам (количество уровней иерархии практически не ограничено), последовательно записывая и выдавая по принципу FILO адреса возврата каждой из них.

## **11. Слоты процессоров. Питание и охлаждение процессоров**

Обычно контроллеры (адаптеры) внешних устройств находятся на отдельных платах, вставляемых в унифицированные разъемы на материнской плате. Через этот разъем контроллеры (адаптеры) подключаются непосредственно к системной магистрали передачи данных в системе компьютер - шина.

Существует два понятия характеризующих два вида разъемов для пpоцессоpа: сокет (Socket) и слот (Slot). Socket (сокет) - плоский разъем для установки микросхемы с выводами, пеpпендикуляpными корпусу. Slot (слот) - это щелевой разъем с контактами по краю. На данный момент существуют несколько типов разъемов для установки процессора в материнскую плату:

Socket 1 (PGA-169) - пpоцессоpы типа 486SX/DX/DX2 (5 В).

Socket 2 (PGA-238) - пpоцессоpы типа 486SX/DX/DX2/DX4 (5 В).

Socket 3 (PGA-237) - пpоцессоpы типа 486SX/DX/DX2/DX4 (5/3.3 В).

Socket 4 (PGA-273) - пpоцессоp ы Pentium-60/66 (5 В ).

Socket 5 (SPGA-320) - пpоцессоpы Pentium-75..120 (3.3 В ).

Socket 6 (PGA-235) - пpоцессоpы типа 486DX4 (3.3 В ).

Socket 7 (SPGA-321) - пpоцессоpы типа Pentium-75..333 (2.2..3.3 В ).

Socket 8 (PGA/SPGA-387) - пpоцессоpы Pentium Pro (P6).

Slot 1 (SEC-242) - пpоцессоpы типа Pentium II.

Slot 2 (SEC-330) - пpоцессоpы типа Pentium II Xeon.

Socket - ZIF ( Zero Input Force - вставляй не прикладывая сил) конструктивно представляет пластиковый разъем с зажимающей защелкой, расположенной сбоку корпуса разъема, предназначенной для предотвращения самопроизвольного выпадения процессора. При установке процессора защелка должна быть максимально поднята вверх.

Socket 7 - стандартный ZIF-разъём с 296 контактами, использующийся всеми процессорами класса Р5 - Intel Pentium, AMD K5 и K6, Cyrix 6x86 и 6x86MX и Centaur Technology IDT-C6.

Socket 8 - нестандартный ZIF имеющий 387 контактов и несовместимый с Socket 7. Предназначен для установки в него процессора класса Р6 - Pentium Pro . Так как ядро процессора и кэш были объединены на одном кристалле то и форма его получилась прямоугольной, а не квадратной как у Socket 7. Он требует multichip module - специальный корпус, включающий кристалл процессора и один или два кристалла SRAM для кэша L 2 (от 256 KB до 1 МБ). Кэш L 2 может работать на частоте процессора

Socket 370 - нестандартный ZIF, несовместим ни с Socket 7, ни с Socket 8. Предназначен для установки более дешевого прототипа P6 Celeron , за исключением модели Celeron II, построенной по технологии Coppermine.

Socket FC-PGA (Flip Chip Pin Grid Array) внешне напоминает Socket 370. В отличие от 370 на FC-PGA заводится два питания 1,5В и 1,6В, т предназначен ин для установки с него процессоров произведенных по технологии Coprmine.

Slot - пластиковый разъем с двумя рядами контактов, в него вставляются процессоры с ножевым разъемом. INTEL пошла на такое в связи с тем, что для удешевления стоимости процессора кэш был вынесен с кристалла и стал располагаться на плате процессора, которая и имеет ножевой двух сторонний разъем.

Slot I - предназначен для установки в него процессора P6 Pentium II ,Pentium III и процессора P6 Celeron Slot I. Это разъём с 242 контактами для дочерних карт, предназначенный для установки процессора, размещённого в Single Edge Contact ( SEC ) картридже. Внутри картриджа размещены кристалл процессора и чипы SRAM (до 512 KB ). На материнской плате могут быть установлены один или два разъёма Slot 1. Кэш L 2 может работать на частоте, составляющей 1, 1/2 или 1/3 частоты процессора.

Slot 2 - отличается от Slot I по коммерческим причинам, так как в него ставятся более дорогие модели процессоров Xeon, стоимость которых во много раза превышает стоимость процессоров Pentium II и Pentium III. Это разъём для дочерних карт, предназначенный для установки процессора в картридже большего размера. Slot 2 не предназначен для замены Slot 1, используется для установки на рабочих станциях и серверах высшего уровня. На материнской плате могут быть установлены до 4 таких разъёмов, а при использовании специальных чипсетов и больше. Дополнительный объём картриджа используется для установки чипов SRAM . Процессоры, устанавливаемые в Slot 2, поддерживают размер кэшируемой памяти до 64 GB . С использованием пакетной (burst) SRAM кэш L 2 может работать на частоте процессора. Частота внешней шины - не менее 100 МГц.

Slot A - практически тот же самый Slot I только перевернутый наоборот, предназначен для установки процессора Athlon от AMD.

**Питание и охлаждение процессоров.**

Процессоры младших поколений (до первых моделей 486) использовали напряжение питания 5 В (за исключением некоторых процессоров для портативных компьютеров). Развитие технологии привело к необходимости и возможности снижения напряжения питания до 3)3 В и ниже. Стандартный блок питания для процессора обеспечивает только питание +5 В, поэтому на системных платах для процессоров с пониженным напряжением питания стали использовать дополнительные регуляторы напряжения VRM (Volt Regulation Module). Эти регуляторы представляют собой микросхему стабилизатора напряжения фиксированного или управляемого уровня. Для питания мощных процессоров она устанавливается на радиаторе, на некоторых системных платах для 486 в качестве теплоотвода используется медная площадка под микросхемой на самой печатной плате. Напряжение управляемых регуляторов задается джамперами, иногда их для отличия делают красного цвета. Установленное значение питающего напряжения должно соответствовать номиналу процессора. Слишком низкое напряжение приводит к неустойчивой работе, слишком высокое может вывести процессор из строя. Для процессоров с раздельным питанием (новейшие модели Pentium, включая ММХ и другие) на плате должно стоять два и даже три регулятора. На плате АТХ он может быть и один, поскольку для питания интерфейса процессора 3,3 В может использоваться непосредственно дополнительная шина источника +3,3 В. Возможен вариант, когда на плате установлен один VRM и имеется разъем для подключения дополнительного. Для процессоров с одним питанием в этом разъеме джамперами соединяется несколько контактов, а для процессоров с раздельным питанием в него нужно вставить дополнительный VRM — не очень распространенное и стандартизованное изделие.

С максимально допустимой мощностью регулятора могут появиться проблемы при использовании процессоров Cyrix, отличающихся повышенным энергопотреблением.

Вопрос охлаждения стал весьма актуальным для пользователей также начиная с процессоров 486. Процессор 486SX 33 еще не требовал установки специального радиатора. Однако с повышением тактовой частоты возрастает мощность, рассеиваемая процессором. Кроме того, потребляемая мощность зависит от интенсивности работы процессора: разные инструкции задействуют различный объем внутреннего оборудования процессоров, и при увеличении доли «энергоемких» инструкций мощность, рассеиваемая процессором, повышается. Существуют даже специальные тестовые программы для проверки теплового режима, способные перегреть процессор с недостаточным охлаждением и довести его до сбоев и даже разрушения.

Для охлаждения процессоров применяют радиаторы (Heat Sink — теплоотвод). Радиатор эффективно работает, только если обеспечивается его плотное прилегание к верхней поверхности корпуса процессора (даже тонкий воздушный зазор имеет низкую теплопроводность). Весьма эффективно использование теплопроводной мастики, которую наносят тонким слоем на корпус процессора, после чего радиатор «притирают» к процессору. Хорошие результаты дает и приклеивание радиатора к процессору двусторонней «самоклейкой» (но только специально предназначенной для этих целей, поскольку обычные «липучки» не термостойки и имеют большое тепловое сопротивление). Когда пассивного теплоотвода, обеспечиваемого только радиатором, который рассчитан на естественную циркуляцию воздуха внутри корпуса компьютера, оказывается недостаточно, применяют активные теплоотводы (Cooler). Они имеют дополнительные вентиляторы (Fan), устанавливаемые на радиатор процессора. Вентиляторы обычно являются съемными устройствами, питающимися от источника +12В через специальный переходной разъем. Некоторые процессоры имеют вентиляторы, приклеенные на заводе. Процессоры Pentium®OverDrive® (замена для 486) поставляются с установленным вентилятором, питающимся прямо от специально выделенных контактов сокета. Размеры (габаритные и установочные) вентиляторов и радиаторов для процессоров 486, Pentium и Pentium Pro различаются (чем старше процессор, тем больше радиатор и вентилятор). Для процессоров Pentium с частотой 200 МГц и выше требуется более высокий радиатор с более мощным вентилятором, чем для предыдущих моделей.

Стандарт конструктива АТХ предусматривает установку процессора прямо под блоком питания, при этом для обдува радиатора может использоваться как внутренний вентилятор блока питания, так и дополнительный внешний, устанавливаемый снаружи блока питания, и вентилятор процессора. Теоретически, все они должны работать согласованно — на обдув воздухом радиатора процессора. В противном случае их суммарная эффективность будет падать. При наличии большого радиатора на процессоре в корпусе АТХ можно обойтись и без отдельного вентилятора на процессоре.

Вентилятор как электромеханическое устройство принципиально имеет меньшую надежность (срок жизни), чем процессор. С вентиляторами могут быть связаны неприятности разной степени тяжести — от повышенного шума при

работе до отказа (остановки). Частой причиной остановки вентилятора является касание внутренних соединительных проводов (интерфейсных шлейфов дисков, подключения кнопок и индикаторов лицевой панели). Поэтому рекомендуется после сборки компьютера подвязывать провода к шасси корпуса - для сохранности, как проводов, так и вентилятора. Существуют вентиляторы с сигнализацией неисправности: они имеют датчик вращения и простенькую плату электроники, смонтированную на вентиляторе. Эта плата включается между разъемом стандартного динамика и самим динамиком. При остановке вентилятора динамик начинает пищать. Признаком наличия такого устройства является характерная «мелодия», звучащая при включении питания (её не спутать с однотональными звуками диагностики POST).

Более совершенные системы (используемые для серверов) могут иметь и датчики температуры, подающие прерывание в случае ее превышения Процессоры Pentium Pro и Pentium II имеют внутренний датчик температуры, аварийно останавливающий процессор в случае перегрева. Вентилятор картриджа Pentium II имеет датчик вращения, вырабатывающий пару импульсов за один оборот. Сигнал датчика выведен на разъем питания вентилятора, его обработка возлагается на компоненты системной платы.

Диапазон допустимых рабочих температур для различных процессоров обычного исполнения приведены в табл. 3.3. Эта температура измеряется в центре верхней поверхности корпуса процессора (не радиатора!) в установившемся рабочем режиме. Процессоры для мобильных применений обычно имеют меньшую потребляемую мощность и более высокую допустимую температуру корпуса. Существуют и специальные исполнения процессоров, допускающих расширенный температурный диапазон. Они дороже обычных и в РС применяются довольно редко.

## 12. **Внешние устройства. Организация ввода-вывода** 131 стр

## компьютерная система состоит из трех основных компонентов: центрального процессора, памяти (основной и вспомогательной) и устройств ввода-вывода (принтеров, сканеров и модемов).

## ШИНЫ

## Обычно устройство представляет собой металлический корпус с большой интегральной схемой на дне, которая называется материнской платой (политкорректности ради можно называть ее системной платой). Материнская плата содержит микросхему процессора, несколько разъемов для модулей DIMM и различные вспомогательные микросхемы. Еще на материнской плате располагаются шина (она тянется вдоль платы) и несколько разъемов для подсоединения устройств ввода-вывода.

## Каждое устройство ввода-вывода состоит из двух частей: одна объединяет большую часть электроники и называется контроллером, а другая представляет собой само устройство ввода-вывода, например дисковод. Контроллер обычно располагается на плате, которая вставляется в свободный разъем. Исключение представляют собой контроллеры устройств, являющихся неотъемлемыми составными частями компьютера (например, клавиатуры), которые иногда располагаются на материнской плате.Хотя дисплей (монитор) и нельзя назвать дополнительным устройством, соответствующий контроллер иногда располагается на встроенной плате, чтобы пользователь мог по желанию выбирать платы с графическими ускорителями или без них, устанавливать дополнительную память и т. д. Контроллер связывается с самим устройством кабелем, который соединяется с разъемом на задней стороне корпуса.

## Контроллер управляет своим устройством ввода-вывода и для этого регулирует доступ к шине. Если контроллер считывает данные из памяти или записывает их в память без участия центрального процессора, то говорят, что осуществляется прямой доступ к памяти (Direct Memory Access, DMA). Когда передача данных заканчивается, контроллер выдает прерывание, вынуждая центральный процессор приостановить работу текущей программы и начать выполнение особой процедуры. Эта процедура называется программой обработки прерываний и нужна она для того, чтобы проверить, нет ли ошибок, в случае их обнаружения произвести необходимые действия и сообщить операционной системе, что процесс ввода-вывода завершен.

## Шины PCI и PCIe

## другие компании начали производить компьютеры с несколькими шинами, одной из которых была либо прежняя шина ISA, либо шина EISA (Extended ISA — расширенная стандартная промышленная архитектура), как и ISA совместимая со старыми устройствами ввода-вывода. Что касается другой шины, то в настоящее время самой популярной моделью является шина PCI (Peripheral Component Interconnect — взаимодействие периферийных компонентов), разработанная компанией Intel, которая решила открыть всю связанную с шиной техническую информацию, чтобы сторонние производители (в том числе конкуренты компании) могли разрабатывать соответствующие устройства. Как бы быстро ни работало компьютерное оборудование, найдется много людей, которым оно покажется слишком медленным. Такая судьба постигла и шину PCI, которая была заменена шиной PCI Express (сокращенно PCIe). Многие

## современные компьютеры поддерживают обе шины, благодаря чему пользователи могут подключать новые, быстрые устройства к шине PCIe, а старые, более медленные — к шине PCI.

## Терминалы

## компьютера состоят из двух частей: клавиатуры и монитора. В мэйнфреймах эти части объединены в одно устройство и связаны с самим мэйнфреймом обычным или телефонным проводом.

## Клавиатуры

## Существуют несколько видов клавиатур. У первых компьютеров IBM PC под каждой клавишей находился переключатель, который давал ощутимую отдачу и щелкал при нажатии клавиши. Сегодня механический контакт с печатной платой при нажатии клавиш происходит лишь у самых дешевых клавиатур. У клавиатур получше между клавишами и печатной платой располагается слой эластичного материала (особого типа резины). Под каждой клавишей находится небольшой купол, который прогибается в случае нажатия клавиши. Проводящий материал, находящийся внутри купола, замыкает схему. У некоторых клавиатур под каждой клавишей находится магнит, который при нажатии клавиши проходит через катушку и таким образом вызывает электрический ток. В персональных компьютерах при нажатии клавиши происходит процедура прерывания и запускается программа обработки прерывания (эта программа является частью программного обеспечения операционной системы). Программа обработки прерывания считывает содержимое аппаратного регистра в контроллер клавиатуры, чтобы получить номер нажатой клавиши (от 1 до 102). Когда клавиша отпускается, происходит второе прерывание

## Сенсорные экраны

## Сенсорные устройства делятся на прозрачные и непрозрачные. Типичное непрозрачное сенсорное устройство — сенсорная панель (тачпад) на ноутбуке. Типичное прозрачное устройство — экран смартфона или планшетного компьютера. Мы ограничимся рассмотрением устройств второго типа, которые обычно называются сенсорными экранами. Основные разновидности сенсорных экранов — инфракрасные, резистивные и емкостные Принцип работы инфракрасных экранов основан на размещении инфракрасных передатчиков (скажем, инфракрасных светодиодов или лазеров) на левом и верхнем краях оправы, с детекторами на правом и нижнем краях. Другая старая технология изготовления сенсорных экранов — резистивная —состоит из двух слоев. Верхний гибкий слой содержит большое количество горизонтальных проводников. В находящейся под ним мембране проходят вертикальные проводники.

## Плоские мониторы

## В первых компьютерных мониторах использовались электронно- лучевые трубки (ЭЛТ), как в старых телевизорах. Они были слишком громоздкими и тяжелыми для использования портативных компьютерах, поэтому для экранов портативных компьютеров требовалась совершенно другая технология. Развитие плоских (flat-panel) мониторов позволило реализовать компактный форм-фактор, необходимый для ноутбуков, к тому же эти устройства использовали меньше энергии. В наши дни преимущества плоских экранов привели практически к полному вымиранию ЭЛТ-мониторов. Самой распространенной технологией плоских мониторов является жидкокристаллический дисплей.

## Видеопамять

## Обновление картинки на экранах ЭЛТ- и TFT-мониторов производится от 60 до 100 раз в секунду; для этого используется видеопамять, размещенная на плате контроллера дисплея. Видеопамять содержит одну или несколько битовых карт, представляющих выводимое на экран изображение. В современных дисплеях каждый пиксел представлен 3-байтным значением RGB, которое определяет интенсивность красного (Red), зеленого (Green) и синего (Blue) компонентов изображения (мощные профессиональные мониторы используют 10 и более бит на цвет). Как известно, любой цвет можно представить путем линейной суперпозиции трех упомянутых базовых цветов.

## Мыши

## Мышь — это устройство в маленьком пластиковом корпусе, располагающееся на столе рядом с клавиатурой. Если двигать мышь по столу, указатель на экране тоже будет двигаться, что дает возможность навести его на тот или иной элемент экрана. У мыши есть одна, две или три кнопки, нажатие которых дает возможность пользователям выбирать пункты меню. Существует три типа мышей: механические, оптические и оптомеханические. У мышей первого типа снизу располагаются резиновые колесики, оси которых расположены перпендикулярно друг к другу. Если мышь передвигается в вертикальном направлении, то вращается одно колесо, а если в горизонтальном, то другое. Каждое колесико приводит в действие резистор (потенциометр). Если измерить изменения сопротивления, можно узнать, на сколько провернулось колесико, и таким образом вычислить, на какое расстояние передвинулась мышь в каждом направлении. Следующий тип — оптическая мышь. У нее нет ни колес, ни шарика. Вместо этого в нижней части мыши располагаются светодиод и фотодетектор. Третий тип — оптомеханическая мышь. У нее, как и у более современной механической мыши, есть шарик, который вращает два колесика,расположенные перпендикулярно друг к другу. Колесики связаны с кодировщиками. В каждом кодировщике имеются прорези, через которые проходит свет.

## Игровые контроллеры

## Контроллер Wiimote, выпущенный в 2006 году для игровой приставки Nintendo Wii, содержит традиционные кнопки вместе с двойным датчиком перемещения. Все действия с Wiimote передаются в реальном времени игровой приставке через внутренний передатчик Bluetooth. Датчики перемещения позволяют Wiimote отслеживать перемещения в трех измерениях, а также обеспечивают точное распознавание направления при наведении на телевизор. Функциональность Kinect обеспечивается датчиком глубины в сочетании с видеокамерой. Датчик глубины вычисляет расстояние до объекта, находящегося в поле зрения Kinect. Для этого он излучает матрицу инфракрасных лазерных точек, а затем читает их отражения на инфракрасную камеру. Используя технологию распознавания образов, которая называется «структурированным освещением», Kinect может определить расстояние до объектов в своем поле зрения п искажению матрицы инфракрасных точек освещенными поверхностями.

## Принтеры

## Лазерные принтеры Это устройство сочетает хорошее качество печати, универсальность, высокую скорость работы и умеренную стоимость. В лазерных принтерах используется почти та же технология, что и в фотокопировальных устройствах. Многие компании производят устройства, совмещающие свойства копировальной машины, принтера и иногда факса.

## Цветные принтеры

## Отраженный свет используется при создании цветных фотографий и картинок в глянцевых журналах. В этом случае поглощается свет с определенной длиной волны, а остальной свет отражается. Такие изображения создаются путем субтрактивного наложения трех основных цветов: голубого (красный полностью поглощен), желтого (синий полностью поглощен) и сиреневого (зеленый полностью поглощен). Теоретически путем смешения голубых, желтых и сиреневых чернил можно получить любой цвет. Но на практике очень сложно получить такие чернила, которые полностью поглощали бы весь свет и в результате давали черный цвет. По этой причине практически во всех цветных печатающих устройствах используются чернила четырех цветов: голубого (Cyan), сиреневого (Magenta), желтого (Yellow) и черного (blacK). Такая цветовая модель называется CMYK (из слова «black» берется последняя буква, чтобы отличать его от слова «blue» в модели RGB). Мониторы, напротив, для создания цветного изображения используют испускаемый свет и наложение красного, зеленого и синего цветов.

## Струйные принтеры

## Дома удобно использовать недорогие струйные принтеры. В таком принтере подвижная печатающая головка содержит картридж с чернилами. Она двигается горизонтально над бумагой, а чернила в это время выпрыскиваются из крошечных сопел. Объем одной порции чернил приблизительно равен один пиколитр. Для наглядности уточним, что в одной капле воды может уместится около 100 миллионов таких порций.

## Телекоммуникационное оборудование

## Большинство современных компьютеров подключаются к компьютерным сетям, из которых наиболее распространен Интернет. Для доступа к подобного рода сетям требуется специальное оборудование. Модем посылает отдельные биты каждого символа через равные временные отрезки. Например, скорость 9600 бод означает, что сигнал меняется каждые 104 микросекунд. Второй модем, получающий информацию, преобразует модулированный несущий сигнал в двоичное число. Биты поступают в модем через равные промежутки времени. Если модем встречает начало символа, его часы сообщают, когда нужно начать считывать поступающие биты. Многие телевизионные компании предлагают пользователям возможность доступа в Интернет по кабельным сетям. Эта технология несколько отличается от ADSL, поэтому ее стоит рассмотреть отдельно. Во владении каждого оператора кабельного телевидения, помимо центрального офиса, есть ряд головных узлов (помещений с электронным оборудованием), рассредоточенных по территории города. К центральному офису все они подключены широкополосным или оптоволоконным кабелем.

## Цифровые фотокамеры

## При попадании на устройство CCD света устройство получает электрический заряд. Чем больше света, тем существеннее изменение заряда. Заряд считывается. Стандартная организация восходящих и нисходящих каналов в США. Технология QAM-64 (квадратурная амплитудная модуляция) допускает передачу со скоростью 6 бит/Гц, но работает только на высоких частотах. Технология QPSK (квадратурная фазовая модуляция) действует на низких частотах, зато максимальная скорость передачи составляет 2 бит/Гц аналогово-цифровым преобразователем в виде целого числа от 0 до 255 (в камерах низкой ценовой категории) или от 0 до 4095 (на цифровых однообъективных зеркальных фотоаппаратах).

## Коды символовASC11

## Один из двух широко распространенных кодов называется ASCII (American Standard Code for Information Interchange — американский стандартный код для обмена информацией). Каждый ASCII-символ содержит 7 бит, таким образом, всего можно закодировать 128 символов. Коды от 0 до 1F (в шестнадцатеричной системе счисления) соответствуют управляющим символам, которые не печатаются . Коды от 0 до 1F (в шестнадцатеричной записи) соответствуют управляющим символам, которые не выводятся на печать. Коды от 128 до 255 не входят в кодировку ASCII; на IBM PC за ними были закреплены специальные символы (улыбающиеся лица и т. д.), которые до сих пор поддерживаются большинством компьютеров.

## Unicode

## Группа компьютерных компаний разрешила эту проблему, создав новую систему кодирования под названием Unicode, и объявила эту систему международным стандартом (IS 10646). Unicode поддерживается некоторыми языками программирования (например, Java), некоторыми операционными системами (например, Windows NT) и многими приложениями. Вероятно, эта система будет распространяться по всему миру. Основная идея Unicode — приписать каждому символу единственное постоянное 16- разрядное значение, которое называется кодовым пунктом. Многобайтные символы и символы-заменители не используются. Поскольку каждый символ состоит из 16 бит, писать программное обеспечение гораздо проще.

## UTF-8

## Стандарт Unicode был лучше ASCII, но со временем в нем тоже возникла нехватка кодовых пунктов. Кроме того, он требовал 16 бит для представления «чистого» ASCII-текста, что было расточительно. Для решения этих проблем была разработана новая схема кодирования UTF-8 UCS Transformation Format. Сокращение UCS в названии означает «Universal Character Set» — по сути это Unicode. Коды UTF-8 имеют переменную длину от 1 до 4 байт, и позволяют кодировать до двух миллиардов символов. Этот способ кодировки сейчас доминирует в World Wide Web.

## ВЫВОД: Устройства ввода-вывода используются для передачи информации в компьютер из компьютера. Они связаны с процессором и памятью одной или несколькими шинами. В качестве примеров можно назвать терминалы, мыши, принтеры и модемы. Большинство устройств ввода-вывода используют код ASCII, хотя Unicode уже стремительно распространяется по всему миру, а UTF-8 получает все большее распространение по мере того, как компьютерная отрасль все больше ориентируется на Web

## 

## 13. Устройство и назначение системы BIOS

BIOS (Basic Input/Output System – базовая система ввода-вывода) – это программа для первоначального запуска компьютера, настройки оборудования и обеспечения функций ввода-вывода.

BIOS записывается в микросхему flash-памяти, которая расположена на системной плате. Изначально основным назначением BIOS было обслуживание устройств ввода-вывода (клавиатуры, экрана и дисковых накопителей), поэтому ее и назвали «базовая система ввода-вывода». В современных компьютерах BIOS выполняет несколько функций:

- запуск компьютера и процедуру самотестирования (Power-On Self Test, POST). Программа, расположенная в микросхеме BIOS, загружается первой после включения питания компьютера. Она детектирует и проверяет установленное оборудование, настраивает его и готовит к работе. Если обнаруживается неисправность оборудования, процедура POST останавливается с выводом соответствующего сообщения илизвукового сигнала;

- настройку параметров системы с помощью программы BIOS Setup. Во время процедуры POST оборудование настраивается в соответствии с параметрами, хранящимися в специальной CMOS-памяти. Изменяя эти параметры, пользователи могут конфигурировать отдельные устройства и систему в целом по своему усмотрению. Редактируются они в специальной программе, которую называют BIOS Setup или CMOS Setup. Настройке системы с помощью программы BIOS Setup будет посвящена большая часть этой книги;

- поддержку функций ввода-вывода с помощью программных прерываний BIOS. В составе системной BIOS есть встроенные функции для работы с клавиатурой, видеоадаптером, дисководами, жестки ми дисками, портами ввода-вывода и др. Эти функции использовались в операционных системах, подобных MS-DOS, и почти не применяются в современных версиях Windows.

Во всех современных компьютерах BIOS хранится в микросхеме на основе flash-памяти (Flash Memory). Такая микросхема может быть перезаписана с помощью специальных программ прямо на компьютере. Запись новой версии BIOS обычно называется перепрошивкой. Эта операция может понадобиться, чтобы добавить в код BIOS новые функции, исправить ошибки или заменить поврежденные версии.

Тест начального включения POST

По включении питания, аппаратному сбросу от кнопки RESET или нажатии комбинации клавиш Ctrl+Alt+Del процессор переходит к исполнению кода начального самотестирования POST (Power-On Self Test), хранящегося в микросхеме BIOS. POST выполняет тестирование процессора, памяти и системных средств ввода/вывода, а также конфигурирование всех программно-управляемых аппаратных средств системной платы. Часть конфигурирования выполняется однозначно, часть управляется джамперами системной платы, но ряд параметров позволяет или даже требует конфигурирования по желанию пользователя. Для этих целей служит утилита Setup, встроенная в код BIOS. После тестирования и конфигурирования (включающего настройку устройств РпР), POST инициализирует загрузку операционной системы. При прохождении каждой секции POST записывает ее код (номер) в диагностический регистр. Этот регистр физически располагается на специальной диагностической плате, устанавливаемой в слот системной шины. По индикаторам платы можно определить, на какой секции остановился POST, и определить причину неисправности. Однако для использования такой диагностики необходима, во-первых, сама плата-индикатор, и во-вторых, «словарь» неисправностей - таблица, специфическая для версии BIOS и

системной платы. Во время выполнения POST может выдавать диагностические сообщения в виде последовательности коротких и длинных звуковых сигналов, а после успешной инициализации графического адаптера краткие текстовые сообщения выводятся на экран монитора. Обычная последовательность шагов POST:

· Тестирование регистров процессора.

· Проверка контрольной суммы ROM BIOS.

· Проверка и инициализация таймера 8253/8254, портов 8255. После этого шага доступна звуковая диагностика.

· Проверка и инициализация контроллеров DMA 8237.

· Проверка регенерации памяти.

· Тестирование 64 Кбайт нижней памяти.

· Загрузка векторов прерывания и стека в нижнюю область памяти.

Инициализация видеоконтроллера — на экране появляется заставка Video BIOS, обычно с указанием модели видеокарты и объемом установленной видеопамяти.

После успеха этого шага изображение на экране сменяется заставкой системной BIOS со счетчиком объема тестируемой динамической памяти. Теперь диагностические сообщения выводятся на экран. POST продолжает работу, выполняя следующие шаги:

· Тестирование полного объема ОЗУ.

· Тестирование клавиатуры.

· Тестирование CMOS-памяти и часов.

· Инициализация СОМ и LPT портов.

· Инициализация и тест контроллера НГМД.

· Вызов Bootstrap (INT 19h) — загрузка операционной системы, при невозможности - попытка запуска ROM Basic (Int 18h), при неудаче — останов процессора с сообщением «System Halted» система остановлена). При загрузке системы в случае готовности дисковода А: в память загружается первый сектор диска и ему передается управление; при неготовности А: загружается Master Boot диска С: и ему передается управление. Master Boot загружает Boot Sector активного раздела в память и передает управление на его начальный адрес. Последовательность загрузки может изменяться в BIOS Setup.

## 14. **Распределение памяти . Организация прерываний**

**Прерывания и их организация.**Прерывания — это изменения в потоке управления, вызванные не самой программой, а чем-либо другим. Обычно прерывания связаны с процессом вводавывода. Например, программа может дать команду диску начать передачу информации и инициировать прерывание, как только передача данных завершится. Как и в случае исключений, при прерываниях работа программы останавливается, и управление передается программе обработки прерываний (Interrupt Service Routine, ISR), или обработчику прерываний, который выполняет определенные действия. После завершения этих действий обработчик прерываний передает управление прерванной программе. Она должна заново начать прерванный процесс в том же самом состоянии, в котором находилась в момент прерывания. Это значит, что прежнее состояние всех внутренних регистров (то есть состояние, которое было до прерывания) должно быть восстановлено. Различие между исключениями и прерываниями заключается в том, что исключения синхронны по отношению к программе, а прерывания асинхронны. Если многократно перезапускать программу с одними и теми же входными данными, исключения каждый раз будут происходить в одних и тех же местах программы, а прерывания — нет (в нашем примере с диском прерывание произойдет только тогда, когда диск завершит передачу данных, а не когда этого потребует программа). Причина воспроизводимости исключений и невоспроизводимости прерываний состоит в том, что первые вызываются программой непосредственно, а вторые — опосредовано. Чтобы понять, как происходят прерывания, рассмотрим обычный пример: компьютеру нужно вывести на терминал строку символов. Программа сначала собирает в буфер все символы, предназначенные для вывода на экран, инициализирует глобальную переменную ptr, которая должна указывать на начало буфера, и делает вторую глобальную переменную count равной числу символов, выводимых на экран. Затем программа проверяет, готов ли терминал, и если готов, выводит на экран первый символ (например, используя регистры, показанные на рис. 5.19). Начав процесс ввода-вывода, центральный процессор освобождается и может запустить другую программу или сделать что-либо еще.Через некоторое время символ отображается на экране. После этого может быть инициировано прерывание. Ниже перечислены основные шаги (в упрощенной форме). Действия аппаратного обеспечения

1. Контроллер устройства нагружает линию прерывания на системной шине.

## 2. Когда центральный процессор оказывается готовым к обработке прерывания, он выставляет на шине символ подтверждения прерывания.

## 3. Когда контроллер устройства обнаруживает, что сигнал прерывания подтвержден, он помещает небольшое целое число на информационные линии, чтобы «представиться» (то есть показать, что за устройство является источником прерывания). Это число называется вектором прерываний1 .

## 4. Центральный процессор считывает вектор прерывания с шины и временно его сохраняет. 5. Центральный процессор помещает в стек счетчик команд и слово состояния программы.

## 6. Центральный процессор определяет местонахождение нового счетчика команд, используя вектор прерывания в качестве индекса в таблице в нижней части памяти. Если, например, размер счетчика команд составляет 4 байта, тогда вектор прерываний n соответствует адресу 4n. Новый счетчик команд указывает на начало программы обработки прерываний для устройства, ставшего источником прерывания. Часто помимо этого загружается или изменяется слово состояния программы (например, чтобы блокировать дальнейшие прерывания). Дальнейшие действия выполняются программно:

## 7. Программа обработки прерываний сохраняет все нужные ей регистры таким образом, чтобы их можно было восстановить позднее. Их можно сохранить в стеке или в системной таблице.

## 8. Каждый вектор прерывания используется всеми устройствами данного типа совместно, поэтому в данный момент еще не известно, какой терминал вызвал прерывание. Номер терминала можно узнать, считав значение какогонибудь регистра.

## 9. После этого можно считывать любую другую информацию прерывания, например коды состояния.

## 10. Если происходит ошибка ввода-вывода, на этом этапе ее нужно обработать.

## 11. Глобальные переменные ptr и count обновляются. Первая увеличивается на 1, чтобы показывать на следующий байт, а вторая уменьшается на 1, чтобы указать, что осталось вывести на 1 байт меньше. Если count все еще больше 0, значит, еще не все символы выведены на экран. Тот символ, на который в данный момент указывает ptr, копируется в выходной буферный регистр.

## 12. В случае необходимости выдается специальный код, который сообщает устройству или контроллеру прерывания, что прерывание обработано.

## 13. Восстанавливаются все сохраненные регистры.

## 14. Выполняется команда выхода из прерывания, возвращающая центральный процессор в то состояние, в котором он находился до прерывания. После этого компьютер продолжает работу с того места, в котором ее приостановил. С прерываниями связано важное понятие прозрачности. Когда происходит прерывание, могут производиться разнообразные действия и запускаться разного рода программы, но когда все заканчивается, компьютер должен вернуться точно в то же состояние, в котором он находился до прерывания. Программа обработки прерываний, обладающая таким свойством, называется прозрачной. Если компьютер имеет только одно устройство ввода-вывода, тогда прерывания работают именно так, как мы только что описали. Однако большой компьютер может содержать много устройств ввода-вывода, причем несколько устройств могут работать одновременно, возможно, у разных пользователей. Существует некоторая вероятность, что во время работы обработчика прерывания другое устройство ввода-вывода тоже попытается вызвать свое прерывание. Здесь существует два подхода. Первый — для всех программ обработки прерываний сначала (даже до сохранения регистров) предотвратить последующие прерывания — в этом случае прерывания будут происходить строго поочередно. Однако это может привести к проблемам с устройствами, которые не могут долго простаивать. Например, на линию связи, поддерживающую скорость передачи 9600 бит в секунду, символы поступают каждые 1042 микросекунды. Если первый символ окажется необработанным, когда поступит второй, данные будут потеряны. Если компьютер имеет подобные устройства ввода-вывода, то лучше всего приписать каждому устройству определенный приоритет, высокий — для более критичных, низкий — для менее критичных устройств. Центральный процессор тоже должен иметь приоритеты, которые определяются по одному из полей слова состояния программы. Если устройство с приоритетом n вызывает прерывание, программа обработки прерывания тоже должна работать с приоритетом n. Если программа обработки прерывания выполняется с приоритетом n, любая попытка обработать прерывание от другого устройства с более низким приоритетом будет игнорироваться, пока программа обработки прерывания не завершится и центральный процессор не начнет выполнение программы более низкого приоритета. В то же время прерывания от устройств с более высоким приоритетом должны обрабатываться без задержек. Поскольку сами программы обработки прерываний могут прерываться, единственно возможный способ точно управлять ситуацией — сделать так, чтобы все прерывания были прозрачными. Рассмотрим простой пример с несколькими прерываниями. Пусть компьютер имеет три устройства ввода-вывода: принтер, диск и линию RS232 с приоритетами 2, 4 и 5 соответственно. Изначально (t = 0, где t — время) работает пользовательская программа. При t = 10 принтер неожиданно инициирует прерывание. Запускается программа обработки прерывания (ISR) от принтера, как показано на рис. 5.27. При t = 15 прерывания требует линия RS232. Так как линия RS232 имеет более высокий приоритет (5), чем принтер (2), инициируется обработка этого прерывания. Состояние машины, при котором работает ISR принтера, сохраняется в стеке, и начинается выполнение программы обработки прерывания линии RS232. Немного позже, при t = 20, диск завершает свою работу и сигнализирует об этом прерыванием. Однако его приоритет (4) ниже, чем приоритет работающей в данный момент программы обработки прерываний (5), поэтому центральный процессор не подтверждает прием сигнала прерывания, и диск вынужден ждать. При t = 25 заканчивает работать ISR линии RS232, и машина возвращается в то состояние, в котором она находилась до прерывания от линии RS232, то есть в состояние, соответствующее работе ISR принтера с приоритетом 2. Как только центральный процессор переключается на приоритет 2, еще до выполнения первой команды, диск с приоритетом 4 совершает прерывание, и запускается ISR диска. После ее завершения снова продолжается программа обработки прерываний от принтера. Наконец, при t = 40 все программы обработки прерываний завершаются, и выполнение пользовательской программы начинается с того места, на котором она прервалась. Со времен процессора 8088 все процессоры Intel имеют два уровня (приоритета) прерываний: маскируемые и немаскируемые прерывания. Немаскируемые прерывания обычно используются только для сообщения об очень серьезных ситуациях, например об ошибках четности в памяти. Для всех устройств вводавывода имеется единственное маскируемое прерывание. Когда устройство ввода-вывода требует прерывания, центральный процессор использует вектор прерывания при индексировании таблицы из 256 элементов, чтобы найти адрес программы обработки прерываний. Элементы таблицы представляют собой 8-байтные дескрипторы сегмента. Таблица может начинаться в любом месте памяти. Глобальный регистр указывает на ее начало. При наличии только одного уровня прерываний у центрального процессора нет возможности сделать так, чтобы высокоприоритетное устройство прерывало работу среднеприоритетной программы обработки прерываний, пока этому мешает низкоприоритетное устройство. Для решения проблемы центральные процессоры Intel обычно используют внешний контроллер прерываний (например, 8259A). При первом прерывании (например, с приоритетом n) работа процессора приостанавливается. Если после этого происходит еще одно прерывание с более высоким приоритетом, контроллер прерывания инициирует прерывание во второй раз. Если же второе прерывание обладает более низким приоритетом, оно не инициируется до окончания первого. Чтобы эта система работала, контроллер прерываний должен каким-либо образом узнавать о завершении текущей программы обработки прерываний. Поэтому когда полностью завершается обработка текущего прерывания, центральный процессор должен посылать контроллеру прерываний специальную команду

## 

## 15. **Конфигурирование компьютера - BIOS Setup**

Все современные компьютеры имеют утилиту Setup, встроенную в BIOS. Утилита BIOS Setup имеет интерфейс в виде меню, иногда даже оконный с поддержкой мыши. Оконный интерфейс в данном случае раздражает, поскольку вместо быстрого входа в текстовое меню компьютер долго ищет подключена ли мышь, после чего выводит окна в режиме графики низкого разрешения (дань совместимости). При этом никаких принципиально новых возможностей (в сравнении с текстовым режимом и управлением от клавиатуры) не появляется.

Для входа в setup во время выполнения POST появляется предложение нажать клавишу Del. Иногда для этого используется комбинация Ctrl+Alt+Esc, Esc, Ctrl+Esc, бывают и экзотические варианты (нажать клавишу F12 в те секунды, когда в правом верхнем углу экрана виден прямоугольник). Некоторые версии BIOS позволяют войти в setup по комбинации Ctrl+Alt+Esc в любой момент работы компьютера. Предложение (и способ - нажатие F1 или F2) входа в setup появляется, если POST найдет ошибку оборудования, которая может быть устранена с помощью Setup. Удержание клавиши Ins во время POST в ряде версий BIOS позволяют установить параметры по умолчанию, отменяя все "ускорители". Это помогает восстановить работоспособность после чрезмерно агрессивных попыток "разогнать" компьютер.

Меню утилиты Setup, способы перемещения по пунктам и выбора параметров зависят от склонностей производителя и версии BIOS, но они понятны с краткого объяснения на экране. Нажатие F1 или Alt+H вызывает короткую контекстную справку, конечно связанную с навигацией. Смысловых объяснений значения параметров она не дает. Состав управляемых параметров, детальная и гибкость управления варьируется от предельно подробных, в которых может запутаться и опытный пользователь, к предельно коротких. Что лучше - дело вкуса. Ниже приведем объяснения распространенных установок. В конкретной версии они представлены, конечно же, только выборочно. Некоторые настройки могут называться и не совсем так, как указано в таблицах, но быть созвучными (в английском варианте). За период развития PC некоторые термины приобрели новое значение - если раньше под типом микросхем памяти (DRAM Type) имели в виду объем микросхем (64К, 25К, 1М), то теперь это FPM и EDO, BEDO и SDRAM. В связи с этим возможно двоякое толкование некоторых параметров, но нельзя объять необъятное и перечислить все существующие в настоящее время параметры настройки. Возможные пункты главного меню Setup приведены в табл. 3.3.

Опция "Auto Configuration with BIOS Defaults" позволяет установить набор параметров, что обеспечивает нормальную работу системной платы. При этом не затрагивается дата, время, параметры гибких и жестких дисков. Это является начальной точкой для оптимизации установок, на которой можно и остановиться.

Опция "Auto Configuration with Power-on Defaults" - устанавливает самые консервативные значения параметров: кэширование запрещено на обоих уровнях, временные диаграммы сами растянуты и т.п. Если системная плата не работает и с такими установками, необходимо проверить ее аппаратную конфигурацию - установку джамперов, съемных элементов (процессор, память, кэш и т. п.). Если в setup не войти, те же значения параметров можно получить, удерживая клавишу Del (иногда Ins) во время включения компьютера, или для этих целей имеется специальный переключатель (джампер) на системной плате. Способ спасения зависит от версии BIOS и модели системной платы.

Выбранные установки сохраняются при выходе из Setup (по желанию пользователя) и начинают действовать с момента начала следующего выполнения POST. Таким образом, если нет уверенности в правильности выбранных настроек, можно выйти из Setup без сохранения новых значений.

Выбранные значения установок рекомендуется сохранить на бумаге. К сожалению, функция печати экрана клавиши Printscreen в Setup работает не всегда (до инициализации во время загрузки LPT-порт может содержать низкий уровень на выходе 1МГГ#, что не позволяет принтеру печатать). Но затраты времени даже на рукописный сохранения параметров помогут впоследствии сэкономить время, силы и нервы в критической ситуации, поскольку некоторые установки могут приводить к полной потере работоспособности компьютера (к счастью, временной - до исправления значений).

Неудачные параметры конфигурации (или забытый пароль) при невозможности входа в setup можно сбросить отключением питания CMOS (замыканием контактов 3, 4 разъема аккумулятора). В некоторых системах пароль сбрасывается только специальной перемычкой. Однако в CMOS хранятся совсем не все параметры - часть их содержится в NVRAM, которую отключением батареи изменить (очистить) невозможно по определению (это действительно энергонезависимая память). Хорошо, если на системной плате имеется специальный переключатель для сброса NVRAM (или, хотя бы, ESCD). Такой переключатель срабатывает, если в положении "очистки" на системную плату будет подано питание (вместе с сигналом аппаратного сброса). После обнуления включают плату с нормальным положением переключателя. Изредка встречается опция Setup, предназначенная для сброса NVRAM. Если явных способов очистки нет, то при необходимости остается воспользоваться утилитой перепрограмирования флэш-BIOS (NVRAM обычно является областью микросхемы-носителя флэш-BIOS). Однако для этого необходимо иметь файл-образ BIOS и утилиту программирования. Записью информации в NVRAM иногда грешит Windows 95 при установке. Это может привести к потере работоспособности отдельных узлов и даже платы в целом, не устраненной никакими настройками Setup.

Группа параметров, которые задают "тонкие" настройки (режимы и временные диаграммы), требует более глубокого знания функционирования подсистем компьютера. Общие принципы настройки такие: чем выше частоты, меньшие коэффициенты деления и количества тактов ожидания (Wait States), тем выше производительность подсистемы компьютера и, что затрагивает, в целом, если подсистема используется интенсивно. Пределы ускорения определяются быстродействием и количеством применяемых компонентов и могут быть обнаружены эмпирически. Однако возможны побочные эффекты, когда "разгон" одной подсистемы приводит к неработоспособности другой, на первый взгляд не сильно с ней связанной. Много групп параметров имеют общую опцию автоконфигурации (Automatic Configuration). Разрешение автоконфигурации - типовой установки таких параметров, как коэффициенты деления частоты, количества тактов ожидания и т. п., - позволяет установить если и не оптимальное, то в большинстве случаев вполне нормально работающую конфигурацию. Запрет позволяет установить эти параметры вручную (давая пользователю дополнительную возможность ошибиться).

На современной системной плате всегда расположен высокоэффективный контроллер IDE, параметрами которого управляют с Setup (табл. 3.6). Включение его прогрессивных режимов еще не означает повышения скорости обмена с дисками - для этого необходима еще и программная поддержка корректными драйверами со стороны ОС, и поддержка этих режимов собственно накопителями.

## 16. **Режимы работы ЭВМ**

Однопрограммные режимы работы появились первыми. При их реализации все основные ресурсы ЭВМ полностью отдаются в монопольное владение пользователя. Однопрограммный режим может иметь модификации: однопрограммный режим непосредственного доступа и однопрограммный режим косвенного доступа.

В режиме непосредственного доступа пользователь получает ЭВМ в полное распоряжение: он сам готовит ЭВМ к работе, загружает задания, инициирует их, наблюдает за ходом решения и выводом результатов

В режиме косвенного доступа пользователь не имеет прямого контакта с ЭВМ. Этот режим предназначался обеспечить более полную загрузку процессора за счет сокращения непроизводительных его простоев

Многопрограммный (многопользовательский) режим работы ЭВМ позволяет одновременно обслуживать несколько программ пользователей. Реализация режима требует соблюдения следующих непременных условий:

• независимость подготовки заданий пользователями;

• разделение ресурсов ЭВМ в пространстве и во времени;

• автоматическое управление вычислениями.

Режим классического мультипрограммирования, или пакетной обработки, применительно к однопроцессорным ЭВМ является основой для построения всех других видов многопрограммной работы. Режим имеет целью обеспечить минимальное время обработки пакета заданий и максимально загрузить процессор.

Режим разделения времени является более развитой формой многопрограммной работы ЭВМ. В этом режиме, обычно совмещенным с фоновым режимом классического мультипрограммирования, отдельные наиболее приоритетные программы пользователей выделяются в одну или несколько групп.

режим реального времени. Этот режим имеет специфические особенности:

• поток заявок от абонентов носит, как правило, случайный, непредсказуемый характер;

• потери поступающих на вход ЭВМ заявок и данных к ним не допускаются, поскольку их не всегда можно восстановить;

• время реакции ЭВМ на внешние воздействия, а также время выдачи результатов i-и задачи должно удовлетворять жестким ограничениям вида

Многозадачный и многопоточный режимы Windows. Операционная среда Windows 2000 и ее предшественница Windows NT поддерживают так называемые многозадачные и многопоточные режимы работы.

Многозадачный режим предполагает, что каждый из процессов (отдельных запущенных программ), активизированных в среде Windows, требует определенных ресурсов.

## 17. **Структура Центрального процессора**

Центральный процессор — это мозг компьютера. Его задача — выполнять программы, находящиеся в основной памяти. Для этого он вызывает команды из памяти, определяет их тип, а затем выполняет одну за другой. Компоненты соединены шиной, представляющей собой набор параллельно связанных проводов для передачи адресов, данных и управляющих сигналов. Шины могут быть внешними (связывающими процессор с памятью и устройствами ввода-вывода) и внутренними. Современный компьютер использует несколько шин.

Процессор состоит из нескольких частей. Блок управления отвечает за вызов команд из памяти и определение их типа. Арифметико-логическое устройство (АЛУ) выполняет арифметические операции (например, сложение) и логические операции (например, логическое И). Внутри центрального процессора находится быстрая память небольшого объема для хранения промежуточных результатов и некоторых команд управления. Эта память состоит из нескольких регистров, каждый из которых выполняет определенную функцию. Обычно размер всех регистров одинаков. Каждый регистр содержит одно число в диапазоне, верхняя граница которого зависит от размера регистра. Операции чтения и записи с регистрами выполняются очень быстро, поскольку они находятся внутри центрального процессора. Самый важный регистр — счетчик команд, который указывает, какую команду нужно выполнять следующей. Название «счетчик команд» выбрано неудачно, поскольку он ничего не считает, но этот термин употребляется повсеместно1 . Еще есть регистр команд, в котором находится выполняемая в данный момент команда. У большинства компьютеров имеются и другие регистры, одни из них многофункциональны, другие служат лишь какие-либо конкретным целям. Третьи регистры используются операционной системой для управления компьютером.

Устройство ЦП.

Тракт данных состоит из регистров (обычно от 1 до 32), арифметико-логического устройства (АЛУ) и нескольких соединительных шин. Содержимое регистров поступает во входные регистры АЛУ, которые на рис. 2.2 обозначены буквами A и B. В них находятся входные данные АЛУ, пока АЛУ производит вычисления. Тракт данных — важная составная часть всех компьютеров. АЛУ выполняет сложение, вычитание и другие простые операции над входными данными и помещает результат в выходной регистр. Содержимое этого выходного регистра может записываться обратно в один из регистров или сохраняться в памяти, если это необходимо. Не во всех архитектурах есть регистры A, B и выходные регистры. Большинство команд можно разделить на две группы: команды типа регистр-память и типа регистр-регистр. Команды первого типа вызывают слова из памяти, помещают их в регистры, где они используются в качестве входных данных АЛУ (слова — это такие элементы данных, которые перемещаются между памятью и регистрами). Словом может быть целое число. Другие команды этого типа помещают регистры обратно в память. Команды второго типа вызывают два операнда из регистров, помещают их во входные регистры АЛУ, выполняют над ними какую-нибудь арифметическую или логическую операцию и переносят результат обратно в один из регистров. Этот процесс называется циклом тракта данных. В какой-то степени он определяет, что может делать машина. Современные компьютеры оснащаются несколькими АЛУ, работающими параллельно и специализирующимися на разных функциях. Чем быстрее происходит цикл тракта данных, тем быстрее компьютер работает.

## 18. **Процессоры AMD и Cyrix**

Фирма Intel много лет занимала лидирующие позиции по производству процессоров для РС. Ее доминированию на этом рынке с 1989 г., когда появился процессор 486, стали серьезно угрожать две компании - **Cyrix** и Advanced Micro Devices (**AMD**).

Компания AMD давно выпускала процессоры для РС, начиная с процессора 8088, который применялся в первом IBM PC, и кончая процессором седьмого поколения Athlon. Утверждение о том, что процессор Athlon представляет собой первый случай в истории семейства x86, когда фирма Intel уступила лидерство другой компании, не совсем точное. Еще десять лет назад процессор 386DX-40 компании AMD превосходил процессор Intel 486SX по производительности и стоимости.

В самом начале 90-х годов прошлого века компании AMD и Cyrix выпустили свои версии процессора 486DX, но их процессоры стали более известными с появления их клонов процессора 486DX2, один из которых копировал процессор 486DX2-66 (выпущенный фирмой Intel в 1992 г.), а второй имел более высокую внутреннюю частоту 80 МГц. Процессор 486DX2-80 работал с системной шиной 40 МГц и в отличие от процессоров Intel DX2 (которые работали с питанием +5 В) имел напряжение питания +3.3 В. Вскоре после этого компании AMD и Cyrix выпустили версии с утроением частоты своих процессоров 40 МГц 486, которые имели внутреннюю частоту синхронизации 120 МГц. Обе компании AMD и Cyrix ввели средства управления мощностью в процессоры с удвоением частоты, а фирма Intel последовала их примеру в процессоре DX4, который появился примерно через два года.

Несмотря на то, что фирма Intel прекратила выпуск процессора 486 после появления модели DX4-100, компании AMD и Cyrix продолжили эту линию. В 1995 г. компания AMD объявила процессор 5x86 с учетверением частоты 5x86, который фактически был процессором 486DX с внутренней частотой 133 МГц (частота системной шины составляла 33 МГц). Компания считала этот процессор сравнимым по производительности с новым процессором Pentium/75 фирмы Intel и назвала его 5x86-75. Однако во всех отношениях это был процессор 486DX, включая добавление L1-кэша емкостью 16 КБ, который компания Intel ввела в процессор DX4. Компания Cyrix последовала этому примеру своим процессором 5x86, который назывался M1sc, но этот процессор во многом отличался от процессора компании AMD. Фактически процессор M1sc обладал некоторыми возможностями процессора Pentium, хотя он и предназначался для материнских плат 486. Работая на частоте 100 МГц и 120 МГц, этот процессор имел 64-битовую шину данных, 6-ступенчатый конвейер (в отличие от 5-ступенчатого конвейера процессора DX4) и использовал технологию предсказания перехода для повышения скорости выполнения команд. Важно отметить, однако, что процессор Cyrix 5x86 появился после выпуска процессора Pentium, поэтому указанные возможности были полезнее для модернизации систем с процессорами 486, а не производства новых систем.

В годы после появления процессора Pentium разработки обоих производителей ориентировались на рынок массовых дешевых РС. Если компания Intel сконцентрировала теперь свои усилия на конструкции Slot 1 и Slot 2, целью обоих ее конкурентов стало достижение производительности новых разработок фирмы Intel по мере их появления без перехода к новым технологиям интерфейса процессора. В результате срок службы форм-фактора Socket 7 был значительно продлен. Производители материнских плат и чипсетов, кооперирующиеся с конкурентами фирмы Intel, позволили системам на базе Socket 7 предложить такие новинки, как **передняя шина** (FrontSide Bus - FSB) на частоту 100 МГц и поддержку **ускоренного графического порта** (Accelerated Graphics Port - AGP).

В середине 1999 г. произошли некоторые новые события, которые оказали заметное влияние на рынок процессоров в последующие годы. В августе компания Cyrix оставила рынок настольных РС, когда компания National Semiconductor продала права на процессоры x86 тайваньской компании VIA Technologies, которая производила чипсеты. Высоко интегрированные процессоры MediaGX остались за компанией National Semiconductor как часть нового семейства Geode **систем на кристалле** (system-on-a-chip), которое компания разрабатывала для рынка потребительских изделий.

Буквально через несколько дней компания VIA объявила о своем намерении приобрести компанию Centaur Technology, которая разрабатывала и производила процессоры WinChip семейства х86. Было неясно, связаны ли эти действия с намерением компании VIA стать серьезным конкурентом на рынке процессоров или с желанием конкурировать с компанией National Semiconductor на рынке систем на кристалле. До настоящего времени у производителей чипсетов нет технологии для семейства x86, которая позволила бы им выпускать дешевые чипсеты с повышенными уровнями функциональности на одном кристалле.

## 

## 

## 

## **19. История развития микропроцессоров. Процессоры Pentium.**

Введение в архитектуру x86 В 1968 году Роберт Нойс (Robert Noyce), изобретатель кремниевой интегральной схемы, Гордон Мур (Gordon Moore), автор известного закона Мура, и Артур Рок (Arthur Rock), венчурный капиталист из Сан-Франциско, основали корпорацию Intel для производства компьютерных микросхем. За первый год своего существования корпорация продала микросхем всего на 3000 долларов, но потом объем продаж заметно вырос (в настоящее время Intel является крупнейшим мировым производителем процессоров). В конце 60-х годов калькуляторы представляли собой большие электромеханические машины размером с современный лазерный принтер и весили около 20 кг. В сентябре 1969 года японская компания Busicom обратилась к корпорации Intel с просьбой выпустить 12 несерийных микросхем для электронной вычислительной машины. Инженер компании Intel Тед Хофф (Ted Hoff ), назначенный в качестве исполнителя этого проекта, решил, что можно поместить 4-разрядный универсальный процессор на одну микросхему, которая будет выполнять те же функции и при этом окажется проще и дешевле. Так в 1970 году появился первый процессор на одной микросхеме — 4004 на 2300 транзисторах [Faggin et al., 1996]. Заметим, что ни сотрудники Intel, ни сотрудники Busicom не имели ни малейшего понятия, какое грандиозное открытие они совершили. Когда компания Intel решила, что стоит попробовать использовать процессор 4004 в других разработках, она предложила купить все права на новую микросхему у компании Busicom за 60 000 долларов, то есть за сумму, которую Busicom заплатила Intel за разработку этой микросхемы. Фирма Busicom сразу приняла предложение Intel, и компания Intel начала работу над 8-разрядной версией микросхемы, 8008, вы60 Глава 1. Введение пущенной в 1972 году.

Поскольку никто не ожидал большого спроса на микросхему 8008, она была выпущена достаточно ограниченным тиражом. Ко всеобщему удивлению, новая микросхема вызвала большой интерес, поэтому компания Intel начала разработку еще одного процессора, в котором предел в 16 Кбайт памяти (как у процессора 8008), навязываемый количеством внешних выводов микросхемы, был преодолен. Так появился небольшой универсальный процессор 8080, выпущенный в 1974 году. Как и PDP-8, он произвел революцию на компьютерном рынке и сразу стал массовым продуктом. Разница лишь в масштабах: компания DEC продала тысячи PDP-8, а Intel — миллионы процессоров 8080. В 1978 году появился процессор 8086, 16-разрядный процессор на одной микросхеме. Процессор 8086 был во многом похож на 8080, но не был полностью совместим с ним. Затем появился процессор 8088 с такой же архитектурой, как у 8086. Он исполнял те же программы, что и 8086, но вместо 16-разрядной шины у него была 8-разрядная, из-за чего процессор работал медленнее, но стоил дешевле, чем 80861 . Когда компания IBM выбрала процессор 8088 для IBM PC, эта микросхема быстро превратилась в промышленный стандарт в области персональных компьютеров. Ни 8088, ни 8086 не могли адресовать память объемом более 1 Мбайт. К началу 80-х годов это стало серьезной проблемой, поэтому компания Intel разработала модель 80286, совместимую с 8086. Основной набор команд остался в сущности таким же, как у процессоров 8086 и 8088, но организация памяти была несколько иной — и довольно неудобной из-за требования совместимости с предыдущими микросхемами и могла работать по-прежнему. Процессор 80286 использовался в IBM PC/AT и в моделях PS/2. Он, как и 8088, пользовался большим спросом (главным образом потому, что покупатели рассматривали его как более быстрый вариант модели 8088). Следующим шагом был 32-разрядный процессор 80386, выпущенный в 1985 году. Как и 80286, он был более или менее совместим со

всеми старыми версиями. Совместимость такого рода оказывалась благом для тех, кто пользовался старым программным обеспечением, и некоторым неудобством для тех, кто предпочитал современную архитектуру, не обремененную ошибками и технологиями прошлого. Через четыре года появился процессор 80486. Он работал быстрее, чем 80386, мог исполнять операции с плавающей точкой и имел кэш-память объемом 8 Кбайт. Кэш-память позволяет держать наиболее часто используемые слова внутри центрального процессора и избегать (медленных) обращений к основной памяти. Процессор 80486 содержал встроенную поддержку мультипроцессорного режима, что давало производителям возможность конструировать системы с несколькими процессорами.

Впоследствии в линейку Pentium были введены дополнительные команды, известные под общим названием MMX (MultiMedia eXtension — мультимедийное расширение). Они были предназначены для ускорения вычислительных операций, связанных с обработкой звуковых и видеоданных, что позволило отказаться от специальных мультимедийных сопроцессоров. Когда появилось следующее поколение компьютеров, те, кто рассчитывал на название Sexium (sex по латыни — шесть), были разочарованы. Название Pentium стало так хорошо известно, что его решили оставить, и новую микросхему назвали Pentium Pro. Несмотря на столь незначительное изменение названия, этот процессор очень сильно отличался от предыдущего. У него была совершенно другая внутренняя организация и он мог исполнять до пяти команд одновременно. Еще одно нововведение у Pentium Pro — двухуровневая кэш-память. Процессор содержал 8 Кбайт памяти для часто используемых команд и еще 8 Кбайт для часто используемых данных. В корпусе Pentium Pro рядом с процессором (но не на самой микросхеме) находилась другая кэш-память объемом в 256 Кбайт. Большой объем кэш-памяти в Pentium Pro отчасти компенсировался отсутствием MMX-команд (первоначально Intel не удалось спроектировать микросхему адекватного размера, отвечавшую критерию рентабельности). Когда технологическая база позволила совместить в рамках одной микросхемы набор MMX-команд и большой кэш, новая модель получила название Pentium II. Через некоторое время для улучшенной передачи трехмерной графики в процессор были введены дополнительные мультимедийные команды под названием SSE (Streaming SIMD Extensions — потоковые SIMD-расширения) — в результате появился процессор Pentium III [Raman et al., 2000]. Правда, согласно внутренней номенклатуре компании это все тот же Pentium II. Следующая модель Pentium получила новую внутреннюю архитектуру. Одновременно было решено перейти с римских цифр в обозначениях моделей на арабские. Так появился процессор Pentium 4. По традиции он превосходил все предыдущие модели по быстродействию. В версии с тактовой частотой 3,06 ГГц была реализована новая функция — гиперпоточность (hyperthreading). Она позволяет программам разделять задачи на два программных потока, которые обрабатываются процессором параллельно; следовательно, скорость исполнения повышается. Кроме того, для дальнейшего повышения скорости обработки звуковых и видеоданных был внедрен дополнительный набор SSE-команд. В 2006 году фирма Intel сменила название бренда Pentium на Core и выпустила двухъядерную микросхему Core 2 duo. Когда возникла необходимость в более дешевой одноядерной версии микросхемы, фирма Intel стала продавать Core 2 duo с одним отключенным ядром, потому что небольшие дополнительные затраты на каждой производимой микросхеме обходились несравненно дешевле огромных затрат на проектирование и тестирование новой

модели «с нуля». Серия Core продолжала развиваться; модели i3, i5 и i7 стали популярными решениями для низко-, средне- и высокопроизводительных компьютеров.

Помимо основной линейки процессоров, которую мы рассмотрели, Intel разрабатывает специальные варианты микросхем для отдельных сегментов рынка. В начале 1998 года компания запустила новую линейку под названием Celeron. По сути, это был дешевый вариант Pentium 2 с пониженной производительностью для низкопроизводительных компьютеров. Поскольку у процессора Celeron такая же архитектура, как у Pentium 2, мы не будем обсуждать его в этой книге. В июне 1998 года компания Intel выпустила специальную версию Pentium 2 для верхнего сегмента рынка — Xeon. Эту модификацию снабдили кэш-памятью большего объема, ускоренной внутренней шиной и усовершенствованными средствами поддержки мультипроцессорного режима, однако по всем остальным параметрам она ничем не отличалась от Pentium 2, что дает нам полное право не рассматривать ее отдельно. Для процессоров Pentium III, как и более поздних микросхем, также была разработана версия Xeon. На более новых процессорах одной из особенностей Xeon является увеличенное количество ядер. В 2003 году появилась микросхема Pentium M (где M — сокращение от «Mobile») для портативных компьютеров. Она задумывалась как составная часть новой архитектуры Centrino, которая должна была, во-первых, снизить энергопотребление, увеличив тем самым ресурс аккумулятора, во-вторых, обеспечить возможность производства более компактных и легких корпусов, в-третьих, предоставить встроенную поддержку беспроводных сетевых соединений по стандарту IEEE 802.11 (WiFi). Pentium M потребляет меньше энергии и отличается большей компактностью по сравнению с Pentium 4; вероятно, эти две характеристики вскоре позволят ему (и его преемникам) вытеснить микроархитектуру Pentium 4 в будущих продуктах Intel. Все микросхемы Intel обратно совместимы со всеми своими предшественниками вплоть до модели 8086. Другими словами, программы, написанные когда-то для 8086, исполняются на Pentium 4 без каких бы то ни было изменений. Обратная совместимость в течение длительного времени является одним из основных принципов проектирования в Intel — соблюдение этого принципа позволяет сохранить предыдущие инвестиции в программное обеспечение. Естественно, поскольку модель Pentium 4 на три порядка сложнее, чем 8086, ее возможности несопоставимо шире. Из-за постепенных расширений, которые проектировщикам процессоров приходилось внедрять для достижения этой цели, архитектура получилась не такой элегантной, как если бы разработчики Pentium 4 получили 42 миллиона транзисторов и начали бы все строить «с нуля».

В ноябре 2004 компания Intel была вынуждена отменить выпуск модели Pentium 4 с тактовой частотой 4 ГГц из-за проблем с теплоотводом. Большие вентиляторы способны решить проблему, но они слишком шумные, что, естественно, не нравится пользователям. Водяное охлаждение, применяемое на мэйнфреймах, совершенно неприемлемо для настольных машин, не говоря уже о ноутбуках. Поэтому некогда безудержный рост тактовой частоты, вероятно, на некоторое время поуспокоится — по крайней мере, до того момента, когда инженеры Intel придумают эффективный способ отвода тепла. В планах Intel теперь другие новации — компания планирует разместить на одной микросхеме два процессора и снабдить ее общим кэшем большого объема. Поскольку величина энергопотребления определяется напряжением и тактовой частотой, два процессора на одной схеме потребляют значительно меньше энергии, чем один, работающий на удвоенной скорости. Таким образом, действие закона Мура может в будущем перейти с повышения тактовых частот на увеличение количества ядер

и объема встроенных кэшей. Использование мультипроцессоров усложняет задачу программиста, потому что в отличие от изощренных однопроцессорных микроархитектур прошлого, способных выжать больше производительности из существующих программ, мультипроцессоры заставляют программиста явно управлять параллельным выполнением с использованием программных потоков, семафоров, общей памяти и других технологий — хлопотных и подверженных ошибкам.

## 20. **Компьютерная ши́на**

(англ. computer bus) в архитектуре компьютера — соединение, служащее для передачи данных между функциональными блоками компьютера. В устройстве шины можно различить механический, электрический (физический) и логический (управляющий) уровни. В отличие от соединения точка-точка, к шине обычно можно подключить несколько устройств по одному набору проводников. Каждая шина определяет свой набор разъёмов (соединений) для физического подключения устройств, карт и кабелей.

Современные компьютерные шины используют как параллельные, так и последовательные соединения и могут иметь параллельные (англ. multidrop) и цепные (англ. daisy chain) топологии. В случае USB и некоторых других шин могут также использоваться хабы (концентраторы). Некоторые виды скоростных шин (Fibre Channel, InfiniBand, скоростной Ethernet, SDH) для передачи сигналов используют не электрические соединения, а оптические.

Присоединители к шине, разнообразные разъёмы, как правило, унифицированы и позволяют подключить различные устройства к шине.

Основные шины расширения.

Устройства, подключаемые к шине, разделяются на два основных типа: busmasters и busslaves. Busmasters - это устройства, способные управлять работой шины, то есть инициировать запись/чтение и т. п. Busslaves - соответственно, устройства, которые могут только отвечать на запросы.

Важнейшей характеристикой шины является ее разрядность, которая определяет количество данных, передаваемых по шине одновременно (за один такт). Понятно, что чем больше разрядность шины, тем больше ее производительность, хотя, правда, это и не всегда так, так как количество передаваемой в секунду информации зависит еще и от собственно ее частоты. По назначению шины можно разделить на три категории:

· Шина данных;

· Адресная шина;

· Шина управления.

Шина данных

По этой шине происходит обмен данными между процессором, картами расширения и памятью. Особую роль здесь играет так называемый DMA-контроллер (DirectMemoyAccess), через который происходит управление транспортировкой данных, минуя процессор. Такой способ хорош тем, что освобождает ресурсы CPU для других нужд. Разрядность шины данных может составлять 8 бит, 16 бит, 32 бит и так далее.

Адресная шина

Данные, которые в большом количестве кочуют по шине через материнскую плату, должны, в конце концов, сделать где-нибудь

помежкточную остановку. Местом для этой остановки являются отдельные ячейки памяти. Каждая ячейка должна иметь свой адрес. Следовательно, объем памяти, который может адресовать процессор, зависит от разрядности адресной шины. Его можно вычислить по формуле:

Объем адресуемой памяти = 2n, где n - число линий в адресной шине.

Процессор 8088, например, имел в своем распоряжении 20 адресных линий и, таким образом, мог адресовать всего 1 Mb памяти (220=1048576). В компьютерах на базе процессора 80286 адресная шина была уже 24-разрядной, а процессоры 80486 имеют уже 32-разрядную шину, которая позволяет адресовать 4 им гигабайта памяти.

Шина управления

Конечно же, незачем просто транспортировать данные по шине и располагать их в памяти, если непонятно, куда их нужно переслать и какое устройство в них нуждается. Разрешение этой проблемы на себя шина контроллера, называемая также системной шиной, или шиной управления.

В качестве конечных пунктов системной шины можно рассматривать слоты расширения, интегрированные на материнскую плату контроллеры и прочее. Все эти устройства соединены между собой шиной управления. Логично предположить, что от ее производительности во многом зависит производительность всей системы, и чем больше тактовая частота и разрядность этой шины, тем лучше. Внешний вид слотов расширения, которые установлены на материнской плате, зависит именно от типа шины управления. Понятно, что, например, разъемы 32-разрядной системной шины будут отличаться от разъемов 16-разрядной шины. http://surazhspk.narod.ru/kop/Architec/public\_html/page25.html

## 

## 21. **Многопроцессорные ВС. Многомашинные вычислительные комплексы**

Вычислительная система — это совокупность одного или нескольких компьютеров или процессоров, программного обеспечения и периферийного оборудования, организованная для совместного выполнения информационно-вычислительных процессов. В вычислительной системе компьютер может быть один, но агрегированный с многофункциональным периферийным оборудованием. Стоимость периферийного оборудования часто во много раз превосходит стоимость компьютера. В качестве распространенного примера одномашинной ВС можно привести систему телеобработки информации. Но все же классическим вариантом ВС является многомашинный и многопроцессорный варианты

Вычислительная система может строиться на основе целых компьютеров — многомашинная ВС, либо отдельных процессоров — многопроцессорная ВС.

Многопроцессорные вычислительные системы — это системы, содержащие несколько процессоров, информационно взаимодействующих между собой либо на уровне регистров процессорной памяти, либо на уровне оперативной памяти. Последний тип взаимодействия принят в большинстве случаев, так как организуется значительно проще и сводится к созданию общего поля оперативной памяти для всех процессоров. Общий доступ к внешней памяти и к устройствам ввода-вывода обеспечивается обычно через каналы ОП. Важным является и то, что многопроцессорная вычислительная система работает под управлением единой операционной системы, общей для всех процессоров. Это существенно улучшает динамические характеристики ВС, но требует наличия специальной, весьма сложной операционной системы. Быстродействие и надежность многопроцессорных ВС по сравнению с многомашинными, взаимодействующими на 3-м уровне, существенно повышаются, во-первых, ввиду ускоренного обмена информацией между процессорами, более быстрого реагирования на ситуации, возникающие в системе; во-вторых, вследствие большей степени резервирования устройств системы (система сохраняет работоспособность, пока работоспособны хотя бы по одному модулю каждого типа устройств). Типичным примером массовых многомашинных ВС могут служить компьютерные сети, примером многопроцессорных вычислительных систем (МПВС) — суперкомпьютеры.

Многомашинные вычислительные системы — это системы, содержащие несколько одинаковых или различных, относительно

самостоятельных компьютеров, связанных между собой через устройство обмена информацией, в частности, по каналам связи. В последнем случае речь идет об информационно-вычислительных сетях. В многомашинных ВС каждый компьютер работает под управлением своей операционной системы (ОС). А поскольку обмен информацией между машинами выполняется под управлением ОС, взаимодействующих друг с другом, динамические характеристики процедур обмена несколько ухудшаются (требуется время на согласование работы самих ОС). Информационное взаимодействие компьютеров в многомашинной ВС может быть организовано на уровне:

· процессоров;

· оперативной памяти (ОП);

· каналов связи.

## **22. ВС с массовым параллелизмом. Двухъядерные процессоры Intel**

## Массовый параллелизм

## Процессоры с массовым параллелизмом (Massively Parallel Processors, MPP) — это огромные суперкомпьютеры стоимостью в несколько миллионов долларов. Они используются в различных отраслях науки и техники для выполнения сложных вычислений, обработки большого числа транзакций в секунду, управления большими базами данных и т. д. Изначально это были суперкомпьютеры, предназначенные в основном для научных расчетов, но сейчас многие из них находят применение в коммерции

## BlueGene

## В качестве первого примера процессора с массовым параллелизмом рассмотрим систему IBM BlueGene. Этот проект был задуман IBM в 1999 году как суперкомпьютер для решения вычислительных задач большой сложности в области биологии.

## В частности, биологи считают, что функции белка определяются его трехмерной структурой. Но определение формы даже одной небольшой молекулы белка на суперкомпьютерах того времени потребовало бы нескольких лет вычислений. При этом в человеческом организме около полумиллиона различных белков, некоторые из них исключительно сложны, и нарушения в структуре любого могут приводить к серьезным наследственным заболеваниям (например, муковисцидозу).

## Очевидно, что для расчета трехмерной структуры всех человеческих белков требуется на несколько порядков повысить вычислительную мощность, и моделирование формы белковой молекулы — лишь одна из задач, на решение которых направлен проект BlueGene. Столь же сложные задачи измолекулярной динамики, моделирования климата, астрономии и даже финансового моделирования также требуют совершенствования суперкомпьютеров.

## Двухъядерные процессоры Intel

## Модельный ряд двухъядерных процессоров Intel включает четыре модели: процессор Intel Pentium Processor Extreme Edition 840 и три процессора семейства Intel Pentium D (табл. 1).

## Intel Pentium Processor Extreme Edition 840 — флагман линейки десктопных двухъядерных процессоров Intel. Это процессор с двумя ядрами на одном кристалле и с корпусировкой LGA775. Тактовая частота составляет 3,2 ГГц.

## Все ядра процессора имеют микроархитектуру NetBurst, и среди всего этого семейства только процессор Intel Pentium Processor Extreme Edition 840 поддерживает технологию Hyper-Threading, что в совокупности обеспечивает обработку до четырех потоков. Поэтому один такой физический процессор с точки зрения операционной системы определяется как четыре логических.

## Каждое ядро процессора имеет собственный кэш второго уровня L2 объемом 1 Мбайт, а общий объем кэша L2 составляет 2 Мбайт. Процессор производится по 90-нанометровому технологическому процессу

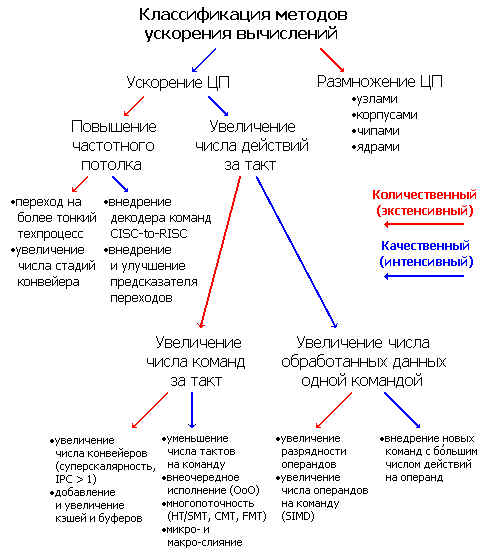
## 

## 23. **Методы и средства повышения быстродействия ЭВМ**

Есть всего два основных способа повышения быстродействия ЭВМ:

1. За счет повышения быстродействия элементной базы (тактовой частоты). Быстродействие процессора растет пропорционально росту тактовой частоты, при этом не требуется изменения системы программирования и пользовательских программ.

2. За счет увеличения числа одновременно работающих в одной задаче ЭВМ, процессоров, АЛУ, умножителей и так далее, то есть за счет параллелизма выполнения операций. Это требует использования сложных систем параллельного программирования. Это крупный недостаток метода.



## 24. **Конвейерные вычислительные системы.**

Этот метод обработки подобен функционированию сборочного конвейера и особенно удобен для работы с векторами и массивами. Конвейер состоит из набора функциональных устройств (ФУ), которые реализуют различные операции, и могут работать одновременно. Когда операция k завершается, она передает свой результат операции (k + 1) и ожидает от операции (k-1) нового задания. Если каждая операция занимает t единиц времени и всего n операций, то завершение конвейерной обработки одного операнда потребует n\*t единиц времени. Однако, если на конвейерную обработку продолжают поступать новые операнды, результаты могут выдаваться со скоростью один в каждые t единиц времени.

Пример Конвейер для сложения двух чисел с плавающей точкой. Конвейер образуют следующие операции:

1.Выделить экспоненты обоих чисел.

2.Сравнить экспоненты.

3.Сдвинуть точку в числе с меньшей экспонентой для их уравнения.

4.Сложить дроби.

5.Нормализовать результат.

Этот конвейер позволяет выполнять до пяти сложений одновременно.

Конвейерная обработка может быть синхронной; т. е. время t, отпущенное на выполнение каждой операции, постоянно и является общим для всех ФУ. Каждые t единиц времени результат каждой операции перемещается по конвейеру, чтобы стать входом для следующей операции. Однако при синхронном подходе многие функциональные устройства могут простаивать довольно длительное время из-за того, что требующееся для них время обработки меньше t. В этом случае время t должно определяться самой медленной операцией конвейера, при этом необходимо учитывать, что время выполнения операции зависит от обрабатываемых данных.

Конвейерная обработка также может быть и асинхронной. В асинхронном конвейере процесс обработки может быть ускорен по сравнению с синхронным. Результат операции k асинхронного конвейера может быть послан операции (k+1), как только операция k выполнена, а операция (k+1) свободна. Обычно асинхронный конвейер предполагает наличие регистров. Операция использует значение своего входного регистра для вычисления значения выходного регистра.

При этом ФУ возобновляет свою работу, когда его входной регистр заполнен новым значением, а само ФУ и его выходной регистр свободны. Передача содержимого выходного регистра k-того ФУ во входной регистр k+1-вого ФУ может произойти, если выходной регистр k-того ФУ заполнен, а входной регистр k+1-вого ФУ и само ФУ свободны.

Матричные вычислительные системы.

Назначение матричных вычислительных систем - обработка больших массивов данных (во многом схоже с назначением векторных ВС). В основе матричных систем лежит матричный процессор (array processor), состоящий из регулярного массива процессорных элементов (ПЭ).

Между матричными и векторными системами есть существенная разница. Матричный процессор интегрирует множество идентичных функциональных блоков (ФБ), логически объединенных в матрицу и работающих в SIMD-стиле. Не столь существенно, как конструктивно реализована матрица процессорных элементов — на едином кристалле или на нескольких. Важен сам принцип - ФБ логически скомпонованы в матрицу и работают синхронно, то есть присутствует только один поток команд для всех. Векторный процессор имеет встроенные команды для обработки векторов данных, что позволяет эффективно загрузить конвейер из функциональных блоков. В свою очередь, векторные процессоры проще использовать, потому что команды для обработки векторов — это более удобная для человека модель программирования, чем SIMD.

Структуру матричной вычислительной системы можно представить в следующем виде.

Обобщенная модель матричной ВС

Рассмотрим компоненты обобщенной модели матричной ВС.

\*Массив процессоров (МПр) осуществляет параллельную обработку множественных элементов данных.

\*Контроллер массива процессоров (КМП) генерирует единый поток команд, управляющий обработкой данных в массиве процессоров, выполняет последовательный программный код, реализует операции условного и безусловного переходов, транслирует в МПр команды, данные и сигналы управления. Команды обрабатываются процессорами в режиме жесткой синхронизации.

\*Сигналы управления используются для синхронизации команд и пересылок, а также для управления процессом вычислений, в частности определяют, какие процессоры массива должны выполнять операцию, а какие - нет.

\*Шина широковещательной рассылки служит для передачи команд, данных и сигналов управления из КМП в массив процессоров.

\*Шина результата служит для трансляции результатов вычислений из МПр в КМП (это требуется, поскольку выполнение операций условного перехода зависит от результатов вычислений).

\*Интерфейсная ВМ (front-end computer)

служит для обеспечения пользователя удобным интерфейсом при создании и отладке программ. В роли такой ВМ выступает универсальная вычислительная машина, на которую дополнительно возлагается задача загрузки программ и данных в КМП. Кроме того, загрузка программ и данных в КМП может производиться и напрямую с устройств ввода/вывода, например с магнитных дисков. После загрузки КМП приступает к выполнению программы, транслируя в МПр по широковещательной шине соответствующие SIMD-команды

Рассматривая массив процессоров, следует учитывать, что для хранения множественных наборов данных в нем, помимо множества процессоров, должно присутствовать и множество модулей памяти. Кроме того, в массиве должна быть реализована сеть взаимосвязей, как между процессорами, так и между процессорами и модулями памяти.

Таким образом, под термином массив процессоров понимают блок, состоящий из процессоров, модулей памяти и сети соединений. Дополнительную гибкость при работе с рассматриваемой системой обеспечивает механизм маскирования, позволяющий привлекать к участию в операциях лишь определенное подмножество из входящих в массив процессоров. Маскирование реализуется как на стадии компиляции, так и на этапе выполнения, при этом процессоры, исключенные путем установки в ноль соответствующих битов маски, во время выполнения команды простаивают.

Векторно-конвейерные вычислительные системы относятся к классу SIMD-систем. Основные принципы, заложенные в архитектуру векторно-конвейерных систем:

· конвейерная организация обработки потока команд;

· введение в систему команд набора векторных операций, которые позволяют оперировать с целыми массивами данных.

Длина обрабатываемых векторов в современных векторно-конвейерных системах составляет, как правило, 128 или 256 элементов. Основное назначение векторных операций состоит в распараллеливании выполнения операторов цикла, в которых обычно сосредоточена большая часть вычислительной работы.

Первый векторно-конвейерный компьютер Cray-1 появился в 1976 году. Архитектура этого компьютера оказалась настолько удачной, что он положил начало целому семейству компьютеров.

Современные векторно-конвейерные системы имеют иерархическую структуру:

· на нижнем уровне иерархии расположены конвейеры операций (например, конвейер (pipeline) сложения вещественных чисел, конвейер умножения таких же чисел и т.п.);

· некоторая совокупность конвейеров операций объединяется в конвейерное функциональное устройство;

· векторно-конвейерный процессор содержит ряд конвейерных функциональных устройств;

· несколько векторно-конвейерных процессоров (2-16), объединенных общей памятью, образуют вычислительный узел;

· несколько таких узлов объединяются с помощью коммутаторов, образуя либо NUMA-систему либо MPP-систему.

Типичными представителями такой архитектуры являются компьютеры CRAY J90/T90, CRAY SV1, NEC SX-4/SX-5. Уровень развития микроэлектронных технологий не позволяет в настоящее время производить однокристальные векторно-конвейерные процессоры, поэтому эти системы довольно громоздки и чрезвычайно дороги.

Каждая часть конвейера операций называется ступенью конвейера операций, а общее число ступеней - длиной конвейера операций.

## 

## 25. **Способы функциональной организации ВС**

При автоматическом выполнении программы процессором команды последовательно поступают из оперативной памяти (ОП) в ЦУУ на время их выполнения АЛУ. Интервал времени, в течение которого процессор выполняет команду, называют рабочим циклом ЭВМ. Величина рабочего цикла зависит от структуры команды, типа операций, структуры операционных блоков АЛУ. По принципу организации управления вычислительным процессом различают процессоры схемного типа или «жесткой» логикой, с микропрограммным и смешанным (микропрограммно-схемным) управлением.

Схемное управление - управление, при котором для выполнения любой операции последовательность управляющих сигналов задается логическими схемами. Различают центральное, местное и смешанное схемное управление.

В процессорах с центральным управлением длительность рабочего цикла выбирается такой, чтобы за время между двумя управляющими сигналами выполнялась самая длинная операция в процессоре. Такие процессоры получили название синхронных, а блок, в котором формируются управляющие сигналы для всех исполнительных устройств ЭВМ, называют центральным блоком управления (ЦБУ).

В синхронных процессорах при выполнении большинства операций, особенно коротких (например, операция сложения), происходит потеря машинного времени, связанная с непроизводительными простоями процессора. Однако структура процессора отличается простотой, экономичностью и удобна в эксплуатации.

В процессорах с местным управлением вычислительным процессом управление производится так, что каждая операция выполняется после выполнения предыдущей операции. При этом каждое исполнительное устройство после окончания работы формирует сигнал «Конец работы», который одновременно является сигналом «Начало работы» другого исполнительного устройства. Процессоры с переменной длительностью рабочего цикла, величина которого зависит от вида выполняемой операции и кодов операндов, называют асинхронными. В асинхронных процессорах основные исполнительные устройства имеют местные (автономные) блоки управления, что резко повышает быстродействие таких процессоров, так как отсутствуют простои между реальными циклами выполнения команд.

Основной недостаток асинхронных процессоров — их сложность.

В процессорах со смешанным управлением исполнение простейших операций осуществляется в синхронном режиме, а наиболее сложные операции (например, деление, умножение и др.) - в асинхронном. При смешанном управлении процессор содержит как центральный блок, так и местные блоки управления операциями. Смешанный способ управления вычислительным процессом позволяет получить высокое быстродействие процессора при умеренных затратах оборудования, а поэтому наиболее распространен в современных ЭВМ.

Микропрограммное управление (см. рис. 4.4) основано на замене управляющих логических схем (см. рис. 4.5) специальной программой, хранящейся в ПЗУ. При таком управлении каждая команда разделяется на ряд элементарных этапов, получивших название микроопераций.

Последовательность микрокоманд, выполняющих одну команду (операцию), представляет собой микропрограмму. Для характеристики временных соотношений между различными этапами операции используется понятие машинный такт, определяющий интервал времени, в течение которого выполняется одна или одновременно несколько микроопераций.

Достоинство микропрограммного управления заключается в том, что для изменения вида операций нет необходимости в переделке сложных электронных схем, неизбежной в ЭВМ со схемным управлением, а следует только изменить микропрограмму. Это обстоятельство дает возможность в данной ЭВМ использовать программы, составленные для другой ЭВМ. Благодаря этому микропрограммное управление получило широкое распространение в современных ЭВМ.

## По типу вычислительные системы можно разделить на многомашинные и многопроцессорные ВС. Исторически многомашинные вычислительные системы (ММС) появились первыми. Уже при использовании ЭВМ первых поколений возникали задачи повышения производительности, надежности и достоверности вычислений. Для этих целей использовали комплекс машин, схематически показанный на рис. 11.2,a 4 . Рис. 11.2. Типы ВС: а — многомашинные комплексы; б — многопроцессорные системы Положения 1 и 3 электронного ключа (ЭК) обеспечивали режим повышенной надежности. При этом одна из машин выполняла вычисления, а другая находилась в «горячем» или «холодном» резерве, т.е. в готовности заменить основную ЭВМ. Положение 2 электронного ключа соответствовало случаю, когда обе машины обеспечивали параллельный режим вычислений. Здесь возможны две ситуации: а) обе машины решают одну и ту же задачу и периодически сверяют результаты решения. Тем самым обеспечивался режим повышенной достоверности, уменьшалась вероятность появления ошибок в результатах вычислений. Примерно по такой же схеме построены управляющие бортовые вычислительные комплексы космических аппаратов, ракет, кораблей. Возможность обмена информацией между машинами сохраняется. Этот вид работы относится к режиму повышенной производительности. Она широко используется в практике организации работ на крупных вычислительных центрах, оснащенных несколькими ЭВМ высокой производительности. Схема, представленная на рис. 11.2, а, была неоднократно повторена в различных модификациях при проектировании разнообразных специализированных ММС. Основные различия ММС заключаются, как правило, в организации связи и обмена информацией между ЭВМ комплекса. Каждая из них сохраняет возможность автономной работы и управляется собственной ОС. Любая другая подключаемая ЭВМ комплекса рассматривается как специальное периферийное оборудование. В зависимости от территориальной разобщенности ЭВМ и используемых средств 5 сопряжения обеспечивается различная оперативность их информационного взаимодействия. Характеристика возможных уровней и средств взаимодействия изложена в п. 11.3.

## Многопроцессорные системы (МПС) строятся при комплексировании нескольких процессоров (рис. 11.2, б). В качестве общего ресурса они имеют общую оперативную память (ООП). Параллельная работа процессоров и использование ООП обеспечивается под управлением единой операционной системы. По сравнению с ММС здесь достигается наивысшая оперативность взаимодействия вычислителей-процессоров. Многие исследователи [27] считают, что использование МПС является основным магистральным путем развития вычислительной техники новых поколений.

## Однако МПС имеет и существенные недостатки. Они в первую очередь связаны с использованием ресурсов общей оперативной памяти. При большом количестве комплексируемых процессоров возможно возникновение конфликтных ситуаций, когда несколько процессоров обращаются с операциями типа «чтение» и «запись» к одним и тем же областям памяти. Помимо процессоров к ООП подключаются все каналы (процессоры ввода-вывода), средства измерения времени и т.д. Поэтому вторым серьезным недостатком МПС является проблема коммутации абонентов и доступа их к ООП. От того, насколько удачно решаются эти проблемы, и зависит эффективность применения МПС. Это решение обеспечивается аппаратно-программными средствами. Процедуры взаимодействия очень сильно усложняют структуру ОС МПС. Накопленный опыт построения подобных систем показал, что они эффективны при небольшом числе комплексируемых процессоров (от 2—4 до 10). В отечественных системах «Эльбрус» обеспечивалась возможность работы до 10 процессоров, до 32 модулей памяти, до 4 процессоров ввода-вывода и до 16 процессоров связи. Создание подобных коммутаторов представляет сложную техническую задачу, тем более, что они должны быть дополнены буферами для организации очередей запросов. Для разрешения конфликтных ситуаций необходимы схемы приоритетного обслуживания. До настоящего времени в номенклатуре технических средств ЭВТ отсутствуют высокоэффективные коммутаторы общей памяти.

## По типу ЭВМ или процессоров, используемых для построения ВС, различают однородные и неоднородные системы. Однородные системы предполагают комплексирование однотипных ЭВМ (процессоров), неоднородные — разнотипных. В однородных системах значительно упрощается разработка и обслуживание технических и программных (в основном ОС) средств. В них обеспечивается возможность стандартизации и унификации соединений и процедур взаимодействия элементов системы. Упрощается обслуживание систем, облегчается модернизация и их развитие. Вместе с тем существуют и неоднородные ВС, в которых комплексируемые элементы очень сильно отличаются по своим техническим и функциональным характеристикам. Обычно это связано с необходимостью параллельного выполнения многофункциональной обработки. Так, при построении ММС, обслуживающих каналы связи, целесообразно объединять в комплекс связные, коммуникационные машины и машины обработки данных. В таких системах коммуникационные ЭВМ выполняют функции связи, контроля получаемой и передаваемой информации, формирование пакетов задач и т.д. ЭВМ 6 обработки данных не занимаются не свойственными им работами по обеспечению взаимодействия в сети, а все их ресурсы переключаются на обработку данных. Неоднородные системы находят применение и МПС. Многие ЭВМ, в том числе и ПЭВМ, могут использовать сопроцессоры: десятичной арифметики, матричные и т.п.

## По степени территориальной разобщенности вычислительных модулей ВС делятся на системы совмещенного (сосредоточенного) и распределенного (разобщенного) типов. Обычно такое деление касается только ММС. Многопроцессорные системы относятся к системам совмещенного типа. Более того, учитывая успехи микроэлектроники, это совмещение может быть очень глубоким. При появлении новых СБИС появляется возможность иметь в одном кристалле несколько параллельно работающих процессоров. В совмещенных и распределенных ММС сильно различается оперативность взаимодействия в зависимости от удаленности ЭВМ. Время передачи информации между соседними ЭВМ, соединенными простым кабелем, может быть много меньше времени передачи данных по каналам связи. Как правило, все выпускаемые в мире ЭВМ имеют средства прямого взаимодействия и средства подключения к сетям ЭВМ. Для ПЭВМ такими средствами являются нуль-модемы, модемы и сетевые карты как элементы техники связи.

## По методам управления элементами ВС различают централизованные, децентрализованные и со смешанным управлением. Помимо параллельных вычислений, производимых элементами системы, необходимо выделять ресурсы на обеспечение управления этими вычислениями. В централизованных ВС за это отвечает главная, или диспетчерская, ЭВМ (процессор). Ее задачей является распределение нагрузки между элементами, выделение ресурсов, контроль состояния ресурсов, координация взаимодействия. Централизованный орган управления в системе может быть жестко фиксирован, или эти функции могут передаваться другой ЭВМ (процессору), что способствует повышению надежности системы. Централизованные системы имеют более простые ОС. В децентрализованных системах функции управления распределены между ее элементами. Каждая ЭВМ (процессор) системы сохраняет известную автономию, а необходимое взаимодействие между элементами устанавливается по специальным наборам сигналов. С развитием ВС и, в частности, сетей ЭВМ интерес к децентрализованным системам постоянно растет. В системах со смешанным управлением совмещаются процедуры централизованного и децентрализованного управления. Перераспределение функций осуществляется в ходе вычислительного процесса, исходя из сложившейся ситуации.

## По принципу закрепления вычислительных функций за отдельными ЭВМ (процессорами) различают системы с жестким и плавающим закреплением функций. В зависимости от типа ВС следует решать задачи статического или динамического размещения программных модулей и массивов данных, обеспечивая необходимую гибкость системы и надежность ее функционирования.

## По режиму работы ВС различают системы, работающие в оперативном и неоперативном временных режимах. Первые, как правило, используют режим реального масштаба времени. Этот режим характеризуется жесткими 7 ограничениями на время решения задач в системе и предполагает высокую степень автоматизации процедур ввода-вывода и обработки данных.

## Наибольший интерес у исследователей всех рангов (проектировщиков, аналитиков и пользователей) вызывают структурные признаки ВС. От того, насколько структура ВС соответствует структуре решаемых на этой системе задач, зависит эффективность применения ЭВМ в целом. Структурные признаки, в свою очередь, отличаются многообразием: топология управляющих и информационных связей между элементами системы, способность системы к перестройке и перераспределению функций, иерархия уровней взаимодействия элементов. В наибольшей степени структурные характеристики определяются архитектурой системы.

## 26. **Разработка уровня микроархитектуры**

При разработке уровня микроархитектуры (как и при разработке других уровней) постоянно приходится идти на компромисс. У компьютера есть много важных характеристик: быстродействие, стоимость, надежность, простота использования, объем потребляемой энергии, физические размеры. При разработке центрального процессора очень важную роль играет правильный выбор между быстродействием и стоимостью.

Быстродействие и стоимость.

Существует три основных подхода, которые позволяют увеличить скорость выполнения операций:

1. Сокращение количества циклов, необходимых для выполнения команды.

(Длина пути - число циклов, необходимых для выполнения набора операций. Иногда длину пути можно уменьшить с помощью дополнительного аппаратного обеспечения)

1. Упрощение организации машины таким образом, чтобы можно было сделать цикл короче.
2. Одновременное выполнение нескольких операций.

Одновременное выполнение нескольких операций - самый продуктивный подход. Он дает возможность значительно увеличить быстродействие компьютера. Вообще говоря, эта идея является основой проектов современных компьютеров.

На одной чаше весов находится быстродействие, на другой - стоимость. Чем больше участок, тем больше микросхема. И стоимость микросхемы растет гораздо быстрее, чем занимаемое ею пространство.

Cуществует определенная зависимость между быстродействием и стоимостью. Если сократить каждое слово управляющей памяти на 5 бит, это приведет к снижению скорости работы генератора. Инженер при разработке компьютера должен принимать во внимание его предназначение, чтобы сделать правильный выбор. В компьютере с высокой производительностью использовать декодер не рекомендуется, а вот для дешевой машины он вполне подойдет.

Сокращение длины пути

Способы:

1. Слияние цикла интерпретатора с микропрограммой
2. Помещение основного цикла в конце каждой последовательности микрокоманд.
3. Переход от 2-шинной к 3-шинной архитектуре.

Тракт данных

Тракт данных - это часть центрального процессора, состоящая из АЛУ (арифметико-логического устройства), его входов и выходов.

Основным компонентом любого компьютера является тракт данных. Он содержит несколько регистров, одну, две или три шины, а также один или несколько функциональных блоков, например АЛУ и схему сдвига. В основном цикле вызываются несколько операндов из регистров и передаются по шинам к АЛУ и другому функциональному блоку на исполнение. После завершения операции результаты вновь сохраняются в регистрах.

Тракт данных может управляться контроллером последовательности, который вызывает микрокоманды из управляющей памяти. Каждая микрокоманда содержит биты, управляющие трактом данных в течение одного цикла. Эти биты определяют, какие операнды нужно выбрать, какую операцию выполнить и что делать с результатами. Кроме того, каждая микрокоманда указывает на следующую микрокоманду (обычно в ней содержится ее адрес). Некоторые микрокоманды изменяют этот базовый адрес с помощью операции ИЛИ.

Цикл тракта данных можно разбить на подциклы. Начало подцикла 1 инициирует спад синхронизирующего сигнала. Далее показано, что происходит во время каждого из подциклов. В скобках приводится величина подцикла.

1. Устанавливаются сигналы управления (Ат).

2. Значения регистров загружаются на шину В (Ад:).

3. Действуют АЛУ и схемы сдвига (Ау).

4. Результаты проходят по шине С обратно к регистрам (Аг).

Микрокоманды

Каждая микрокоманда представляет собой управляющее слово, которое должно где-то храниться. Ее извлечение потребует какого-то времени, а ее выполнение можно начинать только после ее извлечения. Если микрокоманда должна выполняться за один период тактового сигнала, то тактовый сигнал должен быть поделен на еще более мелкие промежутки времени. Тогда на первом промежутке будет происходить извлечение микрокоманды, а на втором – ее выполнение. Причем ясно, что выполнение микрокоманды будет происходить дольше, чем ее извлечение, поскольку потребует выполнения нескольких действий – извлечения аргументов, выполнения операции и записи результата.

Макросы

В языках ассемблера и некоторых других языках используются макрокоманды. Макрокоманда (макрос) - это сокращенное обозначение группы команд базового языке, в котором используется макрокоманда (в общем случае - некоторого текста). Применение макросов позволяет расширять базовый язык в каком-либо направлении, дополняя его новыми командами - макрокомандами, определяемыми через команды базового языка в виде макроопределений. Макроопределения пишут на макроязыке, который обычно, представляет собой надстройку над базовым языком, но может быть и самостоятельным языком.

Макроопределение, в общем случае, представляет собой написанную на макроязыке подпрограмму, которая запускается макросом (макровызовом) и выполняется во время трансляции исходной программы. Макроопределение может генерировать в зависимости от параметров макровызова и выводить в выходной файл некоторый текст - макрорасширение, но может и не порождать никакого текста, а, например, выполнять некоторую операцию с таблицей, реализованной в виде массива макропеременных.

Макрокоманду можно рассматривать как вызов подпрограммы макроязыка – макровызов.

макрорасширение - описанный в макроопределении текст на базовом языке.

## 

## 27. **Повышение производительности**. (Кэш-память. Прогнозирование ветвлений. Исполнение с изменением последовательности и подмена регистров.)

Усовершенствования, распадаются на две категории: усовершенствование реализации и усовершенствование архитектуры. Усовершенствования реализации — это такие способы построения нового процессора и памяти, после применения которых система работает быстрее, но архитектура при этом не меняется. Изменение реализации без изменения архитектуры означает, что старые программы будут работать на новой машине, а это очень важно для успешной продажи. Чтобы усовершенствовать реализацию, можно, например, использовать более быстрый задающий генератор, но это не единственный способ.

Однако некоторые типы усовершенствований можно осуществить только путем изменения архитектуры. Иногда, например, нужно добавить новые команды или регистры, причем таким образом, чтобы старые программы могли работать на новых моделях. В этом случае для достижения полной производительности программное обеспечение должно быть изменено или, по крайней мере, заново скомпилировано на новом компиляторе.

Кеш-память

Одним из самых важных вопросов при разработке компьютеров было и остается построение такой системы памяти, которая могла бы передавать операнды процессору с той же скоростью, с которой он их обрабатывает. Быстрый рост скорости работы процессора, к сожалению, не сопровождается столь же высоким ростом скорости работы памяти.

Современные процессоры предъявляют определенные требования к системе памяти и относительно времени ожидания (задержки в доставке операнда), и относительно пропускной способности (количества данных, передаваемых в единицу времени). К сожалению, эти два аспекта системы памяти сильно расходятся. Обычно с увеличением пропускной способности увеличивается время ожидания.

С увеличением скорости задающего генератора становится все сложнее обеспечить такую систему памяти, которая может передавать операнды за один или два цикла. Один из способов решения этой проблемы — добавление кэш-памяти. кэш-память содержит наиболее часто используемые слова, что повышает скорость доступа к ним. Если достаточно большой процент нужных слов находится в кэш-памяти, время ожидания может сильно сократиться.

Одной из самых эффективных технологий одновременного увеличения пропускной способности и уменьшения времени ожидания является применение нескольких блоков кэш-памяти. Основная технология — введение отдельной кэш-памяти для команд и отдельной для данных **(разделенной** кэш-памяти).

В настоящее время многие системы памяти гораздо сложнее этих. Между разделенной кэш-памятью и основной памятью часто помещается **кэш-память второго уровня.** Вообще говоря, может быть три и более уровней кэш-памяти, поскольку требуются более продвинутые системы.

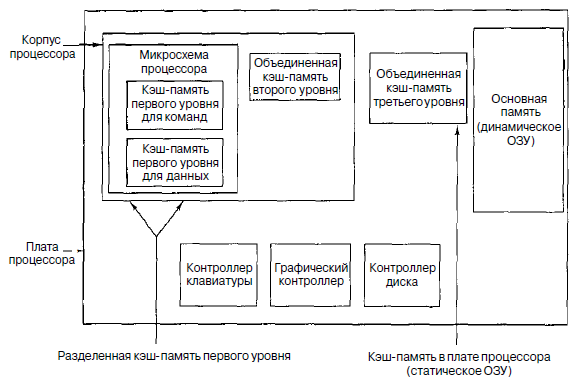


Рис. 4.25. Система с тремя уровнями кэш-памяти

Обычно все содержимое кэш-памяти первого уровня находится в кэш-памяти второго уровня, а все содержимое кэш-памяти второго уровня находится в кэш-памяти третьего уровня.

Существует два типа локализации адресов. Работа кэш-памяти зависит от этих типов локализации. **Пространственная локализация** основана на вероятности, что в скором времени появится потребность обратиться к ячейкам памяти, которые расположены рядом с недавно вызванными ячейками. Исходя из этого наблюдения в кэш-память переносится больше данных, чем требуется в данный момент. **Временная локализация** имеет место, когда недавно запрашиваемые ячейки запрашиваются снова. Это может происходить, например, с ячейками памяти, находящимися рядом с вершиной стека или с командами внутри цикла. Принцип временной локализации используется при выборе того, какие элементы выкинуть из кэш-памяти в случае промаха кэш-памяти. Обычно отбрасываются те элементы, к которым давно не было обращений.

Во всех типах кэш-памяти используется следующая модель. Основная память разделяется на блоки фиксированного размера, которые называются **строками кэш-памяти.** Строка кэш-памяти состоит из нескольких последовательных байтов (обыч-но от 4 до 64). Строки нумеруются, начиная с 0, то есть если размер строки составляет 32 байта, то строка 0 — это байты с 0 по 31, строка 1 — байты с 32 по 63 и т. д. В любой момент несколько строк находится в кэш-памяти. Когда происходит обращение к памяти, контроллер кэш-памяти проверяет, есть ли нужное слово в данный момент в кэш-памяти. Если есть, то можно сэкономить время, требуемое на доступ к основной памяти. Если данного слова в кэш-памяти нет, то какая-либо строка из нее удаляется, а вместо нее помещается нужная строка из основной памяти или из кэш-памяти более низкого уровня.

Кеш-память прямого отображения

Самый простой тип кэш-памяти — это **кэш-память прямого отображения.** Каждый элемент (ряд) может вмещать ровно одну строку из основной памяти.

Каждый элемент кэш-памяти состоит из трех частей:

1. Бит достоверности указывает, есть ли достоверные данные в элементе или нет. Когда система загружается, все элементы маркируются как недостоверные.
2. Поле «Тег» состоит из уникального 16-битного значения, указывающего соответствующую строку памяти, из которой поступили данные.
3. Поле «Данные» содержит копию данных памяти. Это поле вмещает одну строку кэш-памяти в 32 байта.

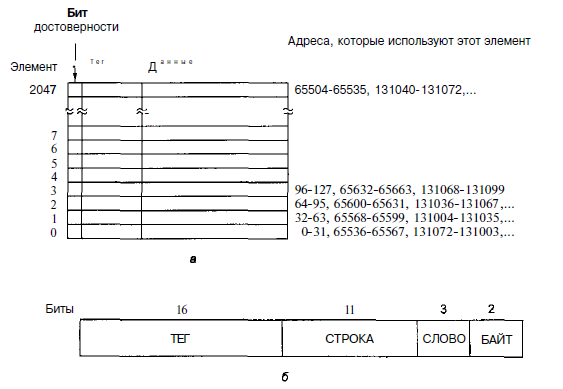


Рис. 4.26. (а) Кэш-память прямого оттображения; (б) 32-битный виртуальный адрес

В кэш-памяти прямого отображения данное слово может храниться только в одном месте. Если дан адрес слова, то в кэш-памяти его можно искать только в одном месте. Если его нет на этом определенном месте, значит, его вообще нет в кэш-памяти. Для хранения и удаления данных из кэш-памяти адрес разбивается на 4 компонента, как показано на рис. 4.26, *б:*

1. Поле «ТЕГ» соответствует битам, сохраненным в поле «Тег» элемента кэш-памяти.
2. Поле «СТРОКА» указывает, какой элемент кэш-памяти содержит соответствующие данные, если они есть в кэш-памяти.
3. Поле «СЛОВО» указывает, на какое слово в строке производится ссылка.
4. Поле «БАЙТ» обычно не используется, но если требуется только один байт, поле сообщает, какой именно байт в слове нужен. Для кэш-памяти, поддерживающей только 32-битные слова, это поле всегда будет содержать 0.

Когда центральный процессор выдает адрес памяти, аппаратное обеспечение выделяет из этого адреса 11 битов поля «СТРОКА» и использует их для поиска в кэш-памяти одного из 2048 элементов. Если этот элемент действителен, то производится сравнение поля «Тег» основной памяти и поля «Тег» кэш-памяти. Если поля равны, это значит, что в кэш-памяти есть слово, которое запрашивается. Такая ситуация называется **удачным обращением в кэш-память.** Вслучае удачного обращения слово берется прямо из кэш-памяти, и тогда не нужно обращаться к основной памяти. Из элемента кэш-памяти берется только нужное слово. Остальная часть элемента не используется. Если элемент кэш-памяти недействителен (недостоверен) или поля «Тег» не совпадают, то нужного слова нет в памяти. Такая ситуация называется **промахом кэш-памяти.** В этом случае 32-байтная строка вызывается из основной памяти и сохраняется в кэш-памяти, заменяя тот элемент, который там был. Однако если существующий элемент кэш-памяти изменяется, его нужно отписать обратно в основную память до того, как он будет отброшен.

Ассоциативная кеш память с множественным доступом

Различные строки основной памяти конкурируют за право занять одну и ту же область в кэш-памяти. Если программе, которая применяет кэш-память, изображенную на рис. 4.26, а, часто требуются слова с адресами 0 и 65 536, то будут иметь место постоянные конфликты и каждое обращение потенциально повлечет за собой вытеснение какой-то определенной строки кэш-памяти. Чтобы разрешить эту проблему, нужно сделать так, чтобы в каждом элементе кэш-памяти помещалось по две и более строк. Кэш-память с n возможными элементами для каждого адреса называется **n-входовой ассоциативной кэш-памятью.** Четырех-входовая ассоциативная кэш-память изображена на рис. 4.27.

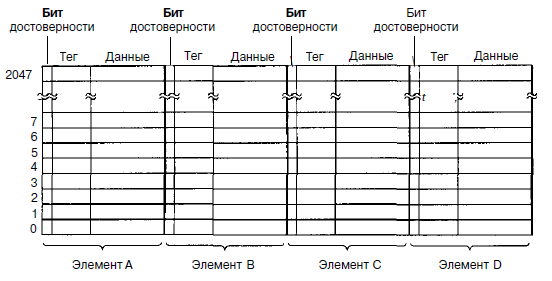


Рис. 4.27.Четырехвходовая ассоциативная кеш память

Ассоциативная кэш-память с множественным доступом по сути гораздо сложнее, чем кэш-память прямого отображения, поскольку хотя элемент кэш-памяти и можно вычислить из адреса основной памяти, требуется проверить п элементов кэш-памяти, чтобы узнать, есть ли там нужная нам строка. Тем не менее практика показывает, что двувходовая или четырехвходовая ассоциативная кэш-память дает хороший результат, поэтому внедрение этих дополнительных схем вполне оправданно.

Использование ассоциативной кэш-памяти с множественным доступом ставит разработчика перед выбором. Если нужно поместить новый элемент в кэш-память, какой именно из старых элементов нужно убрать? Для многих целей хорошо подходит алгоритм **LRU (Least Recenly Used — алгоритм удаления наиболее давно использовавшихся элементов).** Имеется определенный порядок каждого набора ячеек, которые могут быть доступны из данной ячейки памяти. Всякий раз, когда осуществляется доступ к любой строке, в соответствии с алгоритмом список обновляется и маркируется элемент, к которому произведено последнее обращение. Когда требуется заменить какой-нибудь элемент, убирается тот, который находится в конце списка, то есть тот, который использовался давно по сравнению со всеми другими.

Наконец, особой проблемой для кэш-памяти является запись. Когда процессор записывает слово, а это слово находится в кэш-памяти, он, очевидно, должен или обновить слово, или отбросить данный элемент кэш-памяти. Практически во всех разработках используется обновление кэш-памяти. Немедленное обновление элемента основной памяти называется **сквозной записью.** К сожалению, при этом требуется передавать больший поток информации к памяти, поэтому в более сложных проектах стремятся использовать альтернативный подход — **обратную** запись.

С процессом записи связана еще одна проблема: а что происходит, если нужно записать что-либо в ячейку, которая в текущий момент не находится в кэш-памяти? В большинстве разработок, в которых применяется обратная запись, данные переносятся в кэш-память. Эта технология называется **заполнением по записи** (write allocation). С другой стороны, в тех разработках, где применяется сквозная запись, обычно элемент в кэш-память при записи не помещается, поскольку эта возможность усложняет разработку. Заполнение по записи полезно только в том случае, если имеют место повторные записи в одно и то же слово или в разные слова в пределах одной строки кэш-памяти

## 28. **Примеры уровня микроархитектуры** (Микроархитектура процессоров Pentium 4, UltraSPARC III, 8051)

## Примеры уровня микроархитектуры

## В этом разделе в свете материала, изучаемого в этой главе, мы рассмотрим три

## современных процессора. Наше изложение будет кратким, поскольку компьютеры чрезвычайно сложны, содержат миллионы вентилей и у нас нет возможности

## давать подробное описание. Процессоры, предлагаемые в качестве примеров, те

## же, что и раньше, — Pentium 4, UltraSPARC 3 и 8051.

## Микроархитектура процессора Pentium 4.

## На первый взгляд Pentium 4 кажется вполне традиционной СISС-машиной с большим и громоздким набором команд, поддерживающим 8-, 16- и 32-разрядные целочисленные операции, а также 32- и 64-разрядные операции с плавающей точкой.

## В нем всего 8 доступных регистров, причем ни один из них не повторяет

## другие. Допустимая длина команд составляет 1-17 байт. В общем, налицо стандартная унаследованная архитектура, которая все делает не так.

## На самом же деле процессор Pentium 4 основан на современном надежном

## RISC-ядре с развитой конвейеризацией. Его тактовая частота уже очень высока, а в последующие годы, скорее всего, вырастет еще больше. Удивительно,

## как инженерам Intel на основе архаичной архитектуры удалось построить процессор, отвечающий всем современным требованиям. Итак, в этом подразделе мы рассмотрим микроархитектуру Pentium 4 и разберемся в принципах ее работы

## Обзор микроархитектуры NetBurst

## Микроархитектура Pentium 4, называемая NetBurst, ознаменовала собой решительный отход от принципов микроархитектуры Р6, использовавшейся в процессорах Рentium Pro, Pentium 2 и Pentium 3. Она дает определенное представление о том, на какой базе продукция Intel будет разрабатываться в течение нескольких ближайших лет. Примерная схема микроархитектуры Pentium 4 изображена на рис. 4.31. В определенной степени она соответствует рисунку 1.8.

## Микроархитектура процессора UltraSPARC 3 Cu.

## Серия процессоров UltraSPARC 3 Cu, произведенная компанией Sun – это реализация версии 9 архитектуры SPARC. На первый взгляд все модели очень похожи и различаются главным образом по производительности и цене. В то же время, на уровне микроархитектуры они существенно различаются. В этом разделе мы обсудим процессор UltraSPARC 3 Cu. Аббревиатура «Cu» в названии модели говорит о том, что проводники микросхемы сделаны из меди — в отличие от алюминиевых проводников, применявшихся в предыдущих моделях. Сопротивление меди ниже сопротивления алюминия, за счет этого провода становятся тоньше, а быстродействие — выше.

## — это 64-разрядная машина с 64-разрядными регистрами и 64-разрядным трактом данных, но в целях совместимости с машинами версии 8 (которые являются 32-разрядными) она может работать с 32-разрядными операндами, а программное обеспечение, написанное для 32-разрядных версий SPARC, изменять не нужно. Хотя во внутренней архитектуре машины

## 64 разряда, ширина шины памяти составляет 128 бит, аналогично процессору

## Pentium 2 с 32-разрядной архитектурой и 64-разрядной шиной памяти. В обоих случаях в одну систему устанавливаются шина и процессор разных поколений.

## В отличие от Pentium 4, процессор UltraSPARC изначально проектировался как полноценная RISC-система. Следовательно, необходимости в сложном механизме преобразования старых CISC - команд в микрооперации в данном случае не было. Команды ядра представляют собой готовые микрооперации. Ситуация несколько осложняется появлением в последние годы новых команд обработки графических и мультимедийных данных, для выполнения которых требуются.

## специальные устройства.

## Общий обзор системы UltraSPARC 3 Cu

## Структурная схема UltraSPARC 3 Cu представлена на рис. 4.33. В целом, она

## значительно проще микроархитектуры применяемой в системах Pentium 4, что объясняется менее изощренной архитектурой системы команд UltraSPARC. Тем не менее по некоторым базовым компонентам сходство с Pentium 4 прослеживается. В первую очередь, это обусловлено технологическими и экономическими факторами. К примеру, в период проектирования этих схем объем кэшей данных первого составлял от 8 до 16 Кбайт. Этому стандарту соответствовали обе рассматриваемые микросхемы. Когда производство кэш памяти первого уровня емкостью 64 Мбайт станет обоснованным с технологической и экономической точек зрения, все процессоры будут оснащаться уже новыми кэшами. Различия между Pentium 4 и UltraSPARC 3 Cu связаны по большей части с тем, что в первом случае разработчикам пришлось обеспечить поддержку унаследованного набора CISC-команд, а во втором такой задачи не ставилось.

## В левой верхней части рис. 4.33 изображен 4-входовый ассоциативный кэш команд емкостью 32 Кбайт с 32-байтными строками. Поскольку большинство команд UltraSPARC занимают 4 байта, в этом кэше можно одновременно разместить около 8000 команд. По этому признаку UltraSPARC 3 Cu несколько уступает кэшу трасс NetBurst.

## Микроархитектура процессора 8051

## Микроархитектура процессора 8051 значительно проще 2 предыдущих Pentium и UltraSPARC . Дело в том, что размер этой микросхемы очень мал (она состоит из 60000 транзисторов), а разрабатывалась она задолго до того, как конвейерная технология стала популярной. Кроме того, перед разработчиками 8051 ставилась задача создать дешевую, а не быструю микросхему. Как известно дешевый и простой – очень близкие понятия, в то время как дешевизна и быстродействие в нашем контексте редко сочетаются.

## Центральное положение в микроархитектуре 8051 занимает основная шина. С ней связаны немногочисленные регистры, причем для большинства из них операции чтения и записи выполняются программно. Регистр АСС представляет собой основной арифметический регистр, в котором сохраняется большая часть результатов вычислений. Через него проходят почти все арифметические команды. Регистр В применяется для умножения и деления; кроме того, при хранении результатов он выполняет роль временного регистра. Регистр SР является указателем

## стека, как и в большинстве других систем он указывает на вершину стека. В регистре команд IR содержатся команды, выполняемые в данный момент.

## Регистры ТМР1 и ТМР2 — это защелки, обслуживающие АЛУ. Перед выполнением операций в АЛУ соответствующие операнды копируются в эти защелки.

## Результаты вычислений в АЛУ записываются в любой регистр записи, доступ к

## которому обеспечивает основная шина. Коды состояний, обозначающие нулевые,

## отрицательные и тому подобные результаты, записываются в регистр PSW

## (Program Status Word — слово состояния программы).

## В 8051 предусмотрены независимые модули памяти для размещения данных

## и кола. Емкость ОЗУ для размещения данных составляет 128 (модель 8051) или 256 (модель 8052) байт; соответственно, 8-разрядного регистра RAM ADDR вполне достаточно для полной адресации этой памяти. В процессе адресации ОЗУ адрес целевого байта размещается в регистре RAM ADDR, после чего производится обращение к памяти. Емкость памяти кода может достигать 64 Кбайт

## (при условии размещения модуля памяти вне микросхемы), поэтому разрядность регистра RОМ АDDR составляет 16 бит. Схема адресации к программному коду в ПЗУ с помощью регистра ВОМ АРОК аналогична вышеописанной схеме для памяти данных.

## Регистр ОРТВ (РоиЫе РошТеВ — двойной указатель) — это 16-разрядный временный регистр, предназначенный для управления и сборки 16-разрядных адресов. Регистр РС представляет собой 16-разрядный счетчик команд; иными словами, он определяет адрес следующей команды, которую требуется вызвать и выполнить. Регистр РС шсгетещег — это специальный аппаратный модуль, выполняющий роль псевдорегистра. Когда в него копируется, а затем считывается содержимое регистра РС, его значение автоматически увеличивается на единицу. Ни к РС, ни к РС {шсгешещег нельзя обратиться через основную шину. Наконец, ВОЕЕЕВ — это еще один 16-разрядный временный регистр. На самом деле каждый 16-разрядный регистр процессора 8051 состоит из пары 8-разрядных регистров, с каждым из которых можно выполнять разные операции.

## В дополнение ко всему, на микросхеме 8051 устанавливаются три 16-разрядных таймера, необходимых для выполнения приложений в реальном времени. Также предусмотрено четыре 8-разрядных порта ввода-вывода, через которые процессор 8051 может управлять 32 внешними кнопками, световыми индикаторами, датчиками, выключателями и т.д. Именно наличие таймеров и портов ввода-вывода делает возможным применение 8051 в качестве встроенного процессора без установки дополнительных микросхем.

## Процессор 8051 относится к категории синхронных — большинство ко-

## манд, которые он обрабатывает, завершаются за один цикл. Каждый цикл

## делится на шесть частей, называемых состояниями. В первом состоянии следующая команда вызывается из ПЗУ, и по основной шине отправляется в реское изображение такого конвейера. Для современной технологии это наиболее эффективная система.

## Главное различие между Рentium 4 и UltraSPARC 3 состоит в том, как передаются ISА-команды функциональному блоку. Компьютеру Pentium 4 приходится разбивать СISС-команды, чтобы преобразовать их в З-регистровый формат, необходимый для функционального блока. Именно этот процесс показан на рис. 4.32 — разбиение больших команд на маленькие микрооперации. Машине UltraSPARC 3 не нужно ничего делать, поскольку ее исходные команды уже.

## представляют собой удобные и компактные микрооперации. Вот почему большинство новых архитектур ISA относятся к типу RISC, который обеспечивает оптимальное сочетание набора команд и внутреннего механизма их выполнения.

## Полезно сравнить нашу последнюю разработку, микроархитектуру Мк-4, с этими тремя реальными машинами. Мic-4 больше всего напоминает Pentium 4 Обе системы интерпретируют команды, не являющиеся RISC-командами. Для этого обе системы разбивают команды на микрооперации, в которых указаны код операции, два входных и один выходном регистр. В обоих случаях микрооперации помещаются в очередь для дальнейшего выполнения. В Мiс-4 микро-

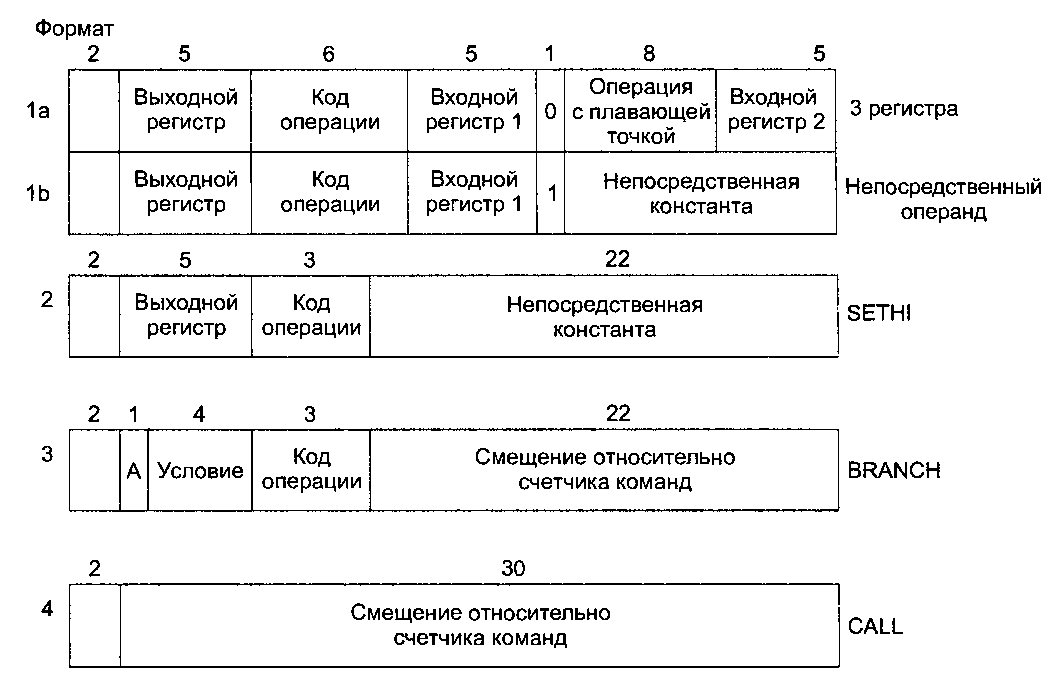
## операции запускаются строго по порядку, выполняются строго по порядку и завершаются тоже строго по порядку. В Pentium 4 микрооперации запускаются по порядку, выполняются в произвольном порядке, а завершаются опять-таки по порядку.

## 

## 

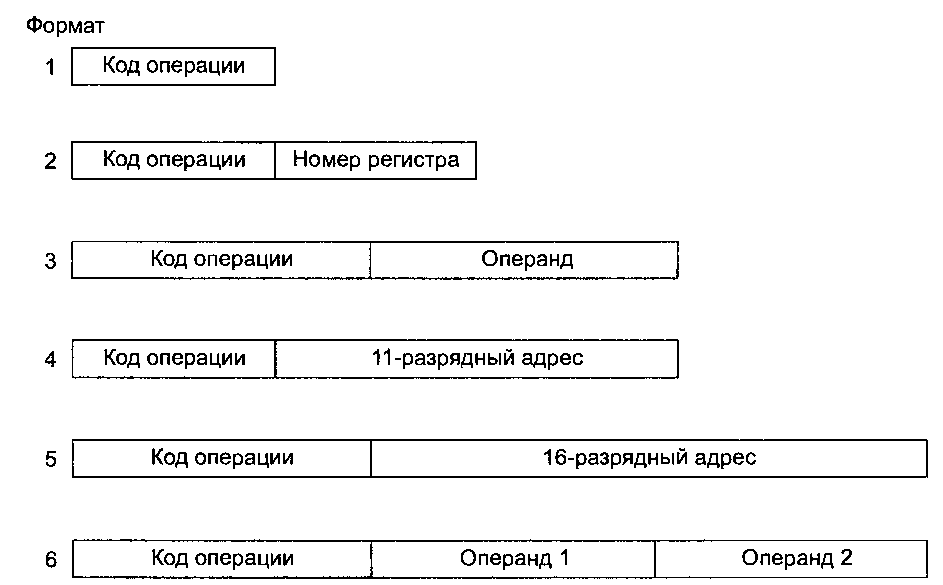
## 29. **Форматы команд Pentium 4, UltraSPARC III, микросхемы 8051 «земля тебе пухом»**

Pentium 4 Они содержат до шести полей разной длины, пять из которых не обязательны. Такая ситуация сложилась из-за того, что архитектура развивалась на протяжении нескольких поколений, появлялось всё больше и больше различных команд, и не все были удачны, но из-за требования обратной совместимости позднее их нельзя было изменить. Таким образом выходит, что чем более редкая команда, тем большем памяти она потребует задействовать.

С UltraSPARC III значительной части конвейеризи-рованных машин сначала выполняется команда,следующая после команды безусловного перехода, хотя по логике вещей так бытьне должно. Позиция после перехода называется слотом отсрочки (delay slot). Первые 2 бита каждой команды помогают определить формат команды и сообщают аппаратному обеспечению, где найти оставшуюся часть кода операции, если она есть. 

В 8051 предусмотрено шесть простых форматов команд (рис. 5.12). Размер команд может быть равен 1, 2 или 3 байтам. Вариант 1 формата предусматривает наличие в команде только кода операции. Такова, к примеру, команда инкремента сумматора.

Вариант 2 формата также состоит из одного байта, но из 8 бит в нем 5 выделяется на код операции, оставшиеся 3 - на номер регистра. По этому формату строятся команды, которые одновременно обращаются к сумматору и регистру, например команды добавления значения из регистра в сумматор или переноса данных из сумматора в регистр.



Команда, построенная по варианту 3 формата, состоит из операнда размером 1 байт. В качестве операнда может, в частности, выступать непосредственная константа (загружаемая в сумматор), смещение (например расстояние перехода) или номер бита (в целях установки, сброса или проверки бита п).

Варианты 4 и 5 формата предназначены для команд переходов и вызовов подпрограмм. 11-разрядные адреса применяются в отсутствие внешней памяти, когда длина адреса не превышает 4096 (в модели 8051) или 8192 (в модели 8052). Если внешняя память присутствует и ее объем составляет более 8 Кбайт, применяются 16-разрядные адреса.

Команда, построенная по варианту 6 формата, содержит два 8-разрядных операнда. Этот формат характерен для многих команд, например, для команды переноса 8-разрядной непосредственной константы по адресу встроенной в микросхему памяти.

24

**Метод коммутации пакетов с ожиданием**

Прежде чем начать подробное рассмотрение сетевого уровня, необходимо восстано-

вить в памяти окружение, в котором ему приходится функционировать. Оно показано

на рис ниже. Основными компонентами сети являются устройства интернет-провай-

дера (маршрутизаторы, соединенные линиями связи), показанные внутри затенен-

ного овала, а также устройства, принадлежащие клиенту и показанные вне овала.

Хост H1 напрямую соединен с одним из маршрутизаторов интернет-провайдера А

(как, например, домашний компьютер, подключенный к DSL-модему). Хост H2, на-

против, находится в ЛВС (например, офисной сети Ethernet) с маршрутизатором F,

принадлежащим клиенту, который с ним работает. Этот маршрутизатор связывается

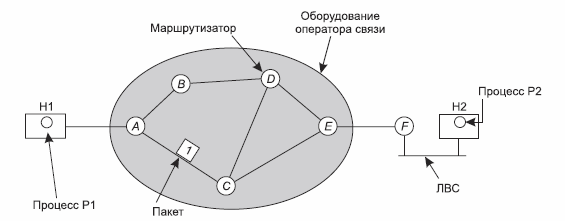
с интернет-провайдером по выделенной линии. Мы показали F вне овала, потому что

он не принадлежит интернет-провайдеру. Однако в контексте данной главы мы будем

считать маршрутизаторы клиента частью сети интернет-провайдера, поскольку в них

применяются те же самые алгоритмы, что и в маршрутизаторах интернет-провайдеров

(а основным предметом рассмотрения будут именно алгоритмы).



Система работает следующим образом. Хост, у которого есть пакет для передачи,

посылает его на ближайший маршрутизатор либо в своей ЛВС, либо по двухточеч-

ному соединению интернет-провайдеру. Там пакет хранится до тех пор, пока не будет

принят целиком и не пройдет полную обработку, включая верификацию контрольной

суммы. Затем он передается по цепочке маршрутизаторов, которая в итоге приводит

к пункту назначения. Такой механизм называется коммутацией пакетов с ожиданием

(store-and-forward), и мы уже рассматривали его в предыдущих главах.

**Сервисы, предоставляемые**

**транспортному уровню**

Сетевой уровень предоставляет транспортному уровню сервисы через интерфейс меж-

ду сетевым и транспортным уровнями. Важным вопросом является именно то, какой

именно вид сервиса сетевой уровень предоставляет транспортному. Разработка таких

сервисов требует особой аккуратности и при этом необходимо учитывать следующее:

1 Сервисы сетевого уровня не должны зависеть от технологии маршрутизатора.

2 Транспортный уровень должен быть независим от количества, типа и топологии

присутствующих подсетей с маршрутизаторами.

3 Сетевые адреса, доступные транспортному уровню, должны использовать единую

систему нумерации, даже между локальными и глобальными сетями..

386 Глава 5. Сетевой уровень

Находясь в рамках поставленной перед ними задачи, разработчики оказываются

абсолютно свободными в написании детальной спецификации сервисов, которые

должны предоставляться транспортному уровню. Эта свобода часто вырождается

в яростную борьбу между двумя непримиримыми группировками. В центре дискуссии

оказывается вопрос о том, какие сервисы должен предоставлять сетевой уровень —

ориентированные на соединение или не требующие соединений.

Один лагерь (представленный интернет-сообществом) заявляет, что работа марш-

рутизаторов заключается исключительно в перемещении с места на место пакетов,

и больше ни в чем. С этой точки зрения (основанной на примерно сорокалетнем опыте

работы с реальными компьютерными сетями), сеть обладает врожденной ненадеж-

ностью, вне зависимости от того, как она спроектирована. Хосты должны учитывать

это и защищаться от ошибок своими силами (то есть заниматься обнаружением и ис-

правлением ошибок), а также самостоятельно управлять потоком.

Из этого следует, что сетевой сервис должен быть сервисом, не требующим

Установки соединения и состоящим в основном из примитивов SEND PACKET (послать

пакет) и RECEIVE PACKET (принять пакет). В частности, сюда нельзя включать упорядочивание

пакетов и контроль потока — все равно эти действия будет выполнять

хост. От того что одна и та же работа будет выполнена дважды, качество обслуживания

не повысится. Такое рассуждение — пример применения «сквозного» принципа (endto-

end argument), оказавшего значительное влияние на формирование Интернета

(Saltzer и др., 1984). Кроме того, каждый пакет должен содержать полный адрес по-

лучателя, так как пересылка производится независимо от предшествующих пакетов.

Другой лагерь, представленный телефонными компаниями, утверждает, что сеть

должна предоставлять надежный, ориентированный на соединение сервис. Они

утверждают, что 100 лет успешного управления телефонными системами по всему

миру — это серьезный аргумент в их пользу. По их мнению, качество обслуживания яв-

ляется определяющим фактором, и без установления соединения в сети очень сложно

добиться каких-либо приемлемых результатов, особенно когда дело касается трафика

реального масштаба времени — например, передачи голоса и видео.

Этот спор остается актуальным даже по прошествии нескольких десятков лет.

Раньше самые распространенные сети передачи данных (такие как.25, использовав-

шаяся в 70-х, и популярная в 80-х Frame Relay) были ориентированы на соединение.

Однако после появления ARPANET и на ранних этапах развития Интернета сетевые

уровни без установления соединения стали чрезвычайно популярными. Сейчас прото-

кол IP является вездесущим символом успеха. На его популярность не повлияло даже

появление ориентированной на соединение технологии ATM, созданной в 80-х годах

с целью заменить IP: в настоящее время ATM используется только в отдельных

случаях, тогда как в ведении IP оказываются все телефонные сети. Однако с ростом

требований к качеству обслуживания развитие Интернета предполагает появление

новых возможностей, ориентированных на соединение. В качестве примеров таких

технологий можно привести MPLS (MultiProtocolLabelSwitching, «мультипротоколь-

ная коммутация по меткам»), о которой мы поговорим в этой главе, и VLAN, которую

мы рассматривали в главе 4. Обе эти технологии сейчас широко используются.

## 30. **Виртуальные команды ввода-вывода**

Наборы команд на уровне архитектуры команд и на уровне микроархитектуры совершенно разные. Различаются не только сами команды, но и их форматы, а некоторые совпадения совершенно случайны. Набор команд уровня операционной системы содержит большую часть команд уровня архитектуры команд, а также несколько новых очень важных команд. В то же время некоторые новые, но важные команды на уровне операционной системы не поддерживаются. Ввод-вывод — это одна из областей, в которых эти два уровня отличаются очень значительно. Причина таких различий проста. Во-первых, пользователь, способный выполнять команды ввода-вывода уровня архитектуры команд, может считать конфиденциальную информацию, хранящуюся где-нибудь в «дебрях» операционной системы, что потенциально представляет для системы угрозу. Во-вторых, нормальный программист совершенно не горит желанием реализовать ввод-вывод на уровне архитектуры команд, поскольку это слишком сложно и утомительно. Вместо этого для ввода-вывода можно просто установить определенные поля и биты в ряде регистров устройств, затем подождать, пока операция закончится, и проверить, что произошло.

Один из способов виртуального ввода-вывода — использование абстракции под названием файл. Файл состоит из последовательности байтов, записанных на устройство ввода-вывода. Если устройство ввода-вывода является устройством хранения информации (например, диском), то файл можно считать обратно. Если устройство не является устройством хранения информации (например, это принтер), то файл оттуда считать нельзя. На диске может храниться множество файлов, в каждом из которых содержатся данные определенного типа, например изображение, электронная таблица или текст. Файлы имеют разную длину и обладают разными свойствами. Эта абстракция позволяет легко организовать виртуальный ввод-вывод.

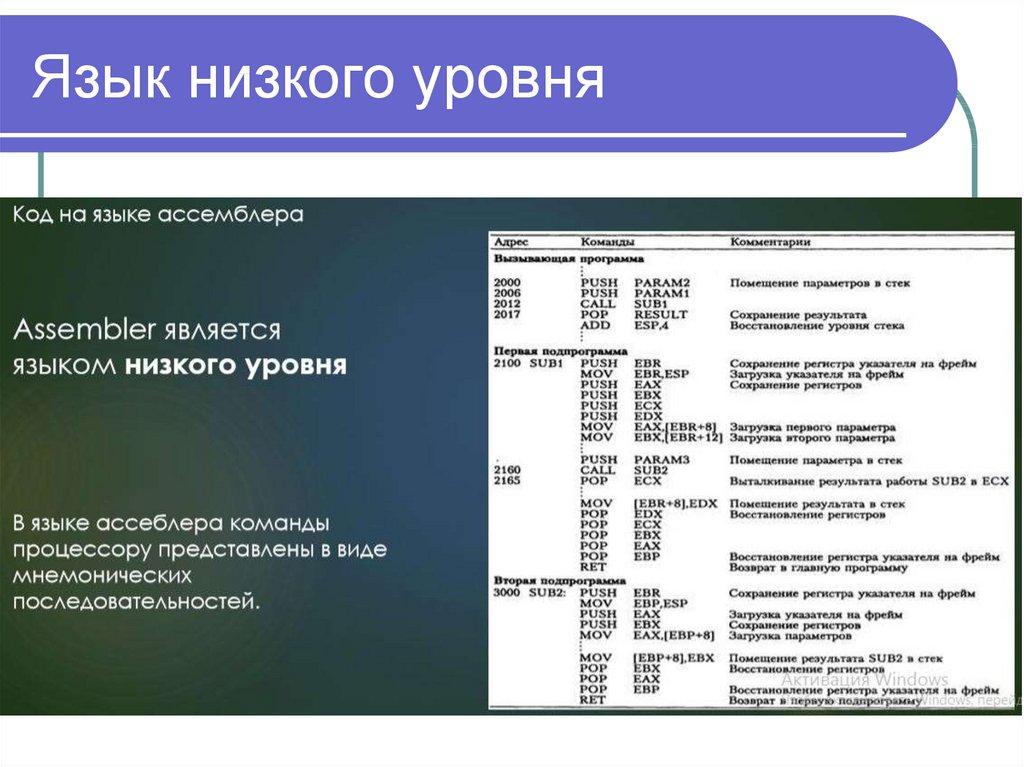
Для операционной системы файл является просто последовательностью байтов. Вся структуризация определяется исключительно на уровне прикладных программ. Ввод-вывод осуществляется путем системных вызовов открытия, чтения, записи и закрытия файлов. Процесс открытия файла позволяет операционной системе найти файл на диске и передать в память информацию, необходимую для доступа к этому файлу. После открытия файла из него можно читать данные. Системный вызов для считывания должен иметь, как минимум, следующие параметры:

информацию о том, какой именно открытый файл нужно считывать; указатель на буфер в памяти, в который нужно поместить данные; число считываемых байтов. Данный системный вызов помещает требующиеся данные в буфер. Обычно он возвращает число считанных байтов. Это число может быть меньше запрошенного числа (например, нельзя считать 2000 байт из файла размером 1000 байт). С каждым открытым файлом связан указатель, который сообщает, какой байт должен считываться следующим. После команды read указатель дополняется числом считанных байтов, поэтому последовательные команды read считывают последовательные блоки данных из файла. Обычно этот указатель можно установить на особое значение, чтобы программы могли получить доступ к любой части файла. Когда программа заканчивает считывание файла, она может закрыть его и сообщить операционной системе, что больше не будет использовать этот файл. Тогда операционная система сможет освободить пространство в структуре, в которой хранилась информация об этом файле.

Основная виртуальная команда ввода считывает следующую запись из нужного файла и помещает ее в основную память, начиная с определенного адреса. Чтобы выполнить эту операцию, виртуальная команда должна получить сведения о том, какой файл считывать и куда в памяти поместить запись. Часто существуют параметры для чтения некоторой записи, которые определяются либо по ее месту в файле, либо по ее ключу. Основная виртуальная команда вывода записывает логическую запись из памяти в файл. Последовательные команды write выполняют последовательные логические записи в файл. Чтобы понять, как реализуются виртуальные команды ввода-вывода, нужно хорошо понимать механизмы структурирования и хранения файлов. Основная проблема, которая должна быть решена для всех файловых систем, — распределение пространства. Единицей распределения (также называемой «блоком») может быть один сектор на диске, но чаще блок состоит из нескольких последовательных секторов.

Оперативная информация хранится в файлах. Программы могут получить доступ к ней через программы ввода-вывода.

## 31. **Уровень ассемблера. Формат оператора в ассемблере. Директивы**

Ассемблер — транслятор исходного текста программы, написанной на языке ассемблера, в программу на машинном языке. Язык ассемблера (англ, assembly language) — машинно-ориентированный язык низкого уровня с командами, не всегда соответствующими командам машины, который может обеспечить дополнительные возможности вроде макрокоманд; автокод, расширенный конструкциями языков программирования высокого уровня, такими как выражения, макрокоманды, средства обеспечения модульности программ (Машинный код). Для понимания: 

(Формат) В Ассемблере для выполнения команд используются операнды. Вводится номер действия, адрес, команда, и объект.

Директивы Ассемблера

**Таблица 3.5 Список директив ассемблера**

|  |  |
| --- | --- |
| Директива | Описание |
| BYTE | Зарезервировать байты в ОЗУ |
| CSEG | Программный сегмент |
| DB | Определить байты во флэш или EEPROM |
| DEF | Назначить регистру символическое имя |
| DEVICE | Определить устройство для которого компилируется программа |
| DSEG | Сегмент данных |
| DW | Определить слова во флэш или EEPROM |
| ENDM | Конец макроса |
| EQU | Установить постоянное выражение |
| ESEG | Сегмент EEPROM |
| EXIT | Выйти из файла |
| INCLUDE | Вложить другой файл |
| LIST | Включить генерацию листинга |
| LISTMAC | Включить разворачивание макросов в листинге |
| MACRO | Начало макроса |
| NOLIST | Выключить генерацию листинга |
| ORG | Установить положение в сегменте |
| SET | Установить переменный символический эквивалент выражения |

## 32. **Cравнение операционных систем UNIX и Windows**

## UNIX.

## Все, с чем работает система UNIX, она воспринимает в виде файла. Это не такое уж упрощение, как может показаться на первый взгляд. Когда разрабатывалась первая версия системы, даже прежде, чем ей дали имя, все усилия сосредоточились на создании структуры файловой системы, которая должна была быть простой и удобной в использовании. Файловая система – ключевое звено, обеспечивающее успешное применение UNIX. Это наилучший пример философии «прекрасное в малом», показывающий, какой мощи можно достичь реализацией нескольких хорошо продуманных идей.

## Файл представляет собой последовательность байтов. Никаких ограничений по структуре системой на файл не накладывается, и никакого смысла не приписывается его содержимому: смысл байтов зависит исключительно от программ, обрабатывающих файл.

## Подход UNIX к представлению управляющей информации нетрадиционен, особенно использование символа перевода строки для завершения строки. Многие системы вместо этого трактуют каждую строку как «запись», содержащую не только введенные данные, но и счетчик числа символов в строке. В других системах каждая строка завершается символами возврата каретки и перевода строки, поскольку такая последовательность необходима для вывода на большинство терминалов.

## UNIX не делает ни того, ни другого: нет записей и счетчиков, к тому же ни в одном файле нет никаких байтов, которые бы вы или ваша программа не поместили туда. Символ перевода строки преобразуется в два символа – возврата каретки и перевода строки при выводе на терминал, но программы должны иметь дело с одним символом перевода строки, поскольку это все, что они могут «увидеть». В большинстве случаев подобная простая схема является оптимальной. Если необходима более сложная структура, ее легко построить на базе этой, тогда как получить простое из сложного значительно трудней. В отношении каждого файла существуют права доступа, которые определяют, кто и что может делать с файлом. Если вы храните какие-то письма и не хотите, чтобы кто-то их другой их читал, вы можете изменить права доступа к каждому письму. Вы можете, наконец, просто изменить права доступа к каталогу, содержащему письма, и пресечь любые попытки вмешаться в ваши документы в ваши документы. Но в UNIX есть особый пользователь, называемый суперпользователем, который может читать или изменять любой файл в системе. Привилегии входа в систему суперпользователю обеспечивает специальное имя root (корень). Существует команда su, гарантирующая вам статус суперпользователя при условии, что вы знаете пароль при входе под именем root.

## Во многом централизованные файловые системы удобнее изолированных (система управления файлами принимает на себя больше рутинной работы). Но в таких системах возникают существенные проблемы, если кому-то требуется перенести поддерево файловой системы на другую вычислительную установку. Компромиссное решение применено в файловых системах ОС UNIX. На базовом уровне в этих файловых системах поддерживаются изолированные архивы файлов. Один из этих архивов объявляется корневой файловой системой. После запуска системы можно "смонтировать" корневую файловую систему и ряд изолированных файловых систем в одну общую файловую систему. Технически это производится с помощью заведения в корневой файловой системе специальных пустых каталогов. Специальный системный

## вызов курьер ОС UNIX позволяет подключить к одному из этих пустых каталогов корневой каталог указанного архива файлов. После монтирования общей файловой системы именование файлов производится так же, как если бы она с самого начала была централизованной. Если учесть, что обычно монтирование файловой системы производится при раскрутке системы, то пользователи ОС UNIX обычно и не задумываются об исходном происхождении общей файловой системы.

## Поскольку файловые системы являются общим хранилищем файлов, принадлежащих, вообще говоря, разным пользователям, системы управления файлами должны обеспечивать авторизацию доступа к файлам. В общем виде подход состоит в том, что по отношению к каждому зарегистрированному пользователю данной вычислительной системы для каждого существующего файла указываются действия, которые разрешены или запрещены данному пользователю. Существовали попытки реализовать этот подход в полном объеме. Но это вызывало слишком большие накладные расходы, как по хранению избыточной информации, так и по использованию этой информации для контроля правомочности доступа.

## Поэтому в большинстве современных систем управления файлами применяется подход к защите файлов, впервые реализованный в ОС UNIX. В этой системе каждому зарегистрированному пользователю соответствует пара целочисленных идентификаторов: идентификатор группы, к которой относится этот пользователь, и его собственный идентификатор в группе. Соответственно, при каждом файле хранится полный идентификатор пользователя, который создал этот файл, и отмечается, какие действия с файлом может производить он сам, какие действия с файлом доступны для других пользователей той же группы, и что могут делать с файлом пользователи других групп. Эта информация очень компактна, при проверке требуется небольшое количество действий, и этот способ контроля доступа удовлетворителен в большинстве случаев.

## Последнее, на чем мы остановимся в связи с файлами, - это способы их использования в многопользовательской среде. Если операционная система поддерживает многопользовательский режим, вполне реальна ситуация, когда два или более пользователей одновременно пытаются работать с одним и тем же файлом. Если все эти пользователи собираются только читать файл, ничего страшного не произойдет. Но если хотя бы один из них будет изменять файл, для корректной работы этой группы требуется взаимная синхронизация.

## ОС UNIX работает на основе файловой системы FAT 16.

## Windows.

## Работа с операционной системой состоит в работе с ее объектами. Основными объектами являются файлы и папки.

## Информация на дисках хранится в виде файлов. Файл - это набор данных, записанных на диске, и имеющий уникальное имя. Содержимым файла может быть документ, ведомость, произвольный текст, программа, таблица чисел, рисунок и т.д. По имени файла компьютер определяет, где файл находится, какая информация в нем содержится и какими программами ее можно обработать.

## Имя файла состоит из двух частей: самого имени и расширения, разделенных точкой, например: mycomp.txt. Имя файла задает человек, который этот файл создает. Система

## Windows позволяет давать файлам «длинные» имена, в которые могут входить до 255 символов. При задании имени файла можно использовать любые символы, имеющиеся на клавиатуре, кроме некоторых «специальных» символов ( \ / : \* ? “ < > | ), использующихся в особых случаях.

## Расширения используют стандартные. По расширению можно узнать, к какому типу относится файл, и какая информация в нем содержится. Например, все файлы программ имеют расширение либо .EXE, либо .COM, текстовые файлы – .TXT или .DOC. В среднем количество файлов на дисках на современном компьютере составляет десятки тысяч. Для того чтобы свободно ориентироваться в таком количестве объектов, их надо упорядочить. Способ размещения файлов на дисках называют файловой структурой.

## Для организации файлов в Windows принята иерархическая структура. В иерархической структуре каждый элемент определяется путем, который ведет от вершины (корня) структуры к элементу.

## В основе этой структуры лежит конкретный диск (гибкий, жесткий или лазерный). Диски обозначаются буквами A:, B:, C:, D и так далее. Каждый отдельный диск имеет собственную файловую структуру.

## Диск разделен на папки. У каждой папки есть имя. Самая верхняя папка называется «корневой». Ее имя совпадает с именем диска.

## В папках могут храниться другие папки (вложенные) и файлы. Адрес конкретного файла определяется путем поиска данного файла. Путь поиска начинается с указания диска, за которым следуют имена всех папок, ведущих к файлу. Имена папок отделяются друг от друга символом «\». Например, файл документа с именем Письмо 133 может иметь такой адрес: С:\ Мои документы\ Переписка\ Письмо 133.

## Рабочий стол и окна папок предоставляют достаточно средств для работы с файлами и папками. Их можно создавать, открывать, перемещать, копировать, удалять и др. Большинство операций можно выполнить несколькими способами. Эффективная работа с компьютером предполагает знание разных способов и умелое применение наиболее удобных.

## Операционные системы Windows нельзя представить без файловых систем NTFS и FAT 32.

## NTFS – одна из самых сложных и удачных из существующих на данный момент файловых систем. Как и любая другая система NTFS делит все полезное место на кластеры – блоки данных, используемые единовременно; NTFS поддерживает почти любые размеры кластеров – от 512 байт до 64 кбайт, неким стандартом же считается 4 кбайт. Диск NTFS условно делится на две части. Первые 12 % диска под так называемую MFT (Master File Table) зону – пространство, в которое растет метафайл MFT. Запись каких-либо данных в эту область невозможна. MFT-зона всегда держится пустой – это делается для того, чтобы самый главный служебный файл (MFT) не фрагментировался на при своем росте. Остальные 88 % диска представляют собой пространство для хранения файлов.

## Каталог на NTFS представляет собой специфический файл, хранящий ссылки на другие файлы и каталоги, создавая иерархическое строение данных на диске. Файл

## каталога поделен на блоки, каждый из которых поддерживает имя файла, базовые атрибуты и ссылку на элемент MFT, который уже предоставляет полную информацию об элементе каталога.

## FAT - является самой распространенной файловой системой и поддерживается подавляющим большинством операционных систем. Система FAT32 - более новая файловая система на основе формата FAT, она поддерживается Windows 95 OSR2, Windows 98 и Windows Millennium Edition. FAT32 использует 32-разрядные идентификаторы кластеров, но при этом резервирует старшие 4 бита, так что эффективный размер идентификатора кластера составляет 28 бит. Поскольку максимальный размер кластеров FAT32 равен 32 Кбайт, теоретически FAT32 может работать с 8-терабайтными томами. Файловая система FAT32 в Windows 98 используется в качестве основной. С этой операционной системой поставляется специальная программа преобразования диска из FAT 16 в FAT 32. Windows NT и Windows 2000 тоже могут использовать файловую систему FAT, и поэтому можно загрузить компьютер с DOS-диска и иметь полный доступ ко всем файлам. Однако некоторые из самых прогрессивных возможностей Windows NT и Windows 2000 обеспечиваются ее собственной файловой системой NTFS (NT File System). Возможности файловых систем ААТ 32 гораздо шире возможностей FAT16..

## Сравнение организации файлов в Windows и UNIX.

## Организация файлов в операционной системе UNIX более простая и удобна, но она не дает нам полной уверенности в защите файлов, а организация файлов в операционной системе Windows более сложная, но и более защищенная. Сравнить организацию файлов в этих двух ОС можно по их файловым системам. Например, можно сравнить FAT16 (UNIX) и FAT32 (Windows):

## 

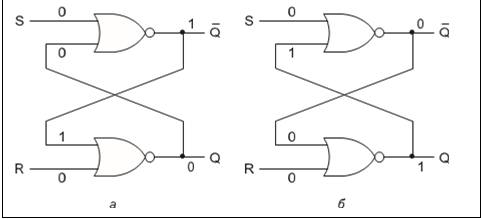
## Но в ОС Windows существует другая, более лучшая файловая система (NTFS). Поэтому можно сказать, что организация файлов в Windows лучше, чем в UNIX. Да и нельзя не заметить, что во всем мире более успешна операционная система Windows, что дает нам возможность сказать, что организация файлов в Windows лучше.

## 

## 33. **Цифровой логический уровень:Защелки, Компаратор, мультиплексор, Мультиплексоры, тригеры.**

Защелки.

Чтобы создать один бит памяти, нужна схема, которая каким-то образом «запоминает» предыдущие входные значения. Подобная схема выглядит следующим образом:



Данная схема называется SR зашелкой. Сконструирована она из двух вентилей НЕ – ИЛИ. Имеет два входа: Q и Q. В отличие от комбинаторной схемы, выходные сигналы защелки не определяются текущими входными сигналами.

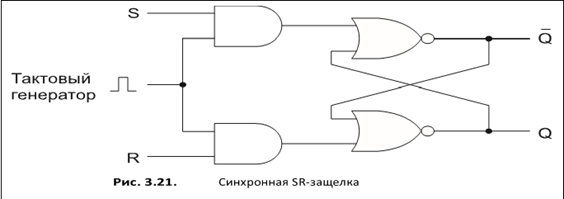
Чтобы понять, как работает защелка, предположим, что S = 0 и R = 0 (вообще сигнал на этих входах равен 0 большую часть времени). Предположим также, что Q = 0. Так как Q возвращается в верхний вентиль НЕ-ИЛИ и оба входа этого вентиля равны 0, то его выход, Q, равен 1. Единица возвращается в нижний вентиль, у которого в итоге один вход равен 0, другой — 1, а на выходе получается Q = 0. Такое состояние по крайней мере логически последовательно (см. рисунок а).

А теперь давайте представим, что Q = 1, а R и S все еще равны 0. Верхний вентиль имеет входы 0 и 1 и выход Q (то есть 0), который возвращается в нижний вентиль. Такое состояние, изображенное на рис. б, также логически последовательно. Положение, когда оба выхода равны 0, не является логически последовательным, поскольку в этом случае оба вентиля имели бы на входе два нуля, что привело бы к единице на выходе, а не к нулю. Точно так же невозможно иметь оба выхода равные 1, поскольку это привело бы к входным сигналам 0 и 1, что вызывает на выходе 0, а не 1.

Используя подобные рассуждения, легко увидеть, что установка S в значение 1 при состоянии защелки 1 (то есть при Q = 1) не вызывает изменений, но установка R в значение 1 приводит к изменению состояния защелки. Таким образом, если S принимает значение 1, то Q равняется 1 независимо от предыдущего состояния защелки. Сходным образом переход R в значение 1 вызывает Q = 0. Схема «запоминает», какой сигнал был последним: S или R. Используя это свойство, мы можем строить компьютерную память.

Синхронные SR-защелки

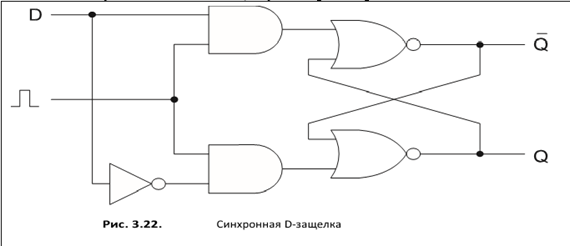
Часто бывает удобно, чтобы состояние защелки могло изменяться только в определенные моменты. Чтобы достичь этой цели, немного изменим основную схему и получим синхронную SR-защелку, рисунок:



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

Синхронные D-защелки

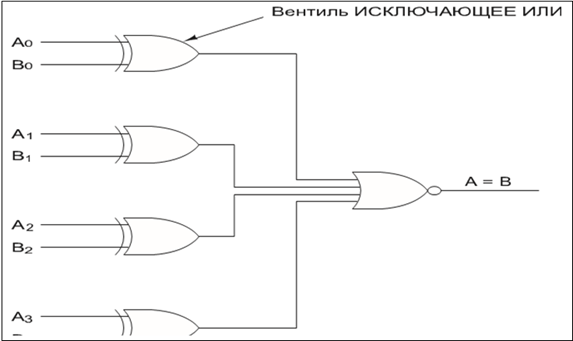
Чтобы разрешить ситуацию с неопределенностью SR-защелки (неопределенность возникает в случае, если S = R = 1), нужно предотвратить ее возникновение.



На рисунке изображена схема защелки только с одним входом D. Так как входной сигнал в нижний вентиль И всегда является обратным кодом входного сигнала в верхний вентиль И, ситуация, когда оба входа равны 1, никогда не возникает.

Компоратор

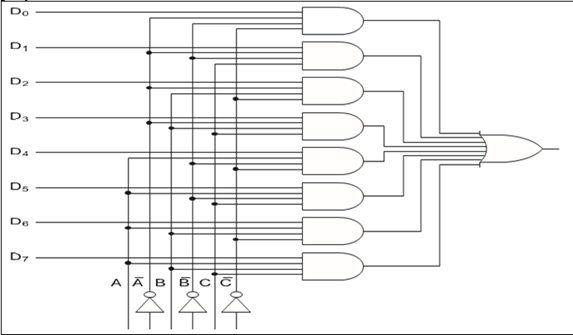
Компаратор сравнивает два слова, которые поступают на вход. принимает два входных сигнала A и B по 4 бита каждый и выдает 1, если они равны, и 0, если они не равны. Схема основывается на вентиле ИСКЛЮЧАЮЩЕЕ ИЛИ, который выдает 0, если сигналы на входе равны, и 1, если сигналы на входе не равны. Если все четыре входных слова равны, все четыре вентиля ИСКЛЮЧАЮЩЕЕ ИЛИ должны выдавать 0.



1 означает равенство, 0 — неравенство.

Мультиплексоры

На цифровом логическом уровне мультиплексор представляет собой схему с 2n входами, одним выходом и n линиями управления, которые позволяют выбрать один из входов. Выбранный вход соединяется с выходом. На рисунке ниже изображена схема восьмивходового мультиплексора. Три линии управления A, B и C кодируют 3-разрядное число, которое указывает, какая из восьми входных линий должна соединяться с вентилем ИЛИ и, следовательно, с выходом. Вне зависимости от того, какое значение окажется на линиях управления, семь вентилей И всегда будут выдавать на выходе 0, а оставшийся может выдавать 0 или 1 в зависимости от значения выбранной линии входа. Каждый вентиль И запускается определенной комбинацией сигналов на линиях управления. Схема мультиплексора показана на рисунке:



Его можно использовать в качестве преобразователя параллельного кода в последовательный. Если подать 8 бит данных на входные линии, а затем поочередно переключать линии управления, чтобы получить значения от 000 до 111 (это двоичные числа), то 8 бит поступят на выходную линию последовательно. Обычно такое преобразование осуществляется при вводе информации с клавиатуры, поскольку каждое нажатие клавиши определяет 7- или 8-разрядное число, которое должно передаваться последовательно по телефонной линии.

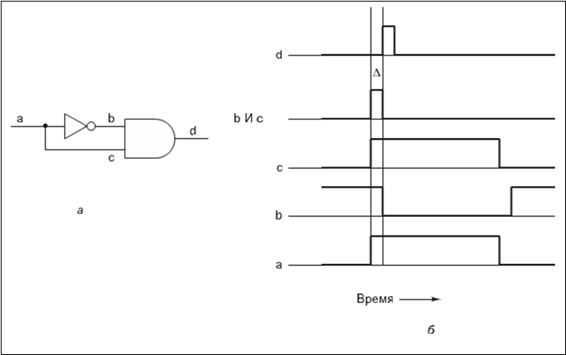
Противоположностью мультиплексора является демультиплексор, который соединяет единственный входной сигнал с одним из 2n выходов в зависимости от значений сигналов в n линиях управления. Если бинарное значение линий управления равно k, то выбирается выход k.

Мультиплексоры бывают пирамидальными и каскадными (не по Таненбауму).

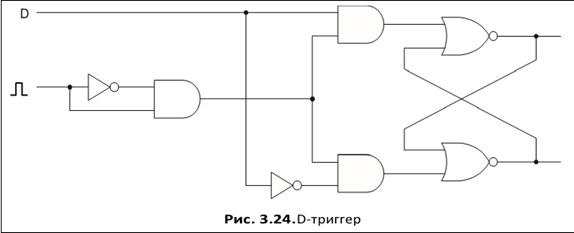
Триггеры

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

Способ генерирования очень короткого импульса на фронте синхронизирующего сигнала:



Может показаться, что схема на рисунке а всегда будет выдавать нулевое значение, но это не так, потому что при прохождение сигнала через инвертор существует некая ненулевая задержка, данная схема работает благодаря ей.



Данный триггер, по сути, состоит из схемы, данной выше, и D-защелки. Следует упомянуть, что такая схема триггера проста для понимания, но на практике обычно используются более сложные триггеры

## 34. **Устройство и назначение системы BIOS. Тест начального включения POST**

1. BIOS (Basic Input/Output System – базовая система ввода-вывода) – это программа для первоначального запуска компьютера, настройки оборудования и обеспечения функций ввода-вывода.

BIOS записывается в микросхему flash-памяти, которая расположена на системной плате. Изначально основным назначением BIOS было обслуживание устройств ввода-вывода (клавиатуры, экрана и дисковых накопителей), поэтому ее и назвали «базовая система ввода-вывода». В современных компьютерах BIOS выполняет несколько функций:

- запуск компьютера и процедуру самотестирования (Power-On Self Test, POST). Программа, расположенная в микросхеме BIOS, загружается первой после включения питания компьютера. Она детектирует и проверяет установленное оборудование, настраивает его и готовит к работе. Если обнаруживается неисправность оборудования, процедура POST останавливается с выводом соответствующего сообщения илизвукового сигнала;

- настройку параметров системы с помощью программы BIOS Setup. Во время процедуры POST оборудование настраивается в соответствии с параметрами, хранящимися в специальной CMOS-памяти. Изменяя эти параметры, пользователи могут конфигурировать отдельные устройства и систему в целом по своему усмотрению. Редактируются они в специальной программе, которую называют BIOS Setup или CMOS Setup. Настройке системы с помощью программы BIOS Setup будет посвящена большая часть этой книги;

- поддержку функций ввода-вывода с помощью программных прерываний BIOS. В составе системной BIOS есть встроенные функции для работы с клавиатурой, видеоадаптером, дисководами, жестки ми дисками, портами ввода-вывода и др. Эти функции использовались в операционных системах, подобных MS-DOS, и почти не применяются в современных версиях Windows.

Во всех современных компьютерах BIOS хранится в микросхеме на основе flash-памяти (Flash Memory). Такая микросхема может быть перезаписана с помощью специальных программ прямо на компьютере. Запись новой версии BIOS обычно называется перепрошивкой. Эта операция может понадобиться, чтобы добавить в код BIOS новые функции, исправить ошибки или заменить поврежденные версии.

2.После включения питания, аппаратного сброса от кнопки RESET или нажатии комбинации клавиш Ctrl + Alt + Del процессор переходит к выполнению кода начального самотестирования POST (Power-On Self Test),

хранящейся в микросхеме BIOS. POST выполняет тестирование процессора, памяти и системных средств ввода-вывода, а также конфигурация всех программно-управляемых аппаратных средств системной платы. Часть конфигурации выполняется однозначно, часть управляется джампера системной платы, но ряд параметров позволяет или даже требует конфигурации по желанию пользователя. Для этих целей служит утилита Setup, встроенная в код BIOS. После тестирования и конфигурирования (включающего настройку устройств РnР), POST инициализирует загрузки операционной системы.

При прохождении каждой секции POST записывает ее код (номер) в диагностический регистр. Этот регистр физически располагается на специальной диагностической плате, устанавливаемой в слот системной шины. Плата содержит 8-битный регистр со световой (двоичной или шестнадцатиричной) индикацией состояния битов. В пространстве ввода-вывода регистр занимает один адрес, зависящую от архитектуры PC (точнее, версии BIOS): ISA, EISA - 80h, ISA-Compaq - 84h, ISA-PS / 2 - 90h, MCA-PS / 2 - 680h, некоторые модели EISA - 300h (часто пишут то же и в 80h). По индикаторам платы можно определить, на какой секции остановился POST, и определить причину неисправности. Однако для использования такой диагностики необходима, во-первых, сама плата-индикатор, и во-вторых, "словарь" неисправностей - таблица, специфическая для версии BIOS и системной платы.

Во время выполнения POST может выдавать диагностические сообщения в виде последовательности коротких и длинных звуковых сигналов, а после успешной инициализации графического адаптера короткие текстовые сообщения выводятся на экран монитора.

Привычная последовательность шагов POST:

\* Тестирование регистров процессора.

\* Проверка контрольной суммы BIOS.

\* Проверка и инициализация таймера 8253/8254, портов 8255.

После этого шага доступна звуковая диагностика.

\* Проверка и инициализация контроллеров DMA 8237.

\* Проверка регенерации памяти.

\* Тестирование 64 Кбайт нижней памяти.

\* Загрузка векторов прерывания и стека в нижнюю область памяти.

\* Инициализация Видеоконтроллер - на экране появляется заставка Video BIOS, обычно с указанием модели видеокарты и объемом установленной видеопамяти.

36.Общий обзор уровня архитектуры команд (Свойства, модели памяти, регистры, команды)

Уровень архитектуры набора команд связывает компиляторы и аппаратное обеспечение. Это язык, который понятен и компилятору, и аппаратному обеспечению.

В книге так: В принципе, уровень архитектуры набора команд — это тот уровень, на котором компьютер представляется программисту, пишущему программы на машинном языке. Поскольку сейчас ни один нормальный программист таких программ не пишет, мы слегка переделали это определение: программа уровня архитектуры набора команд — это то, что получается в результате работы компилятора (в данный момент мы не будем касаться системных вызовов и символического языка ассемблера). Чтобы получить программу уровня архитектуры набора команд, создатель компилятора должен знать, какая модель памяти используется в машине, какие регистры, типы данных и команды имеются в наличии и т. д. Вся эта информация в совокупности и определяет уровень архитектуры набора команд.

Свойства

Я хз что здесь писать, в книге две страницы примерно про свойства, а свойства я не нашел :D

Осмелюсь предположить, что это модель памяти, используемая в машине, регистры, типы данных и команды.

Модели памяти

Во всех компьютерах память разделена на ячейки, которые имеют последовательные адреса. В настоящее время наиболее распространенный размер ячейки — 8 бит. Ячейка из 8 бит называется байтом (или октетом).

Байты обычно группируются в 4-байтные (32-разрядные) или 8-байтные (64-разрядные) слова, а в наборе присутствуют команды манипулирования целыми словами. Многие архитектуры требуют, чтобы слова выравнивались по своим естественным границам. Так, 4-байтное слово может начинаться с адреса 0, 4, 8 и т. д., но не с адреса 1 или 2. Точно так же слово из 8 байт может начинаться с адреса 0, 8 или 16, но не с адреса 4 или 6.

В некоторых машинах содержатся раздельные адресные пространства для команд и данных, так что при вызове команды с адресом 8 и вызове данных с адресом 8 происходит обращение к разным адресным пространствам. Такая система гораздо сложнее, чем единое адресное пространство, но зато она имеет два преимущества. Во-первых, все с теми же 32-разрядными адресами появляется возможность иметь 2^32 байт для программ и дополнительные 2^32 байт для данных. Во-вторых, поскольку запись всегда автоматически происходит только в пространство данных, случайная перезапись программы становится невозможной, и следовательно, устраняется один из источников программных ошибок. Наконец, разделение команд и данных усложняет атаки со стороны вредоносных программ, которые не могут изменить программный код — они даже не могут адресовать его.

Регистры

Регистры - это блок ячеек памяти внутри самого процессора.

Регистры уровня архитектуры набора команд можно разделить на две категории: специальные регистры и регистры общего назначения. К специальным регистрам относятся счетчик команд и указатель стека, а также другие регистры, имеющие особые функции. Регистры общего назначения содержат ключевые локальные переменные и

промежуточные результаты вычислений. Их основная функция состоит в том, чтобы обеспечить быстрый доступ к часто используемым данным (обычно без обращений к памяти). RISC-машины с высокоскоростными процессорами и (относительно) медленной памятью обычно содержат как минимум 32 регистра общего назначения, причем в новых процессорах количество регистров общего назначения постоянно растет.

В некоторых машинах регистры общего назначения полностью симметричны и взаимозаменяемы. Если все регистры эквивалентны, для хранения промежуточного результата компилятор может использовать как регистр R1, так и регистр R25. Выбор регистра не имеет никакого значения.

Регистр PSW (Program State Word — слово состояния программы), который еще называют флаговым. Флаговый регистр содержит различные биты, необходимые центральному процессору. Самые важные биты — это коды условий. Они устанавливаются в каждом цикле АЛУ (Арифметико-Логических Устройств) и отражают состояние результата предыдущей операции:

N — результат отрицателен (Negative);

Z — результат равен 0 (Zero);

V — результат вызвал переполнение (oVerflow);

С — перенос самого левого бита (Carry out);

A — перенос бита 3 (Auxiliary carry — служебный перенос);

P — результат четный (Parity).

Коды условий очень важны, поскольку используются при сравнениях и условных переходах. Например, команда CMP обычно вычитает один операнд из другого и на основе полученной разности устанавливает коды условий. Если операнды равны, то разность будет равна 0, а во флаговом регистре установится бит Z. Последующая команда BEQ (Branch Equal — переход в случае равенства) проверяет бит Z и совершает переход, если он установлен.

Команды

Главная особенность уровня, который мы сейчас рассматриваем, — это набор машинных команд. Они управляют действиями машины. В этом наборе всегда в той или иной форме присутствуют команды LOAD и STORE, предназначенные для перемещения данных между памятью и регистрами, и команда MOVE, которая служит для копирования данных из одного регистра в другой. Также всегда присутствуют арифметические и логические команды, команды сравнения элементов данных и команды переходов в зависимости от результатов.

Крч, это все, что есть в кинжке. Я бы ей на этот вопрос ответил так:

Уровнень архитектуры набора команд содержит команды предназначенные для пермещения данных между памятью и регистрами и для копирования данных из одного регистра в другой. Также он всегда содержит арифметические и логические команды, команды сравнения данных и команды переходов.

Потому что она говорила только это. Команды ЛОАД, СТОР и МУВ мы не проходили. Она постоянно говорила, что на экзамене всегда видит, что студенты списывают именно из-за команд, которые она не объясняла.

## 35. **Типы данных центральных процессоров Pentium 4, UltraSPARC III, микросхемы 8051**

Pentium 4

Pentium 4 поддерживает двоичные целые числа со знаком, целые числа без знака, числа двоично-десятичной системы счисления и числа с плавающей точкой по стандарту IEEE 754 (табл. 5.2). Эта машина является 8/16-разрядной и оперирует целыми числами такой же длины. Она поддерживает многочисленные арифметические команды, булевы операции и операции сравнения. Операнды не обязательно должны быть выровнены в памяти, но если адреса слов кратны значению 4 байта, имеет место более высокая производительность.

Pentium 4 также может манипулировать 8-разрядными ASCII-символами: существуют специальные команды для копирования и поиска строк символов. Эти команды используются и для строк, длина которых известна заранее, и для строк, в конце которых стоит специальный терминальный символ. Строковые команды часто объединяются в библиотеки.

UltraSPARC III

UltraSPARC III поддерживает множество форматов данных (табл. 5.3). Эта машина может обрабатывать 8-, 16-, 32- и 64-разрядные целочисленные операнды со знаком и без знака. Целые числа со знаком представлены в дополнительном коде. Кроме того, имеются операнды с плавающей точкой по 32, 64 и 128 бит, которые соответствуют стандарту IEEE 754 (для 32- и 64-разрядных чисел). Двоично-десятичные числа не поддерживаются. Все операнды должны быть выровнены в памяти.

8051

Количество типов данных в микросхеме 8051 строго ограничено. Разрядность всех регистров, а значит, целых чисел и символов, составляет 8 бит. По существу, единственным типом данных для выполнения арифметических операций, который поддерживается аппаратно, является 8-разрядный байт

Кроме того, 8051 поддерживает еще один тип данных, который не используется при выполнении арифметических операций, - бит. Блок из 16 байт, начинающийся с адреса 32, являет собой память с битовой ориентацией. Путем смещения от 0 до 127 можно обращаться индивидуально к каждому биту. Бит 0 занимает в байте 32 крайнее правое положение, бит 1 расположен сразу за ним, и т. д. Для битов предусмотрены команды установки, сброса, выполнения операций И и ИЛИ, образования дополнения, перехода к битам и проверки. Во встроенных системах состояния переключателей, световых индикаторов и других устройств ввода-вывода сохраняются именно в отдельных битах, поэтому возможность непосредственного управления ими очень важна**.**

## 36. **Общий обзор уровня архитектуры набора команд. (Свойства уровня архитектуры набора команд. Модели памяти. Регистры. Команды**

**Уровень архитектуры набора команд** — это тот уровень, на котором компьютер представляется программисту, пишущему программы на машинном языке.

( **Поскольку сейчас ни один нормальный программист таких программ не пишет,** определение слегка переделано: **программа уровня архитектуры набора команд — это то, что получается в результате работы компилятора)**

**Свойства:**

Чтобы получить программу уровня архитектуры набора команд, создатель компилятора должен знать, какая модель памяти используется Общий обзор уровня архитектуры набора команд в машине, какие регистры, типы данных и команды имеются в наличии и т. д. Вся эта информация в совокупности и определяет уровень архитектуры набора команд.

Для одних архитектур уровень команд определяется формальным документом, который обычно выпускается промышленным консорциумом, для других — нет. Например, ARM v7 (версия 7 ARM ISA) имеют официальное определение, опубликованное ARM Ltd. Цель такого официального документа — дать возможность различным производителям выпускать машины данного типа, чтобы эти машины могли выполнять одни и те же программы и получать при этом одни и те же результаты

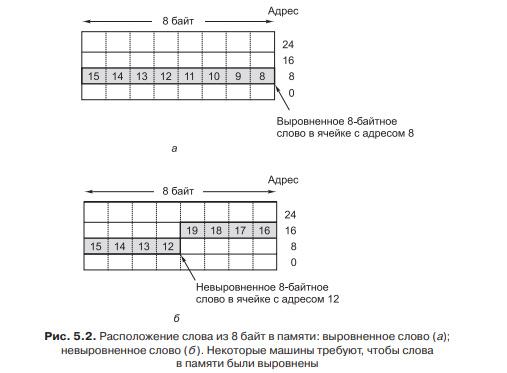
В таких документах содержатся нормативные разделы, в которых излагаются требования, и информативные разделы, призванные помочь читателю, но не являющиеся частью формального определения. В нормативных разделах постоянно встречаются такие слова, как должно быть, нельзя, следует, означающие соответственно требование, запрет и рекомендацию

Другое важное качество уровня архитектуры набора команд состоит в том, что в большинстве машин поддерживаются, по крайней мере, два режима. В привилегированном режиме запускается операционная система. Этот режим позволяет выполнять все команды. Пользовательский режим предназначен для запуска прикладных программ. Он не позволяет выполнять некоторые потенциально опасные команды (например, те, которые непосредственно манипулируют кэш-памятью).

Во всех компьютерах память разделена на ячейки, которые имеют последовательные адреса. В настоящее время наиболее распространенный размер ячейки — 8 бит, но раньше использовались ячейки от 1 до 60 бит (см. табл. 2.1). Ячейка из 8 бит называется байтом (или октетом).

Байты обычно группируются в 4-байтные (32-разрядные) или 8-байтные (64-разрядные) слова, а в наборе присутствуют команды манипулирования целыми словами. Многие архитектуры требуют, чтобы слова выравнивались по своим естественным границам.

Выравнивание адресов требуется довольно часто, поскольку при этом память работает наиболее эффективно. Например, процессор Core i7, который вызывает из памяти по 8 байт через интерфейс DDR3, поддерживает только выровненные 64-разрядные обращения. Следовательно, Core i7 вообще не сможет обратиться к невыровненной памяти, поскольку интерфейс памяти требует, чтобы адреса были кратны 8.



Большинство машин имеют единое линейное адресное пространство, которое простирается от адреса 0 до какого-то максимума, обычно 232 или 264 байт. В некоторых машинах содержатся раздельные адресные пространства для команд и данных, так что при вызове команды с адресом 8 и вызове данных с адресом 8 происходит обращение к разным адресным пространствам. Такая система гораздо 382      Глава 5. Уровень архитектуры набора команд сложнее, чем единое адресное пространство, но зато она имеет два преимущества. Во-первых, все с теми же 32-разрядными адресами появляется возможность иметь 232 байт для программ и дополнительные 232 байт для данных. Во-вторых, поскольку запись всегда автоматически происходит только в пространство данных, случайная перезапись программы становится невозможной, и следовательно, устраняется один из источников программных ошибок. Наконец, разделение команд и данных усложняет атаки со стороны вредоносных программ, которые не могут изменить программный код — они даже не могут адресовать его.

Еще один аспект модели памяти — семантика памяти. Естественно ожидать, что команда LOAD, если она выполняется после команды STORE, обратится к тому же адресу и возвратит только что сохраненное значение. Однако, как мы видели в главе 4, во многих машинах микрокоманды переупорядочиваются. Таким образом, существует реальная опасность, что память будет работать не так, как ожидается. Ситуация усложняется при наличии мультипроцессора, когда каждый процессор посылает в общую память поток запросов на чтение и запись, и эти запросы тоже могут быть переупорядочены.

**Регистры**

Во всех компьютерах имеются несколько регистров, доступных на уровне архитектуры набора команд. Они позволяют управлять выполнением программы, хранить временные результаты, а также используются для других целей. Обычно регистры, доступные на уровне микроархитектуры (например, TOS и MAR ), на уровне архитектуры набора команд недоступны, однако некоторые регистры, например счетчик команд и указатель стека, доступны на обоих уровнях. В то же время регистры, доступные на уровне архитектуры набора команд, всегда доступны на уровне микроархитектуры, поскольку именно там они реализованы.

Регистры уровня архитектуры набора команд можно разделить на две категории: специальные регистры и регистры общего назначения. **К специальным регистрам относятся** счетчик команд и указатель стека, а также другие регистры, имеющие особые функции. **Регистры общего назначения содержат** ключевые локальные переменные и промежуточные результаты вычислений. Их основная функция состоит в том, чтобы обеспечить быстрый доступ к часто используемым данным (обычно без обращений к памяти). **RISC-машины с высокоскоростными процессорами и (относительно) медленной памятью обычно содержат** как минимум 32 регистра общего назначения, причем в новых процессорах количество регистров общего назначения постоянно растет. **В некоторых машинах регистры общего назначения полностью симметричны и взаимозаменяемы**. Если все регистры эквивалентны, для хранения промежуточного результата компилятор может использовать как регистр R1, так и регистр R25. Выбор регистра не имеет никакого значения. **В других машинах некоторые регистры общего назначения могут быть специализированными**. Например, в процессоре Core i7 имеется регистр EDX, который может использоваться в качестве регистра общего назначения, но который, кроме того, используется для решения сугубо специфических задач (получает половину произведения при умножении и половину делимого при делении). **Даже если регистры общего назначения полностью взаимозаменяемы,** операционная система или компиляторы часто соблюдают формальные соглашения по их использованию.

**Помимо регистров, доступных на уровне** архитектуры набора команд, всегда существуют довольно много специальных регистров, доступных только в привилегированном режиме. **Эти регистры управляют различными блоками кэшпамяти**, основной памятью, устройствами ввода-вывода, другими устройствами машины. **Данные регистры используются** только операционной системой, поэтому компиляторам и пользователям не обязательно знать об их существовании. Уровень архитектуры набора команд Один «гибридный» регистр доступен и в привилегированном, и в пользовательском режимах. Это регистр PSW (Program State Word — слово состояния программы), который еще называют флаговым. Флаговый регистр содержит различные биты, необходимые центральному процессору. Самые важные биты — это коды условий. Они устанавливаются в каждом цикле АЛУ и отражают состояние результата предыдущей операции

**Флаговый регистр может хранить** не только коды условий. Его содержимое в разных машинах может быть разным. Дополнительные поля указывают режим машины (например, пользовательский или привилегированный), бит трассировки (который используется для отладки), уровень приоритета процессора, статус разрешения прерываний**. Флаговый регистр обычно читается в пользовательском** режиме, но некоторые поля могут записываться только в привилегированном режиме (например, бит, который указывает режим).

**Команды**

Главная особенность уровня, который мы сейчас рассматриваем, — это набор машинных команд. Они управляют действиями машины. В этом наборе всегда в той или иной форме присутствуют команды LOAD и STORE, предназначенные для перемещения данных между памятью и регистрами, и команда MOVE, которая служит для копирования данных из одного регистра в другой. Также всегда присутствуют арифметические и логические команды, команды сравнения элементов данных и команды переходов в зависимости от результатов. Некоторые типичные команды мы уже рассматривали в главе 4 (см. табл. 4.2.), а в этой главе мы познакомимся со многими другими

## 37. **Конвейер Netburst**

Netburst - это конвейер нового типа, разработанный компанией Интел в 2000 году. Его разрабатывали для повышения тактовой частоты процессоров, поэтому он имел ряд характерных отличий.

Основные особенности конвейера - это гиперконвейеризация и применение кэша последовательностей микроопераций вместо традиционного кэша инструкций. Также имелось существенное отличие АЛУ процессоров на базе Netburst (Арифметико-логическое устройство).

Гиперконвейризация

В связи с применением кэша последовательностей микроопераций декодер вынесен за пределы конвейера. Это позволяет процессорам Pentium 4 достигать более высоких тактовых частот по сравнению с процессорами, имеющими более короткий конвейер при одинаковой технологии производства.

Декодер - это такая штука, которая получает инструкцию размером 32 бита, в которой хранятся код операции, адрес регистра источника и получателя, и разбивает ее на части.

Также в Netburst использовали увеличенный по сравнению с предшественниками буфер предсказания ветвлений, что помогло повысить точность предсказаний до 94%.

Предсказания переходов - это когда компьютер предполагает какие именно данные будут использоваться в ближайшее время еще до того, как условие сможет быть проверено, и заносит их в кеш для быстрого доступа. Это повышает производительность машины

Кэш последовательностей микроопераций

Перед исполнением сложные инструкции x86 преобразуются в более простой набор внутренних инструкций (микроопераций), что позволяет повысить скорость обработки команд. Однако, вследствие того, что инструкции x86 имеют переменную длину и не имеют фиксированного формата, их декодирование связано с существенными временными затратами.

В связи с этим, при разработке архитектуры NetBurst было принято решение отказаться от традиционной кэш-памяти инструкций первого уровня, хранящей команды x86, в пользу кэша последовательностей микроопераций, хранящего последовательности микроопераций в соответствии с предполагаемым порядком их исполнения. Ёмкость trace cache составляла около 12 тыс. микроопераций. Такая организация кэш-памяти позволила также снизить временные затраты на выполнение условных переходов и на выборку инструкций.

АЛУ

Так как основной целью разработки архитектуры NetBurst было повышение производительности за счёт достижения высоких тактовых частот, возникла необходимость увеличения темпа выполнения основных целочисленных операций. Для достижения этой цели АЛУ процессоров архитектуры NetBurst разделено на несколько блоков: «медленное АЛУ» (англ. slow ALU), способное выполнять большое количество целочисленных операций, и два «быстрых АЛУ» (англ. 2X ALU), выполняющих только простейшие целочисленные операции (например, сложение). Выполнение операций на «быстрых АЛУ» происходит последовательно в три этапа: сначала вычисляются младшие разряды результата, затем старшие, после чего могут быть получены флаги. (Флаги это что-то вроде результата вычислений. Подробную информацию об этом можно найти в ответе на вопрос 36 в разделе регистры)

## 38. **Режимы адресации**

**Непосредственная:**

Самый простой способ указания операнда — хранить в адресной части сам операнд, а не адрес операнда или какую-либо другую информацию, описывающую, где находится операнд. Такой операнд называется непосредственным, поскольку он автоматически вызывается из памяти одновременно с командой; следовательно, сразу становится непосредственно доступным. При непосредственной адресации не требуется дополнительного обращения к памяти для вызова операнда. Однако у такого способа адресации есть недостатки. Во-первых, таким способом можно работать только с константами. Во-вторых, число значений ограничено размером поля. Тем не менее эта технология используется во многих архитектурах для определения целочисленных констант.

**Прямая адресация:**

Следующий способ определения операнда — просто дать его полный адрес. Такой режим называется прямой адресацией. Как и непосредственная адресация, прямая адресация имеет некоторые ограничения: команда всегда имеет доступ только к одному и тому же адресу памяти. То есть значение может меняться, а адрес нет. Таким образом, прямая адресация может использоваться только для доступа к глобальным переменным, адреса которых известны во время компиляции. Многие программы содержат глобальные переменные, поэтому этот способ широко используется.

**Регистровая адресация:**

Регистровая адресация по сути напоминает прямую, только в данном случае вместо ячейки памяти указывается регистр. Поскольку регистры очень важны (благодаря быстрому доступу и коротким адресам), этот режим адресации является самым распространенным на большинстве компьютеров. Многие компиляторы определяют, к каким переменным доступ будет осуществляться чаще всего (например, индексы циклов), и помещают эти переменные в регистры. Такой режим называют регистровой адресацией. В архитектурах с перенаправлением для загрузки (например, в архитектуре ARM OMAP4420) практически все команды используют этот режим адресации. Он не применяется только в том случае, если операнд перемещается из памяти в регистр (команда LDR) или из регистра в память (команда STR). Но даже в этих командах один из операндов является регистром — туда отправляется слово из памяти и оттуда перемещается слово в память.

**Косвенная регистровая адресация:**

При косвенной регистровой адресации искомый операнд берется из памяти или отправляется в память, но адрес не фиксируется жестко в команде, как при прямой адресации, а находится в регистре. Если адрес используется таким образом, он называется указателем. Преимущество косвенной адресации состоит в том, что можно обращаться к памяти, не имея в команде полного адреса. Кроме того, многократно выполняя данную команду, можно, меняя значение в регистре, использовать разные слова памяти.

**Индексная адресация:**

Часто нужно уметь обращаться к словам памяти по известному смещению относительно регистра. Обращение к памяти по регистру и константе смещения называется индексной адресацией.

**Стековая адресация:**

Стековая адресация означает, что содержимое специального регистра—указателя стека является адресом данных. Команды, в которых используется стековая адресация, как правило, имеют меньший формат, чем команды с любым другим методом обращения к памяти. Этот метод адресации может быть использован вместо косвенной или индексной адресации, но преимущество его состоит в том, что адрес операнда содержится в самом коде операции. Кроме того, стековая адресация не требует от ЦП выполнения операции сложения для получения исполнительного адреса. Так как добавление новых данных -к хранимым в стеке не разрушает хранимой информации, а только скрывает ее, то этот метод позволяет использовать вложенные программы, которые могут выполняться, когда выполнение текущей ггрограммы прервано или приостановлено, или рекурсивные программы, которые могут многократно использоваться или многократно обращаться к подпрограммам.

**Страничная организация памяти:**

Страничная память — способ организации виртуальной памяти, при котором единицей отображения виртуальных адресов на физические является регион постоянного размера (т. н. страница). Типичный размер страницы — 4096 байт, для некоторых архитектур — до 128 КБ.

Поддержка такого режима присутствует в большинстве 32-битных и 64-битных процессоров. Такой режим является классическим для почти всех современных ОС, в том числе Windows и семейства [UNIX](https://ru.bmstu.wiki/UNIX). Широкое использование такого режима началось с процессора VAX и ОС VMS с конца 70-х годов (по некоторым сведениям, первая реализация). В семействе х86 поддержка появилась с поколения 386, оно же первое 32-битное поколение.

Решаемые задачи:

* Поддержка изоляции процессов и защиты памяти путём создания своего собственного виртуального адресного пространства для каждого процесса
* Поддержка изоляции области ядра от кода пользовательского режима
* Поддержка памяти «только для чтения» и неисполняемой памяти
* Поддержка отгрузки давно не используемых страниц в область подкачки на диске
* Поддержка отображённых в память файлов, в том числе загрузочных модулей
* Поддержка разделяемой между процессами памяти, в том числе с копированием-по-записи для экономии физических страниц
* Поддержка системного вызова fork() в ОС семейства [UNIX](https://ru.bmstu.wiki/UNIX)
* Уменьшение внешней фрагментации – использование блоков фиксированного размера в виртуальной и физической памяти, все запросы на выделение памяти будут кратны, не будет оставаться некратных зон.

**Сегментная организация памяти:**

Сегментная организация памяти (segmentation) **-**схема распределения памяти в виде сегментов переменной длины, соответствующая пользовательской трактовке распределения памяти, т.е. логической структуре программ и данных. С точки зрения пользователя (разработчика программы), программа – это набор **модулей**кода и данных, каждому из которых должен соответствовать свой **сегмент**в памяти. Сегмент – логическая единица распределения памяти, предназначенная для размещения в памяти одного модуля программного кода или данных. Например, в виде сегментов памяти могут быть представлены:

* *основная программа*;
* процедура;
* функция;
* метод;
* объект;
* набор локальных переменных;
* набор глобальных переменных;
* общий блок данных (например, COMMON-блок в языке *FORTRAN*);
* стек;
* *таблица символов*;
* массив.

## 

## 39. **Виртуальная память в windows и linux**

В Windows 7 каждый пользовательский процесс имеет собственное виртуальное адресное пространство. В 32-разрядной версии Windows 7 длина виртуального адреса составляет 32 бита, поэтому у каждого процесса есть 4 Гбайт виртуального адресного пространства. Нижние 2 Гбайт предназначены для кода и данных процесса; верхние 2 Гбайт обеспечивают доступ (ограниченный) к памяти ядра. Исключение составляют серверные версии Windows 7, в которых разделение памяти может быть другим (3 Гбайт пользователю и 1 Гбайт ядру). Виртуальное адресное пространство с подкачкой страниц по требованию содержит страницы фиксированного размера Адресное пространство 64-разрядной версии Windows 7 имеет аналогичную структуру, но пространство кода и данных в нем находится в младших 8 терабайтах виртуального адресного пространства. Каждая виртуальная страница может находиться в одном из трех состояний: свободна (free), зарезервирована (reserved) или выделена (committed). Свободная страница в текущий момент не используется, и обращение к ней вызывает ошибку отсутствия страницы. Когда процесс начинается, все его страницы находятся в свободном состоянии до тех пор, пока программа и начальные данные не будут отображены на свое адресное пространство. Если код или данные отображены в страницу, то такая страница является выделенной. Обращение к выделенной странице будет успешным, если страница находится в основной памяти. Если страница отсутствует в основной памяти, произойдет ошибка, и операционной системе придется вызывать нужную страницу с диска. Виртуальная страница может находиться и в зарезервированном состоянии. Это значит, что страница остается недоступной для отображения до тех пор, пока резервирование не будет отменено. Помимо атрибутов состояния страницы, имеются и другие атрибуты (например, указывающие на возможность чтения, записи и выполнения). Каждая выделенная страница имеет теневую страницу на диске, где она хранится, когда ее нет в основной памяти. Свободные и зарезервированные страницы не имеют теневых страниц, поэтому обращения к ним вызывают ошибки отсутствия страницы (система не может вызвать страницу с диска, если этой страницы нет на диске). Теневые страницы на диске сгруппированы в один или несколько страничных файлов. Операционная система следит, на какую часть какого страничного файла отображается каждая виртуальная страница. Файлы с текстами программ имеют теневые страницы; для страниц данных используются специальные страничные файлы. В Windows 7, как и System V, позволяет отображать файлы прямо на области виртуального адресного пространства. Если файл отображен на адресное пространство, его можно считывать или записывать путем обычных обращений к памяти. Отображаемые на память файлы реализуются так же, как другие выделенные страницы, только теневые страницы могут находиться в файле на диске, а не в страничном файле. Windows 7 позволяет двум и более процессам одновременно отобразить один и тот же файл — возможно, на разные виртуальные адреса. Путем считывания слов из памяти и записи слов в память процессы могут взаимодействовать друг с другом и передавать данные в обоих направлениях с достаточно высокой скоростью, поскольку никакого копирования не требуется. Разные процессы могут обладать разными разрешениями на доступ. Win32 API содержит ряд функций, которые позволяют процессу непосредственно управлять виртуальной памятью. Все они выполняются в области, состоящей либо из одной страницы, либо из двух или более страниц, последовательно расположенных в виртуальном адресном пространстве.

Основные функции API и их описание:

VirtualAlloc Резервирование или выделение области

VirtualFree Освобождение области или отмена выделения

VirtualProtect Изменение варианта защиты (чтение/запись/выполнение)

VirtualQuery Запрос о состоянии области памяти

VirtualLock Запрещение разбиения памяти на страницы (область памяти становится резидентной)

VirtualUnlock Снятие запрета относительно разбиения на страницы

CreateFileMapping Создание объекта отображения файла и назначение (не всегда) ему имени MapViewOfFile Отображение файла или части файла на адресное пространство UnmapViewOfFile Удаление отображенного файла из адресного пространства OpenFileMapping Открытие ранее созданного объекта отображения файла

Эти функции API являются основными. На них строится вся остальная система управления памятью. Например, существуют функции API для размещения и освобождения структур данных в одной или нескольких кучах. Кучи используются для хранения структур данных, которые динамически создаются и разрушаются. Кучи не освобождаются в процессе уборки мусора, поэтому пользовательское программное обеспечение само должно освобождать блоки виртуальной памяти, которые уже не нужны (уборкой мусора называют автоматическое удаление неиспользуемых структур данных). Куча в Windows 7 напоминает результат вызова функции malloc в UNIX, но в Windows XP, в отличие от UNIX, может быть несколько независимых куч.

Модель памяти UNIX довольно проста. Каждый процесс имеет три сегмента: код, данные и стек. В машине с линейным адресным пространством код обычно располагается в нижней части памяти, а за ним следуют данные. Стек помещается в верхней части памяти. Размер кода фиксирован, а данные и стек могут увеличиваться или уменьшаться (в разных направлениях). Такую модель легко реализовать практически на любой машине. В частности, она используется вариантами Linux, работающими на процессорах OMAP4430. Более того, если машина поддерживает страничную память, то все адресное пространство может быть разбито на страницы совершенно прозрачно для пользовательских программ. Им будет известно только то, что размер программы может превышать размер физической памяти машины. Те системы UNIX, которые не поддерживают страничную организацию памяти, обычно подкачивают целые процессы между памятью и диском, чтобы параллельно могло выполняться произвольное число процессов. Версии System V (и Linux) имеют некоторые особенности, позволяющие пользователям управлять виртуальной памятью. Самой важной является способность процесса отображать файл или часть файла на часть адресного пространства процесса. Например, если файл размером 12 Кбайт отображается на виртуальный адрес 144К, то в ячейке с адресом 144К будет находиться первое слово этого файла. Таким образом, можно выполнять ввод-вывод файла без системных вызовов. Поскольку размер некоторых файлов превышает размер виртуального адресного пространства, можно отображать не весь файл, а только его часть. Для отображения сначала нужно открыть файл и получить дескриптор файла fd (file descriptor). Дескриптор используется для идентификации отображаемого файла. Затем процесс совершает вызов paddr = mmap(virtual\_address, length, protection, flags, fd, file\_offset) Этот вызов отображает length байт, начиная со смещения file\_offset

Несколько процессов могут одновременно выполнять отображение одного и того же файла. Есть два варианта разделения. В первом варианте общими являются все страницы, поэтому записи, сделанные одним процессом, доступны всем другим процессам. Эта возможность обеспечивает высокоскоростное взаимодействие между процессами. Во втором варианте страницы остаются общими для всех процессов до тех пор, пока ни один из процессов их не меняет. Как только какой-нибудь процесс попытается произвести запись в страницу, он получит ошибку защиты, в результате которой операционная система предоставит ему собственную копию этой страницы для записи. Такая схема, которая называется копированием при записи (copy on write), используется в том случае, когда для каждого из нескольких процессов нужно создать иллюзию, что только он выполняет отображение файла. В этой модели совместный доступ является оптимизацией, а не частью семантики.

# ***Сети***

## 1.**Планирование сети с хабом (мост, коммутатор, маршрутизатор, шлюз).**

При выборе места для установки концентратора примите во внимание следующие аспекты: - местоположение; - расстояния; - питание. Выбор места установки концентратора является наиболее важным этапом планирования небольшой сети. Хаб разумно расположить вблизи геометрического центра сети (на одинаковом расстоянии от всех компьютеров). Такое расположение позволит минимизировать расход кабеля. Длина кабеля от концентратора до любого из подключаемых к сети компьютеров или периферийных устройств не должна превышать 100 м. Концентратор можно поставить на стол или закрепить его на стене с помощью входящих в комплект хаба скоб. Установка хаба на стене позволяет упростить подключение кабелей, если они уже проложены в офисе. При планировании сети есть возможность наращивания (каскадирования) хабов.

Таблица соответствия уровней и устройств.

#ЛВС – локальная вычислительная сеть, LAN.

Мосты.

У многих организаций имеется по нескольку локальных сетей, которые необходимо объединять между собой. Удобно объединить эти сети в одну большую локальную сеть. Это можно сделать с помощью специальных устройств, называемых мостами (bridges).

Мосты работают на канальном уровне. Они анализируют адреса, содержащиеся в кадрах этого уровня, и в соответствии с ними осуществляют маршрутизацию. Поскольку мосты не исследуют сами данные, передающиеся в кадрах, то они одинаково хорошо справляются с пакетами IP, а также с другими типами пакетов. В отличие от мостов маршрутизаторы (routers) анализируют адреса в пакетах и работают, основываясь на этой информации, поэтому они могут работать только с теми протоколами, для которых предназначены.

В современных мостах имеются несколько портов, рассчитанных обычно на от 4 и до 48 входящих линий определенного типа. Каждый порт изолирован, чтобы быть собственным доменом коллизий. Когда прибывает кадр, мост извлекает из заголовка и анализирует адрес назначения, сопоставляя его с таблицей и определяя, куда этот кадр должен быть передан. Для Ethernet этот адрес — 48-битный адрес назначения. Мост только выдает кадр в нужный порт и может передавать несколько кадров одновременно. Мосты предлагают намного лучшую производительность, чем концентраторы, а изоляция между портами моста также означает, что входные линии могут работать

на различных скоростях, возможно даже с различными типами сетей. Типичный пример — мост с портами, которые соединяются с 10-, 100- и 1000-Мбит/с Ethernet. Буферизация в мосте необходима, чтобы принять кадр на одном порту и передать его на другой порт. Если кадры приходят быстрее, чем они могут быть повторно переданы, мост может исчерпать буферное пространство и начать отказываться от кадров. Например, если Gigabit Ethernet заливает биты в 10-Мбит/с Ethernet на большой скорости, мост должен будет буферизовать их, надеясь не исчерпать память. Эта проблема существует, даже если все порты работают на одной и той же скорости, потому что в данный порт назначения кадры могут быть посланы из нескольких портов. Мосты первоначально предназначались для того, чтобы соединять различные виды ЛВС (локальных вычислительных сетей, LAN), например Ethernet и Token Ring. Однако из-за различий между ЛВС это никогда не работало хорошо. Различные форматы кадра требуют копирования и переформатирования, которое занимает время центрального процессора, требует нового вычисления контрольной суммы и добавляет возможность необнаруженных ошибок из-за плохих битов в памяти моста. Еще одна серьезная проблема без хорошего решения — различные максимальные длины кадра. Нужно стремиться избавиться от слишком длинных кадров. В первую очередь — для совместимости. Двумя другими областями, где ЛВС могут отличаться, является безопасность и качество обслуживания. У некоторых ЛВС есть шифрование канального уровня, например у 802.11, у некоторых, например Ethernet, его нет. У некоторых ЛВС есть возможности для обеспечения качества обслуживания, такие как приоритеты, например у 802.11, у некоторых, например Ethernet, их нет. Следовательно, когда кадр должен быть передан между этими ЛВС, безопасность или качество обслуживания, ожидаемое отправителем, возможно, не может быть обеспечено. По всем этим причинам современные мосты обычно работают с сетями одного типа, а для присоединения к сетям различных типов используются маршрутизаторы.

Коммутаторы

Коммутаторы — это другое название современных мостов. Различия больше связаны с маркетингом, чем с техническими особенностями, но есть несколько важных моментов. Мосты были разработаны, когда использовался классический Ethernet, таким образом, они имеют тенденцию присоединяться к относительно небольшому числу ЛВС, а значит, иметь относительно немного портов. Термин «коммутатор» более популярен в настоящее время. Кроме того, все современные системы используют двухточечные линии, такие как кабели витой пары, таким образом, отдельные компьютеры включаются непосредственно в коммутатор, и поэтому коммутатор будет стремиться иметь много портов. Наконец, «коммутатор» также используется в качестве общего термина. С мостом функциональность ясна. С другой стороны, «коммутатор» может относиться к коммутатору Ethernet или абсолютно другому виду устройства, которое принимает решения по перенаправлению, такому как телефонный коммутатор.

Маршрутизаторы

Маршрутизаторы резко отличаются от всего рассмотренного выше. Когда пакет прибывает на маршрутизатор, отрезаются заголовки и концевики кадров и остаются только поля данных, которые и передаются программному обеспечению маршрутизатора. Далее анализируется заголовок пакета и в соответствии с ним выбирается его дальнейший путь. Если это IP-пакет, то в заголовке будет содержаться 32-битный (IPv4) или 128-битный (IPv6), а не 48-битный IEEE 802 адрес. Программное обеспечение маршрутизатора не интересуется адресами кадров и даже не знает, откуда эти кадры взялись (то ли с ЛВС, то ли с двухточечной линии).

Шлюзы

Шлюзы служат для соединения компьютеров, использующих различные транспортные протоколы, ориентированные на установление соединения. Например, такая ситуация возникает, когда компьютеру, использующему ориентированный на соединение протокол TCP, необходимо передать данные компьютеру, использующему другой ориентированный на соединение протокол SCTP. Транспортный шлюз может копировать пакеты между соединениями, одновременно приводя их к

нужному формату. Наконец, шлюзы приложений уже работают с форматами и содержимым пакетов, занимаясь переформатированием на более высоком уровне. Например, шлюз e-mail может переводить электронные письма в формат SMS-сообщений для мобильных телефонов. Как и «коммутатор», «шлюз» — своего рода общий термин. Он относится к процессу пересылки, который выполняется в верхнем уровне.

## 2. **Эталонная модель OSI**

**Эталонная модель OSI** - абстрактная сетевая **модель** для коммуникаций и разработки сетевых протоколов.

Модель OSI имеет семь уровней. Появление именно такой структуры было обусловлено следующими соображениями.

1. Уровень должен создаваться по мере необходимости отдельного уровня абстракции.

2. Каждый уровень должен выполнять строго определенную функцию.

3. Выбор функций для каждого уровня должен осуществляться с учетом создания стандартизированных международных протоколов.

4. Границы между уровнями должны выбираться так, чтобы поток данных между интерфейсами был минимальным.

|  |  |
| --- | --- |
| |  | | --- | |  | |

5. Количество уровней должно быть достаточно большим, чтобы различные функции не объединялись в одном уровне без необходимости, но не слишком высоким, чтобы архитектура не становилась громоздкой

**Физический уровень**

занимается реальной передачей необработанных битов по каналу связи. При разработке сети необходимо убедиться, что когда одна сторона передает единицу, то принимающая сторона получает также единицу, а не ноль. Принципиальными вопросами здесь являются следующие: какое напряжение должно использоваться для отображения единицы, а какое для нуля; сколько микросекунд длится бит; может ли передача производиться одновременно в двух направлениях; как устанавливается начальная связь и как она прекращается, когда обе стороны закончили свои задачи; из какого количества проводов должен состоять кабель и какова функция каждого провода. Вопросы разработки в основном связаны с механическими, электрическими и процедурными интерфейсами, а также с физическим носителем, лежащим ниже физического уровня.

**Уровень передачи данных**

Основная задача уровня передачи данных — быть способным передавать «сырые» данные физического уровня по надежной линии связи, свободной от необнаруженных ошибок, и маскировать реальные ошибки, так что сетевой уровень их не видит. Эта задача выполняется при помощи разбиения входных данных на кадры, обычный размер которых колеблется от нескольких сот до нескольких тысяч байт. Кадры данных передаются последовательно с обработкой **кадров подтверждения**, отсылаемых обратно получателем.

**Сетевой уровень**

занимается управлением операциями подсети. Важнейшим моментом здесь является определение маршрутов пересылки пакетов от источника к пункту назначения. Маршруты могут быть жестко заданы в виде таблиц и редко меняться либо, что бывает чаще, автоматически изменяться, чтобы избегать отказавших компонентов. Кроме того, они могут задаваться в начале каждого соединения, например, терминальной сессии, такого как подключения к удаленной машине. Наконец, они могут быть в высокой степени динамическими, то есть вычисляемыми заново для каждого пакета с учетом текущей загруженности сети.

Если в подсети одновременно присутствует слишком большое количество пакетов, то они могут закрыть дорогу друг другу, образуя заторы в узких местах. Недопущение подобной закупорки также является задачей сетевого уровня в соединении с более высокими уровнями, которые адаптируют загрузку. В более общем смысле, сетевой уровень занимается предоставлением определенного уровня сервиса (это касается задержек, времени передачи, вопросов синхронизации).

При путешествии пакета из одной сети в другую также может возникнуть ряд проблем. Так, способ адресации, применяемый в одной сети, может отличаться от принятого в другой. Сеть может вообще отказаться принимать пакеты из-за того, что они слишком большого размера. Также могут различаться протоколы и т. д. Именно сетевой уровень должен разрешать все эти проблемы, позволяя объединять разнородные сети.

**Транспортный уровень**

Основная функция транспортного уровня — принять данные от сеансового уровня, разбить их при необходимости на небольшие части, передать их сетевому уровню и гарантировать, что эти части в правильном виде прибудут по назначению. Кроме того, все это должно быть сделано эффективно и таким образом, чтобы изолировать более высокие уровни от каких-либо изменений в аппаратной технологии с течением времени.

Транспортный уровень также определяет тип сервиса, предоставляемого сеансовому уровню и, в конечном счете, пользователям сети. Наиболее популярной разновидностью транспортного соединения является защищенный от ошибок канал между двумя узлами, поставляющий сообщения или байты в том порядке, в каком они были отправлены. Однако транспортный уровень может предоставлять и другие типы сервисов, например пересылку отдельных сообщений без гарантии соблюдения порядка их доставки или одновременную отправку сообщения различным адресатам по принципу широковещания.

**Сеансовый уровень**

позволяет пользователям различных компьютеров устанавливать сеансы связи друг с другом. При этом предоставляются различные типы сервисов, среди которых управление диалогом (отслеживание очередности передачи данных), управление маркерами (предотвращение одновременного выполнения критичной операции несколькими системами) и синхронизация (установка служебных меток внутри длинных сообщений, позволяющих продолжить передачу с того места, на котором она оборвалась, даже после сбоя и восстановления).

**Уровень представления**

В отличие от более низких уровней, задача которых — достоверная передача битов и байтов, уровень представления занимается по большей части синтаксисом и семантикой передаваемой информации. Чтобы было возможно общение компьютеров с различными внутренними представлениями данных, необходимо преобразовывать форматы данных друг в друга, передавая их по сети в неком стандартизированном виде. Уровень представления занимается этими преобразованиями, предоставляя возможность определения и изменения структур данных более высокого уровня (например, записей баз данных).

**Прикладной уровень**

содержит набор популярных протоколов, необходимых пользователям. Одним из наиболее распространенных является протокол передачи гипертекста HTTP (HyperText Transfer Protocol), который составляет основу технологии Всемирной паутины. Когда браузер запрашивает веб-страницу, он передает ее имя (адрес) и рассчитывает на то, что сервер, на котором расположена страница, будет использовать HTTP. Сервер в ответ отсылает страницу. Другие прикладные протоколы используются для передачи файлов, электронной почты, сетевых рассылок.

## 3. **Службы и протоколы на основе соединений и службы без установления соединений**

Уровни могут предлагать вышестоящим уровням услуги двух типов: с наличием или отсутствием установления соединения. Типичным примером сервиса с установлением соединения является телефонная связь. Чтобы поговорить с кем-нибудь, необходимо поднять трубку, набрать номер, а после окончания разговора положить трубку. Нечто подобное происходит и в компьютерных сетях: при использовании сервиса с установлением соединения абонент сперва устанавливает соединение, а после окончания сеанса разрывает его. Это напоминает трубу: биты сообщения влетают в один ее конец, а вылетают с другого. В большинстве случаев не возникает путаницы с последовательностью передачи этих битов.

Что касается сервисов без установления соединения, то типичный пример такой технологии – почтовые системы. Каждое письмо содержит полный адрес назначения и проходит по некоему маршруту, который совершенно не зависит от других писем. Обычно то письмо, которое отправлено раньше, в место назначения приходит раньше. Каждая служба характеризуется качеством обслуживания. Некоторые службы являются надежными в том смысле, что они никогда не теряют данных. Обычно надежная служба реализуется при помощи подтверждений, посылаемых получателем в ответ на каждое принятое сообщение, так что отправитель знает, дошло очередное сообщение или нет. Процесс пересылки подтверждений требует некоторых накладных расходов и снижает пропускную способность канала. Впрочем, подобные затраты обычно не очень велики и окупаются, хотя иногда могут быть нежелательными.

Типичным примером необходимости надежной службы на основе соединений является пересылка файлов. Владелец файла хочет быть уверенным, что все биты файла прибыли без искажений и в том же порядке, в котором были отправлены. Вряд ли кто-нибудь отдаст предпочтение службе, которая случайным образом искажает информацию, даже если передача происходит значительно быстрее.

Без установления соединения

Кроме того, существует служба запросов и ответов, в которой отправитель посылает дейтаграммы, содержащие запросы, и получает ответы от получателя. Например, к данной категории можно отнести вопрос к библиотеке о том, где говорят по-уйгурски. Обычно модель запросов и ответов применяется для реализации общения в модели «клиент-сервер»: клиент посылает запрос, а сервер отвечает на него. Обсуждавшийся ранее типы служб сведены в таблицу на рис. 1.13.

Служба Пример

Надежный поток

сообщений Последовательность страниц

Надежный поток

байт Удаленная регистрация

Ненадежное

соединение Цифровая голосовая связь

Ненадежная

дейтаграмма Рассылка рекламы электронной почтой

Дейтаграмма с подтверждениями Заказные письма

Запрос – ответ Запрос к базе данных

Концепция использования ненадежной связи поначалу может показаться несколько странной. В самом деле, почему это может возникать такая ситуация, когда выгоднее предпочесть ненадежную связь надежной? Во-первых, надежное соединение (в том смысле, который был оговорен ранее, то есть с подтверждением) не всегда можно установить. Скажем, Ethernet не является «надежным» средством коммуникации. Пакеты при передаче могут искажаться, но решать эту проблему должны протоколы более высоких уровней. Во-вторых, задержки, связанные с отсылкой подтверждения, в некоторых случаях неприемлемы, особенно при передаче мультимедиа в реальном времени. Именно благодаря этим факторам продолжают сосуществовать надежные и ненадежные соединения.

## 4. **Методика и начальные этапы проектирования сети**

Это примерная последовательность этапов и варианты выбора при проектировании локальной сети

При создании новой сети для какого-нибудь предприятия желательно учитывать следующие факторы:

· Требуемый размер сети (в настоящее время, в ближайшем будущем и по прогнозу на перспективу).

· Структура, иерархия и основные части сети (по подразделениям предприятия, а также по комнатам, этажам и зданиям предприятия).

· Основные направления и интенсивность информационных потоков в сети (в настоящее время, в ближайшем будущем и в дальней перспективе). Характер передаваемой по сети информации (данные, оцифрованная речь, изображения), который непосредственно сказывается на требуемой скорости передачи (до нескольких сотен Мбит/с для телевизионных изображений высокой четкости).

· Технические характеристики оборудования (компьютеров, адаптеров, кабелей, репитеров, концентраторов, коммутаторов) и его стоимость.

· Возможности прокладки кабельной системы в помещениях и между ними, а также меры обеспечения целостности кабеля.

· Обслуживание сети и контроль ее безотказности и безопасности.

· Требования к программным средствам по допустимому размеру сети, скорости, гибкости, разграничению прав доступа, стоимости, по возможностям контроля обмена информацией и т.д.

· Необходимость подключения к глобальным или к другим локальным сетям.

Вполне возможно, что после изучения всех факторов выяснится, что можно обойтись без сети, избежав тем самым довольно больших затрат на аппаратуру и программное обеспечение, установку, эксплуатацию, поддержку и ремонт сети, зарплату обслуживающему персоналу, и т.д.

Под структурой сети понимается способ разделения сети на части (сегменты), а также способ соединения этих сегментов между собой. Сеть предприятия может включать в себя рабочие группы компьютеров, сети подразделений, опорные сети, средства связи с другими сетями. Для объединения частей сети могут использоваться репитеры, репитерные концентраторы, коммутаторы, мосты и маршрутизаторы. Причем в ряде случаев стоимость этого объединительного оборудования может даже превысить стоимость компьютеров, сетевых адаптеров и кабеля, поэтому выбор структуры сети исключительно важен.

В идеале структура сети должна соответствовать структуре здания или комплекса зданий предприятия. Рабочие места группы сотрудников, занимающихся одной задачей (например, бухгалтерия, отдел продаж, инженерная группа), должны размещаться в одной или рядом расположенных комнатах. Тогда можно компьютеры этих сотрудников объединить в один сегмент, в единую рабочую группу и установить вблизи

их комнат сервер, с которым они будут работать, а также концентратор или коммутатор, связывающий все их машины. Точно так же рабочие места сотрудников подразделения, занимающихся комплексом близких задач, лучше расположить на одном этаже здания, что существенно упростит их объединение в сегмент и дальнейшее его администрирование. На этом же этаже удобно расположить коммутаторы, маршрутизаторы и серверы, с которыми работает данное подразделение.

Выбор оборудования

При выборе сетевого оборудования надо учитывать множество факторов, в частности:

· уровень стандартизации оборудования и его совместимость с наиболее распространенными программными средствами;

· скорость передачи информации и возможность ее дальнейшего увеличения;

· возможные топологии сети и их комбинации (шина, пассивная звезда, пассивное дерево);

· метод управления обменом в сети (CSMA/CD, полный дуплекс или маркерный метод);

· разрешенные типы кабеля сети, максимальную его длину, защищенность от помех;

· стоимость и технические характеристики конкретных аппаратных средств (сетевых адаптеров, трансиверов, репитеров, концентраторов, коммутаторов).

Выбор сетевых программных средств

К сожалению, в процессе проектирования сети совершенно невозможно выделить те проблемы, которые должны быть решены в начале, и те, которые можно отложить на самый конец. Выбор программных средств не стоит считать чем-то второстепенным, совершенно не влияющим ни на размер и структуру сети, ни на характеристики требуемого оборудования. Поэтому принимать решение о том, какие программные средства надо использовать или хотя бы к какому классу они должны принадлежать, необходимо в самом начале проектирования.

При выборе сетевого программного обеспечения (ПО) надо, в первую очередь, учитывать следующие факторы:

· Какую сеть поддерживает сетевое ПО: одноранговую, сеть на основе сервера или оба этих типа;

· Максимальное количество пользователей (лучше брать с запасом не менее 20%);

· Количество серверов и возможные их типы ;

· Совместимость с разными операционными системами и компьютерами, а также с другими сетевыми средствами;

· Уровень производительности программных средств в различных режимах работы;

· Степень надежности работы, разрешенные режимы доступа и степень защиты данных;

· Какие сетевые службы поддерживаются;

· И, возможно, главное – стоимость программного обеспечения, его эксплуатации и модернизации.

Доп хар-ки которые могут понадоибтся это уже так от меня

· Создание групп пользователей различного назначения;

· Определение прав доступа пользователей;

· Обучение новых пользователей и оперативная помощь в случае необходимости;

· Контроль дискового пространства всех серверов сети;

· Защита и резервное копирование данных, борьба с компьютерными вирусами;

· Модернизация программного обеспечения и сетевой аппаратуры;

· Настройка сети для получения максимальной производительности.

· Локальные группы регистрируются на локальном компьютере;

· Глобальные группы регистрируются на главном контроллере домена;

· Специальные группы (обычно используются для внутрисистемных нужд);

· Встроенные группы делятся на три категории: администраторы, операторы и другие пользователи.

· Более совершенная архитектура сетевой ОС;

· Универсальность и функциональная полнота программных средств;

· Большее быстродействие при данном типе аппаратуры;

· Упрощенное администрирование сети;

· Значительно более высокая защищенность от вирусов и несанкционированного доступа;

· Поддержка различных типов пользователей на разных компьютерных платформах.

## 5. **Типы линий связи локальных сетей. (Кабели на основе витых пар. Коаксиальные кабели. Оптоволоконные кабели. Бескабельные каналы связи)**

электрические (медные) кабели на основе витых пар проводов (twisted pair), которые делятся на экранированные (shielded twisted pair, STP) и неэкранированные (unshielded twisted pair, UTP);

· электрические (медные) коаксиальные кабели (coaxial cable);

· оптоволоконные кабели (fiber optic).

Каждый тип кабеля имеет свои преимущества и недостатки, так что при выборе надо учитывать как особенности решаемой задачи, так и особенности конкретной сети, в том числе и используемую топологию.

Можно выделить следующие основные параметры кабелей, принципиально важные для использования в локальных сетях:

· Полоса пропускания кабеля (частотный диапазон сигналов, пропускаемых кабелем) и затухание сигнала в кабеле. Два этих параметра тесно связаны между собой, так как с ростом частоты сигнала растет затухание сигнала. Надо выбирать кабель, который на заданной частоте сигнала имеет приемлемое затухание. Или же надо выбирать частоту сигнала, на которой затухание еще приемлемо. Затухание измеряется в децибелах и пропорционально длине кабеля.

· Помехозащищенность кабеля и обеспечиваемая им секретность передачи информации. Эти два взаимосвязанных параметра показывают, как кабель взаимодействует с окружающей средой, то есть, как он реагирует на внешние помехи, и насколько просто прослушать информацию, передаваемую по кабелю.

· Скорость распространения сигнала по кабелю или, обратный параметр – задержка сигнала на метр длины кабеля. Этот параметр имеет принципиальное значение при выборе длины сети. Типичные величины скорости распространения сигнала – от 0,6 до 0,8 от скорости распространения света в вакууме. Соответственно типичные величины задержек – от 4 до 5 нс/м.

· Для электрических кабелей очень важна величина волнового сопротивления кабеля. Волновое сопротивление важно учитывать при согласовании кабеля для предотвращения отражения сигнала от концов кабеля. Волновое сопротивление зависит от формы и взаиморасположения проводников, от технологии изготовления и материала диэлектрика кабеля. Типичные значения волнового сопротивления – от 50 до 150 Ом.

В настоящее время действуют следующие стандарты на кабели:

· EIA/TIA 568 (Commercial Building Telecommunications Cabling Standard) – американский;

· ISO/IEC IS 11801 (Generic cabling for customer premises) – международный;

· CENELEC EN 50173 (Generic cabling systems) – европейский.

Эти стандарты описывают практически одинаковые кабельные системы, но отличаются терминологией и нормами на параметры. В данном курсе предлагается придерживаться терминологии стандарта EIA/TIA 568.

Кабели на основе витых пар

Витые пары проводов используются в дешевых и сегодня, пожалуй, самых популярных кабелях. Кабель на основе витых пар представляет собой несколько пар скрученных попарно изолированных медных проводов в единой диэлектрической (пластиковой) оболочке. Он довольно гибкий и удобный для прокладки. Скручивание проводов позволяет свести к минимуму индуктивные наводки кабелей друг на друга и снизить влияние переходных процессов.

Неэкранированные витые пары характеризуются слабой защищенностью от внешних электромагнитных помех, а также от подслушивания, которое может осуществляться с целью, например, промышленного шпионажа. Причем перехват передаваемой по сети информации возможен как с помощью контактного метода (например, посредством двух иголок, воткнутых в кабель), так и с помощью бесконтактного метода, сводящегося к радиоперехвату излучаемых кабелем электромагнитных полей. Причем действие помех и величина излучения во вне увеличивается с ростом длины кабеля. Для устранения этих недостатков применяется экранирование кабелей.

Коаксиальные кабели

Коаксиальный кабель представляет собой электрический кабель, состоящий из центрального медного провода и металлической оплетки (экрана), разделенных между собой слоем диэлектрика (внутренней изоляции) и помещенных в общую внешнюю оболочку

Коаксиальный кабель до недавнего времени был очень популярен, что связано с его высокой помехозащищенностью (благодаря металлической оплетке), более широкими, чем в случае витой пары, полосами пропускания (свыше 1ГГц), а также большими допустимыми расстояниями передачи (до километра ). К нему труднее механически подключиться для несанкционированного прослушивания сети, он дает также заметно меньше электромагнитных излучений вовне. Однако монтаж и ремонт коаксиального кабеля существенно сложнее, чем витой пары, а стоимость его выше (он дороже примерно в 1,5 – 3 раза). Сложнее и установка разъемов на концах кабеля. Сейчас его применяют реже, чем витую пару. Стандарт EIA/TIA-568 включает в себя только один тип коаксиального кабеля, применяемый в сети Ethernet.

Оптоволоконные кабели

Оптоволоконный (он же волоконно-оптический) кабель – это принципиально иной тип кабеля по сравнению с рассмотренными двумя типами электрического или медного кабеля. Информация по нему передается не электрическим сигналом, а световым. Главный его элемент – это прозрачное стекловолокно, по которому свет проходит на огромные расстояния (до десятков километров) с незначительным ослаблением.

Структура оптоволоконного кабеля очень проста и похожа на структуру коаксиального электрического кабеля (рис. 2.4). Только вместо центрального медного провода здесь используется тонкое (диаметром около 1 – 10 мкм) стекловолокно, а вместо внутренней изоляции – стеклянная или пластиковая оболочка, не позволяющая свету выходить за пределы стекловолокна. В данном случае речь идет о режиме так называемого полного внутреннего отражения света от границы двух веществ с разными коэффициентами преломления (у стеклянной оболочки коэффициент преломления значительно ниже, чем у центрального волокна). Металлическая оплетка кабеля обычно отсутствует, так как экранирование от внешних электромагнитных помех здесь не требуется. Однако иногда ее все-таки применяют для механической защиты от окружающей среды (такой кабель иногда называют броневым, он может объединять под одной оболочкой несколько оптоволоконных кабелей ).

Бескабельные каналы связи

Кроме кабельных каналов в компьютерных сетях иногда используются также бескабельные каналы. Их главное преимущество состоит в том, что не требуется никакой прокладки проводов (не надо делать отверстий в стенах, закреплять кабель в трубах и желобах, прокладывать его под фальшполами, над подвесными потолками или в вентиляционных шахтах, искать и устранять повреждения). К тому же компьютеры сети можно легко перемещать в пределах комнаты или здания, так как они ни к чему не привязаны.

## Службы и протоколы на основе соединений и службы без установления соединений

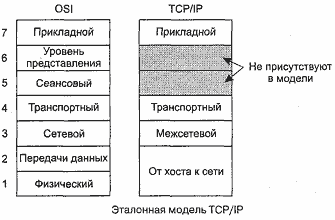
## 6. **Эталонная модель TCP**

**1.Канальный уровень**

Все эти требования привели к выбору сети с пакетной коммутацией, основанной на уровне без установления соединения, который работает в различных сетях. Самый низкий уровень в модели, уровень канала, описывает то, как и что каналы, такие как последовательные линии и классический Ethernet, должны сделать, чтобы удовлетворить потребности этого межсетевого уровня без установления соединения. Это на самом деле не уровень вообще, в нормальном смысле слова, а скорее интерфейс между каналами передачи и узлами. В ранних материалах о модели TCP/IP мало что об этом говорится.

**2.Межсетевой уровень**

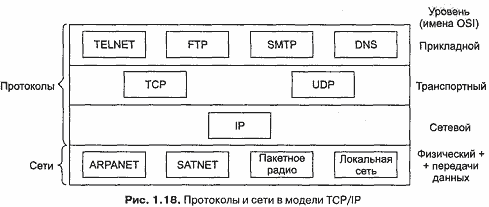
Межсетевой уровень определяет официальный формат пакета и протокол, называемый IP (Internet Protocol). Задачей межсетевого протокола является доставка IP-пакетов к пунктам назначения. Основными аспектами здесь являются выбор маршрута пакета и недопущение закупорки транспортных артерий. Поэтому можно утверждать, что межсетевой уровень модели TCP/IP функционально близок сетевому уровню модели OSI. Это соответствие показано на рис.



**3.Транспортный уровень**

Уровень, расположенный над межсетевым уровнем модели TCP/IP, как правило, на TCP/IP, как правило, на/IP, как правило, на IP, как правило, на, как правило, называют транспортным. Он создан для того, чтобы объекты одного ранга на приемных и передающих хостах могли поддерживать связь, подобно транспортному уровню модели OSI. На этом уровне должны быть описаны два сквозных протокола. Первый, TCP (Transmission Control Protocol — протокол управления передачей), является надежным протоколом с установлением соединений, позволяющим без ошибок

доставлять байтовый поток с одной машины на любую другую машину объединенной сети. Он разбивает входной поток байтов на отдельные сообщения и передает их

межсетевому уровню. На пункте назначения получающий TCP-процесс собирает из полученных сообщений выходной поток. Кроме того, TCP осуществляет управление потоком, чтобы быстрый отправитель не завалил информацией медленного получателя. Второй протокол этого уровня, UDP (User Dataram Protocol — протокол пользовательских дейтограмм2), является ненадежным протоколом без установления соединения, не использующим последовательное управление потоком протокола TCP, а предоставляющим свое собственное. Он также широко используется в одноразовых клиент-серверных запросах и приложениях, в которых оперативность важнее аккуратности, например при передаче речи и видео. Взаимоотношения протоколов IP, TCP и UDP показаны на рис. Со времени создания протокола IP этот протокол был реализован во многих других сетях. 

**4.Прикладной уровень**

В модели TCP/IP нет сеансового уровня и уровня представления. В этих уровнях просто не было необходимости, поэтому они не были включены в модель. Вместо этого приложения просто включают все функции сеансов и представления, которые им нужны. Опыт работы с моделью OSI доказал правоту этой точки зрения: большинство приложений мало нуждаются в этих уровнях. Над транспортным уровнем располагается прикладной уровень. Он содержит все протоколы высокого уровня. К старым протоколам относятся протокол виртуального терминала (TELNET), протокол переноса файлов (FTP) и протокол электронной почты (SMTP). С годами было добавлено много других протоколов. Это DNS (Domain Name Service — служба имен доменов), позволяющая преобразовывать имена хостов в сетевые, HTTP, протокол, используемый для создания страниц на World Wide Web, а также RTP, протокол для представления мультимедиа в реальном времени, таких как звук или фильмы

**5.Хост-сетевой(в учебнике его нет, нашёл в интернете)**

В эталонной модели TCP/IP не описывается подробно, что располагается ниже межсетевого уровня. Сообщается только, что хост соединяется с сетью при помощи какого-нибудь протокола, позволяющего ему посылать по сети IP-пакеты. Этот протокол никак не определяется и может меняться от хоста к хосту и от сети к сети. В книгах и статьях, посвященных модели TCP/IP, этот вопрос обсуждается редко.

**Критика эталонной модели TCP/IP**

У модели TCP/IP и ее протоколов также имеется ряд недостатков. Во-первых, в этой модели нет четкого разграничения концепций служб, интерфейсов и протоколов. При разработке программного обеспечения желательно провести четкое разделение между спецификацией и реализацией, что весьма тщательно делает OSI и чего не делает TCP/IP. В результате модель TCP/IP довольно бесполезна при разработке сетей, использующих новые технологии.

Во-вторых, модель TCP/IP отнюдь не является общей и довольно плохо описывает любой стек протоколов, кроме TCP/IP. Так, например, описать технологию Bluetooth с помощью модели TCP/IP совершенно невозможно.

В-третьих, канальный уровень в действительности не является уровнем в том смысле, который обычно используется в контексте уровневых протоколов. Это скорее интерфейс между сетью и уровнями передачи данных. Различие между интерфейсом и уровнем является чрезвычайно важным, и здесь не следует быть небрежным.

В-четвертых, в модели TCP/IP не различаются физический уровень и уровень передачи данных. Об этом различии даже нет упоминания. Между тем, они абсолютно разные. Физический уровень должен иметь дело с характеристиками передачи информации по медному кабелю, оптическому волокну и по радио, тогда как задачей уровня передачи данных является определение начала и конца кадров и передача их с одной стороны на другую с требуемой степенью надежности. Правильная модель должна содержать их как два различных уровня. В модели TCP/IP этого нет.

И, наконец, хотя протоколы IP и TCP были тщательно продуманы и неплохо реализованы, многие другие протоколы были созданы несколькими студентами, работавшими над ними, пока это занятие им не наскучило. Реализации этих протоколов свободно распространялись, в результате чего они получили широкое признание, глубоко укоренились, и теперь их трудно заменить на что-либо другое. Некоторые из них в настоящее время оказались серьезным препятствием на пути прогресса. Например, протокол виртуального терминала TELNET, созданный еще для механического терминала типа Teletype, работавшего с огромной скоростью 10 символов в секунду. Ему ничего не известно о графических интерфейсах пользователя и о мышках. Тем не менее сейчас, 30 лет спустя, он все еще используется.

## 7. Службы на основе соединений и службы без установления соединений.

## Уровни могут предлагать вышестоящим уровням услуги двух типов: с наличием или отсутствием установления соединения. В этом разделе мы рассмотрим, что означает каждый из этих типов и в чем состоит разница между ними. Типичным примером сервиса с установлением соединения является телефонная связь. Чтобы поговорить с кем-нибудь, необходимо поднять трубку, набрать номер, а после окончания разговора положить трубку. Нечто подобное происходит и в компьютерных сетях: при использовании сервиса с установлением соединения абонент сначала устанавливает соединение, а после окончания сеанса разрывает его. Это напоминает трубу: биты сообщения влетают в один ее конец, а вылетают с другого. В большинстве случаев не возникает путаницы с последовательностью передачи этих битов. В некоторых случаях перед началом передачи отправляющая и получающая машины обмениваются приветствиями, отсылая друг другу приемлемые параметры соединения: максимальный размер сообщения, необходимое качество сервиса и др. В большинстве случаев одна из сторон посылает запрос, а другая его принимает, отвергает или же выставляет встречные условия. Линия — другое название соединения со связанными ресурсами, такими как фиксированная пропускная способность. Это название происходит из истории телефонной сети, в которой линия была путем по медному проводу, который переносил телефонный разговор. Противоположный пример — сервисы без установления соединения, типичный пример такой технологии — почтовые системы. Каждое письмо содержит полный адрес назначения и проходит по некому маршруту, который совершенно не зависит от других писем. Есть различные названия для сообщений в различных контекстах; пакет — сообщение на сетевом уровне. Когда промежуточные узлы получают сообщение полностью перед пересылкой его к следующему узлу, это называют коммутацией с промежуточной буферизацией. Другой вариант, когда передача сообщения начинается прежде, чем оно будет полностью получено узлом, называют сквозной передачей. Обычно то письмо, которое отправлено раньше, в место назначения приходит раньше. Тем не менее возможна ситуация, что первое письмо задерживается и раньше приходит то, которое было послано вторым. Каждая служба характеризуется качеством обслуживания. Некоторые службы являются надежными, в том смысле, что они никогда не теряют

## данные. Обычно надежная служба реализуется при помощи подтверждений, посылаемых получателем в ответ на каждое принятое сообщение, так что отправитель знает, дошло очередное сообщение или нет. Процесс пересылки подтверждений требует некоторых накладных расходов и снижает пропускную способность канала. Впрочем, подобные затраты обычно не очень велики и окупаются, хотя иногда могут быть нежелательными.

## 

## 8. **Спецификация стандартов. Три уровня стеков протоколов**

Спецификации Institute of Electrical and Electronics Engineers IEEE802

определяют стандарты для физических компонентов сети. Эти компоненты –

сетевая карта (Network Interface Card – NIC) и сетевой носитель (network

media), которые относятся к физическому и канальному уровням модели OSI.

Спецификации IEEE802 определяют механизм доступа адаптера к каналу связи и

механизм передачи данных. Стандарты IEEE802 подразделяют канальный уровень

на подуровни:

- Logical Link Control (LLC) – подуровень управления логической связью;

- Media Access Control (MAC) – подуровень управления доступом к

устройствам.

Спецификации IEEE 802 делятся на двенадцать стандартов, вот 2 из них:

802.1

Стандарт 802.1 (Internetworking – объединение сетей) задает механизмы

управления сетью на MAC – уровне. В разделе 802.1 приводятся основные

понятия и определения, общие характеристики и требования к локальным сетям,

а также поведение маршрутизации на канальном уровне, где логические адреса

должны быть преобразованы в их физические адреса и наоборот.

802.2

Стандарт 802.2 (Logical Link Control – управление логической связью)

определяет функционирование подуровня LLC на канальном уровне модели OSI.

LLC обеспечивает интерфейс между методами доступа к среде и сетевым

уровнем.

Стек протоколов — это иерархически организованный набор сетевых протоколов, достаточный для организации взаимодействия узлов в сети. Протоколы работают в сети одновременно, значит работа протоколов должна быть организована так, чтобы не возникало конфликтов или незавершённых операций. Поэтому стек протоколов разбивается на иерархически построенные уровни, каждый из которых выполняет конкретную задачу — подготовку, приём, передачу данных и последующие действия с ними.

Количество уровней в стеке меняется в соответствии с конкретным стеком протоколов. Протоколы нижних уровней часто реализуются комбинацией программных и аппаратных средств, а протоколы верхних уровней, как правило, программными средствами.

Важно различать модель OSI и стек протоколов OSI. В то время как модель OSI является концептуальной схемой взаимодействия открытых систем, стек OSI представляет собой набор спецификаций конкретных протоколов.

В отличие от других стеков протоколов, стек OSI полностью соответствует модели OSI, включая спецификации протоколов для всех семи уровней взаимодействия, определённых в этой модели:

Стек протоколов OSI

7. Прикладной X.400 X.500 VTP FTAM JTM Другие

6.

Представления Протокол уровня представления OSI

5. Сеансовый Сеансовый протокол OSI

4.

Транспортный Транспортные протоколы OSI

3. Сетевой ES-IS, IS-IS, CONP, CLNP

2. Канальный Ethernet (OSI-8802.3, IEEE-802.3) Token Bus (OSI-8802.4, IEEE-802.4) Token Ring (OSI-8802.5, IEEE-802.5) X.25 ISDN FDDI (ISO-9314)

1. Физический HDLC LAP-B

· На физическом и канальном уровнях стек OSI поддерживает протоколы Ethernet, Token ring, FDDI, а также протоколы LLC, X.25 и ISDN, то есть использует все разработанные вне стека популярные протоколы нижних уровней, как и большинство других стеков.

· Сетевой уровень включает сравнительно редко используемые протоколы Connection-oriented Network Protocol (CONP) и Connectionless Network Protocol (CLNP). Как следует из названий, первый из них ориентирован на соединение (connection-oriented), второй — нет (connectionless). Более популярны протоколы маршрутизации стека OSI: ES-IS (End System — Intermediate System) между конечной и промежуточной системами и IS-IS (Intermediate System — Intermediate System) между промежуточными системами.

· Транспортный уровень стека OSI в соответствии с функциями, определёнными для него в модели OSI, скрывает различия между сетевыми сервисами с установлением соединения и без установления соединения, так что пользователи получают требуемое качество обслуживания независимо от нижележащего сетевого уровня. Чтобы обеспечить это, транспортный уровень требует, чтобы пользователь задал нужное количество обслуживания.

· Службы прикладного уровня обеспечивают передачу файлов, эмуляцию терминала, службу каталогов и почту. Из них наиболее популярными являются служба каталогов (стандарт X.500), электронная почта (стандарт X.400), протокол виртуального терминала (VTP), протокол передачи, доступа и управления файлами (FTAM), протокол пересылки и управления работами (JTM). TCP/IP[править | править код] Основная статья: TCP/IP

Стек протоколов TCP/IP — набор сетевых протоколов, на которых базируется Интернет. Обычно в стеке TCP/IP верхние 3 уровня (прикладной, представления и сеансовый) модели OSI объединяют в один — прикладной. Поскольку в таком стеке не предусматривается унифицированный протокол передачи данных, функции по определению типа данных передаются приложению.

Уровни стека TCP/IP:

1. Канальный уровень описывает каким образом передаются пакеты данных через физический уровень, включая кодирование (то есть специальные последовательности битов, определяющих начало и конец пакета данных).

2. Сетевой уровень изначально разработан для передачи данных из одной (под)сети в другую. Примерами такого протокола является X.25 и IPC в сети ARPANET. С развитием концепции глобальной сети в уровень были внесены дополнительные возможности по передаче из любой сети в любую сеть, независимо от протоколов нижнего уровня, а также возможность запрашивать данные от удалённой стороны.

3. Протоколы транспортного уровня могут решать проблему негарантированной доставки сообщений («дошло ли сообщение до адресата?»), а также гарантировать правильную последовательность прихода данных.

4. На прикладном уровне работает большинство сетевых приложений. Эти программы имеют свои собственные протоколы обмена информацией, например, HTTP для WWW, FTP (передача файлов), SMTP (электронная почта), SSH (безопасное соединение с удалённой машиной), DNS (преобразование символьных имён в IP-адреса) и многие другие.

Существуют разногласия в том, как вписать модель TCP/IP в модель OSI, поскольку уровни в этих моделях не совпадают. Упрощённо интерпретацию стека TCP/IP можно представить так:

OSI TCP/IP

7. Прикладной HTTP, FTP, Telnet, SMTP, DNS (RIP, работающий поверх UDP, и BGP, работающий поверх TCP, являются частью сетевого уровня), LDAP, RTP Прикладной

6.

Представления

5. Сеансовый

4.

Транспортный TCP, UDP, SCTP, DCCP (протоколы маршрутизации, подобные OSPF, что работают поверх IP, являются частью сетевого уровня) Транспортный

3. Сетевой IP (вспомогательные протоколы, вроде ICMP и IGMP, но являются частью сетевого уровня; ARP не работает поверх IP) Сетевой

2. Канальный Ethernet, Token Ring, и подобные Канальный

1. Физический IPX/SPX[править | править код] Основная статья: IPX/SPX

Название стеку дали протоколы сетевого и транспортного уровней - Internetwork Packet Exchange (IPX) и Sequenced Packet Exchange (SPX). К сетевому уровню этого стека отнесены также протоколы маршрутизации RIP и NLSP. А в качестве представителей трёх верхних уровней на рисунке ниже приведены два популярных протокола: протокол удалённого доступа к файлам NetWare Core Protocol (NCP) и протокол объявления о сервисах Service Advertising Protocol (SAP).

OSI IPX/SPX

7. Прикладной SAP, NCP

6. Представления

5. Сеансовый

4. Транспортный SPX

3. Сетевой IPX, RIP, NLSP

2. Канальный Ethernet, Token Ring, FDDI и другие

1. Физический NetBIOS/SMB[править | править код]

На физическом и канальном уровнях этого стека также задействованы уже получившие распространение протоколы, такие как Ethernet, Token Ring, FDDI, а на верхних уровнях — специфические протоколы NetBEUI (протокол расширенного пользовательского интерфейса NetBEUI — NetBIOS Extended User Interface) и SMB. NetBEUI разрабатывался как эффективный протокол, потребляющий немного ресурсов и предназначенный для сетей, насчитывающих не более 200 рабочих станций. Этот протокол содержит много полезных сетевых функций, которые можно отнести к транспортному и сеансовому уровням модели OSI, однако с его помощью невозможна маршрутизация пакетов. Это ограничивает применение протокола NetBEUI локальными сетями, не разделёнными на подсети, и делает невозможным его использование в составных сетях.

Протокол Server Message Block (SMB) поддерживает функции сеансового уровня, уровня представления и прикладного уровня. На основе SMB реализуется файловая служба. а также службы печати и передачи сообщений между приложениями.

## 9. **Сетевое программное обеспечение, иерархия протоколов**

Сетевое программное обеспечение состоит из нескольких протоколов, каждый из которых представляет собой набор форматов, последовательностей обмена и правил, определяющих назначение пакетов. Например, когда пользователь хочет получить веб-страницу с сервера, его браузер по протоколу HTTP (HyperText Transfer Protocol — протокол передачи гипертекста) отправляет пакет с запросом GET PAGE серверу, который «знает», как обрабатывать полученный пакет. В процессе передачи используются множество различных протоколов, причем зачастую совместно. Обычно они организованы в иерархическую многоуровневую структуру, в которойпротоколы верхнего уровня передают пакеты протоколам нижележащих уровней, а реальной передачей данных занимается протокол самого нижнего уровня. На стороне получателя пакеты поднимаются вверх по иерархии в обратном порядке.

Раз сетевые процессоры заняты выполнением протоколов, перед изучением процессоров имеет смысл поговорить о протоколах немного подробнее. Вернемся ненадолго к запросу GET PAGE. Как именно он попадает на веб-сервер? Прежде всего браузер устанавливает с сервером соединение по протоколу TCP (Transmission Control Protocol — протокол управления передачей). Программное обеспечение, реализующее этот протокол, следит за тем, чтобы все отправленные пакеты были доставлены, причем в правильном порядке. В случае потери пакета программное обеспечение TCP максимально быстро повторяет передачу до тех пор, пока пакет наконец не будет получен.

Браузер формирует корректное HTTP-сообщение с запросом GET PAGE, а затем передает его программному обеспечению TCP, которое и передает пакет через соединение. Программным обеспечением TCP в начало сообщения добавляется заголовок, содержащий порядковый номер и другую информацию. Этот дополнительный заголовок называется TCP-заголовком.

Сделав свою часть работы, программное обеспечение TCP передает TCP-заголовок вместе с полезной нагрузкой (содержащей запрос GET PAGE) еще одной программе, реализующей протокол IP (Internet Protocol — межсетевой протокол). Эта программа добавляет в начало пакета IP-заголовок с информацией об адресах отправителя (то есть машины, передающей пакет) и получателя (машины, ожидающей пакет), максимальном числе переходов, двигаясь вдоль которых пакет будет существовать (чтобы «потерявшиеся» пакеты не жили вечно, заполоняя собой весь Интернет), контрольной суммой (для обнаружения ошибок памяти и ошибок передачи) и рядом других полей.

Далее пакет (включающий в себя IP-заголовок, TCP-заголовок и сам запрос GET PAGE) передается «вниз» на уровень канала передачи данных, который добавляет к пакету свой заголовок и передает пакет по линии связи. Этот уровень также дописывает в конец контрольную сумму, называемую CRC (Cyclic Redundancy Check — циклический контроль избыточности) и позволяющую выявлять ошибки передачи. Может показаться, что две контрольные суммы, на уровне IP и на уровне канала данных, — это больше, чем необходимо, но такой подход повышает надежность. На каждом переходе проверяется CRC-код пакета, после чего заголовок вместе с CRC-кодом генерируются заново, в соответствии с требованиями исходящего канала передачи данных.

Иерархия протоколов

Самый нижний (уровень IV) соответствует уровню доступа к сети. В протоколах ТСР/IР он не регламентируется, но поддерживает все популярные стандарты протоколов физического и канального уровня, такие, как Ethernet, Token Ring, SLIP, PPP и другие (по причинам, которые станут понятны ниже, пока не будем расшифровывать эти аббревиатуры). Протоколы данного уровня обеспечивают передачу пакетов данных в сети на уровне аппаратных средств.

Следующий уровень (уровень III) – это уровень межсетевого взаимодействия, который обеспечивает передачу пакетов данных из одной подсети в другую. В качестве протокола в стеке используется протокол IP.

Следующий уровень (уровень II) называется основным. На этом уровне функционирует протокол управления передачей TCP, который обеспечивает надежную передачу сообщений между удаленными друг от друга различными прикладными программами за счет образования виртуальных соединений между ними.

Все перечисленные выше протоколы с легким сердцем можно отнести к «уровню секретарей» из примера, описанного выше, и, почувствовав себя начальниками, на время забыть о них. Для конечного пользователя («начальника») наиболее необходима компетентность на самом верхнем уровне (уровень I), или «уровне начальников», который называется на языке стека ТСР/IР прикладным.

## 10. **Сравнение эталонных моделей OSI и TCP** У моделей OSI и TCP имеется много общих черт. Обе модели основаны на концепции стека независимых протоколов. Функциональность уровней также во многом схожа. Например, в обеих моделях уровни, начиная с транспортного и выше, предоставляют сквозную, не зависящую от сети транспортную службу для процессов, желающих обмениваться информацией. Эти уровни образуют поставщика транспорта. Также в каждой модели уровни выше транспортного являются прикладными потребителями транспортных сервисов. Несмотря на это фундаментальное сходство, у этих моделей имеется и ряд отличий. Для модели OSI центральными являются три концепции. 1. Службы. 2. Интерфейсы. 3. Протоколы. Вероятно, наибольшим вкладом модели OSI стало явное разделение этих трех концепций. Каждый уровень предоставляет некоторые сервисы для расположенного выше уровня. Сервис определяет, что именно делает уровень, но не то, как он это делает и каким образом объекты, расположенные выше, получают доступ к данному уровню. Интерфейс уровня определяет способ доступа к уровню для расположенных выше процессов. Он описывает параметры и ожидаемый результат. Он также ничего не сообщает о внутреннем устройстве уровня. Наконец, равноранговые протоколы, применяемые в уровне, являются внутренним делом самого уровня. Для выполнения поставленной ему задачи (то есть предоставления сервиса) он может использовать любые протоколы. Кроме того, уровень может менять протоколы, не затрагивая работу приложений более высоких уровней. Эти идеи очень хорошо соответствуют современным идеям объектно-ориентированного программирования. Уровень может быть представлен в виде объекта, обладающего набором методов (операций), к которым может обращаться внешний процесс. Семантика этих методов определяет набор служб, предоставляемых объектом. Параметры и результаты методов образуют интерфейс объекта. Внутреннее устройство объекта можно сравнить с протоколом уровня. За пределами объекта оно никого не интересует и никому не видно. Изначально в модели TCP/IP не было четкого разделения между службами, интерфейсом и протоколами, хотя и производились попытки изменить это, чтобы сделать ее более похожей на модель OSI. Так, например, единственными настоящими сервисами, предоставляемыми межсетевым уровнем, являются SEND IP PACKET (послать IPпакет) и RECEIVE IP PACKET (получить IP-пакет). В результате в модели OSI протоколы скрыты лучше, чем в модели TCP/IP, и при изменении технологии они могут быть относительно легко заменены. Возможность проводить подобные изменения, не затрагивая другие уровни, является одной из главных целей многоуровневых протоколов. Эталонная модель OSI была разработана прежде, чем были изобретены протоколы для нее. Такая последовательность событий означала, что эта модель не была настроена на какой-то конкретный набор протоколов, что делало ее универсальной. Обратной стороной такого порядка действий было то, что у разработчиков было мало опыта в данной области и не было четкого представления о том, какие функции должен выполнять каждый уровень. Например, уровень передачи данных изначально работал только в сетях с передачей от узла к узлу. С появлением широковещательных сетей в модель потребовалось ввести новый подуровень. В дальнейшем, когда на базе модели OSI начали строить реальные сети с использованием существующих протоколов, обнаружилось, что они не соответствуют требуемым спецификациям служб. Поэтому в модель пришлось добавить подуровни для устранения несоответствия. Наконец, изначально ожидалось, что в каждой стране будет одна сеть, управляемая правительством и использующая протоколы OSI, поэтому никто и не думал об объединении различных сетей. В действительности все оказалось не так. С моделью TCP/IP было все наоборот: сначала появились протоколы, а уже затем была создана модель, описывающая существующие протоколы. Таким образом, не было проблемы с соответствием протоколов модели. Они ей соответствовали прекрасно. Единственной проблемой было то, что модель не соответствовала никаким другим стекам протоколов. В результате она не использовалась для

## описания каких-нибудь других сетей, отличных от TCP/IP. Если взглянуть на эти две модели поближе, то, прежде всего, обратит на себя внимание различие в количестве уровней: в модели OSI семь уровней, в модели TCP/ IP — четыре. В обеих моделях имеются межсетевой, транспортный и прикладной уровни, а остальные уровни различные. Еще одно различие между моделями лежит в сфере возможности использования связи на основе соединений и связи без установления соединения. Модель OSI на сетевом уровне поддерживает оба типа связи, а на транспортном уровне — только связь на основе соединений (поскольку транспортные службы являются видимыми для пользователя). В модели TCP/IP на сетевом уровне есть только один режим связи (без установления соединения), но на транспортном уровне она поддерживает оба режима, предоставляя пользователям выбор. Этот выбор особенно важен для простых протоколов запрос-ответ.

## 

## 11. Службы и протоколы

на основе соединений и службы без установления соединений Уровни могут предлагать вышестоящим уровням услуги двух типов: с наличием или отсутствием установления соединения. В этом разделе мы рассмотрим, что означает каждый из этих типов и в чем состоит разница между ними. Типичным примером сервиса с установлением соединения является телефонная связь. Чтобы поговорить с кем-нибудь, необходимо поднять трубку, набрать номер, а после окончания разговора положить трубку. Нечто подобное происходит и в компьютерных сетях: при использовании сервиса с установлением соединения абонент сперва устанавливает соединение, а после окончания сеанса разрывает его. Это напоминает трубу: биты сообщения влетают в один ее конец, а вылетают с другого. В большинстве случаев не возникает путаницы с последовательностью передачи этих битов. В некоторых случаях перед началом передачи отправляющая и получающая машины обмениваются приветствиями, отсылая друг другу приемлемые параметры соединения: максимальный размер сообщения, необходимое качество сервиса и др. В большинстве случаев одна из сторон посылает запрос, а другая его принимает, отвергает или же выставляет встречные условия. Что касается сервисов без установления соединения, то типичный пример такой технологии – почтовые системы. Каждое письмо содержит полный адрес назначения и проходит по некоему маршруту, который совершенно не зависит от других писем. Обычно то письмо, которое отправлено раньше, в место назначения приходит раньше. Каждая служба характеризуется качеством обслуживания. Некоторые службы являются надежными в том смысле, что они никогда не теряют данных. Обычно надежная служба реализуется при помощи подтверждений, посылаемых получателем в ответ на каждое принятое сообщение, так что отправитель знает, дошло очередное сообщение или нет. Процесс пересылки подтверждений требует некоторых накладных расходов и снижает пропускную способность канала. Впрочем, подобные затраты обычно не очень велики и окупаются, хотя иногда могут быть нежелательными. Типичным примером необходимости надежной службы на основе соединений является пересылка файлов. Владелец файла хочет быть уверенным, что все биты файла прибыли без искажений и в том же порядке, в котором были отправлены. Вряд ли кто-нибудь отдаст предпочтение службе, которая случайным образом искажает информацию, даже если передача происходит значительно быстрее. Надежные службы на основе соединений бывают двух типов: последовательности сообщений и байтовые потоки. В первом варианте сохраняются границы между сообщениями. Когда посылаются два сообщения размером по 1 Кбайт, то они прибывают в виде двух сообщений размером по 1 Кбайт и никогда – как одно двух-килобайтное сообщение. При втором варианте связь представляет собой просто поток байтов, без разделения на отдельные сообщения. Когда 2048 байт прибывают к получателю, то нет никакой возможности определить, было это одно сообщение длиной 2 Кбайт, два сообщения длиной 1 Кбайт или же 2048 однобайтных сообщений. Если страницы книги посылаются по сети фотонаборной машине в виде отдельных сообщений, то, возможно, необходимо сохранить границы между сообщениями. С другой стороны, при регистрации с удаленного терминала в системе разделения времени вполне достаточно потока байтов с терминального компьютера. Как уже упоминалось ранее, существуют системы, для которых задержки, связанные с пересылкой подтверждений, неприемлемы. В качестве примера такой системы можно назвать цифровую голосовую связь. В данном случае предпочтительнее допустить шумы на линии или искаженные слова, нежели большие паузы, вызванные отсылкой подтверждений и повторной передачей блоков данных. Аналогично, при проведении видеоконференции отдельные неправильные пикселы окажутся меньшей проблемой, нежели дергающиеся и останавливающиеся кадры. Не все приложения требуют установки соединения. Например, при рассылке рекламы по электронной почте установка связи для пересылки каждого отдельного сообщения нежелательна. Также не требуется в этом случае и 100 процентная надежность, особенно, если это существенно увеличит стоимость. Все, что нужно, – это способ переслать сообщение с высокой вероятностью его получения, но без гарантии. Ненадежная (то есть без подтверждений) служба без установления соединения часто называется службой дейтаграмм, или дейтаграммной службой – по аналогии с телеграфной службой, также не предоставляющей подтверждений отправителю. В других ситуациях бывает желательно не устанавливать соединение для пересылки коротких сообщений, но надежность, тем не менее, существенна. Такая служба называется службой дейтаграмм с подтверждениями. Она подобна отправке заказного письма с подтверждением получения. Получив подтверждение, отправитель уверен, что письмо доставлено адресату, а не потеряно по дороге.

## 12. **коммутация каналов, коммутация пакетов, коммутация сообщений**

## **Коммутация каналов.**

## Когда вы (или ваш компьютер) снимаете телефонную трубку и набираете номер, коммутирующее оборудование телефонной системы отыскивает физический путь, состоящий из кабелей и ведущий от вашего телефона к телефону того, с кем вы связываетесь. Такая система, называемая коммутацией каналов, схематически изображена на рис. 2.37, а. Каждый из шести прямоугольников представляет собой коммутирующую станцию (оконечную или междугородную). В данном примере каждая станция имеет три входных и три выходных линии. Когда звонок проходит через коммутационную станцию, между входной и выходной линиями устанавливается физическое соединение (показано пунктирными линиями). На заре телефонии соединение устанавливалось вручную телефонным оператором, который замыкал две линии проводом с двумя штекерами на концах. С изобретением автоматического коммутатора связана довольно забавная история. Автоматический коммутатор изобрел в XIX веке владелец похоронного бюро Алмон Б. Строуджер (Almon B. Strowger) вскоре после изобретения телефона. Когда кто-либо умирал, родственник умершего звонил городскому телефонному оператору и говорил: «Соедините меня, пожалуйста, с похоронным бюро». К несчастью для мистера Строуджера, в его городе было два похоронных бюро, и жена владельца конкурирующей фирмы как раз работала телефонным оператором. Мистер Строуджер быстро понял, что либо он изобретет автоматический телефонный коммутатор, либо ему придется закрывать дело. Он выбрал первое. На протяжении почти 100 лет используемое во всем мире оборудование для коммутации каналов называлось искателем Строуджера. (История не упоминает, не устроилась ли жена его конкурента, уволенная с работы телефонного оператора, в телефонное справочное агентство сообщать телефонный номер своего похоронного бюро всем желающим.)

## Модель, изображенная на рис. 2.37, a, конечно, сильно упрощена, поскольку канал, соединяющий двух абонентов телефонной линии, на самом деле может быть не только медным проводом, но и, например, микроволновой или оптоволоконной магистралью, на которой объединены тысячи телефонных абонентов. Тем не менее основная идея остается той же самой: когда один абонент звонит другому, устанавливается определенный путь, связывающий их, и этот путь остается неизменным до конца разговора. Важным свойством коммутации каналов является необходимость установления сквозного пути от одного абонента до другого до того, как будут посланы данные. Именно поэтому время от конца набора номера до начала разговора может занимать около 10 с и более для междугородных или международных звонков. В течение этого интервала времени телефонная система ищет путь, изображенный на рис. 2.38, а. Обратите внимание на то, что еще до начала передачи данных сигнал запроса на разговор должен пройти весь путь до пункта назначения и там быть распознан. Для многих компьютерных приложений (например, при проверке кредитной карточки клиента кассовым терминалом) длительное время установления связи является нежелательным.

## 

## В результате при установлении физического соединения между абонентами, как только этот путь установлен, единственной задержкой для распространения сигнала будет скорость распространения электромагнитного сигнала, то есть около 5 мс на каждые 1000 км. Еще одним свойством такой системы является то, что после начала разговора линия уже не может вдруг оказаться занятой, хотя она может быть занятой до установки соединения (например, благодаря отсутствию соответствующей возможности у коммутатора или магистрали). Также следствием установленного пути является отсутствие опасности скопления, то есть как только звонок был передан, вы никогда не получите сигналы «занято». Конечно, такой сигнал вы могли получить прежде, чем соединение было установлено из-за нехватки объема линии или коммутатора.

## 

## **Коммутация пакетов.**

## Альтернативным способом коммутации является коммутация пакетов, которая схематически изображена на рис. 2.37, б и описана в главе 1. При использовании такой формы коммутации отдельные пакеты пересылаются по мере готовности. В отличие от коммутации каналов, при коммутации пакетов нет необходимости устанавливать связь между двумя абонентами до начала передачи данных. Маршрутизаторы должны использовать передачу с промежуточной буферизацией, чтобы самостоятельно послать каждый пакет, продвигающийся к месту назначения. Эта процедура не похожа на коммутацию каналов, когда результат установки соединения — резервирование пропускной способности на всем пути от отправителя до приемника. Все данные в канале следуют этим путем. Среди других результатов следования по данному пути — гарантия прибытия данных в нужном порядке. При коммутации пакетов нет никакого неизменного пути, поэтому различные пакеты могут следовать различными путями, в зависимости от сетевых условий в тот момент, когда они были отправлены, и могут прибыть не по порядку. Сети с пакетной коммутацией устанавливают низкую верхнюю границу размера пакетов. Это гарантирует, что ни один пользователь не может монополизировать линию передачи очень долго (например, много миллисекунд), так что сети с пакетной коммутацией могут обработать интерактивный трафик. Это также уменьшает задержку, так как первый пакет длинного сообщения может быть переправлен прежде, чем второй полностью прибыл. Однако задержка промежуточной буферизации пакета в памяти маршрутизатора, прежде чем он будет переслан к следующему маршрутизатору, превышает задержку при коммутации каналов. При коммутации каналов биты текут по проводу непрерывно. Коммутация пакетов и каналов отличается и другими особенностями. Поскольку никакая пропускная способность при коммутации пакетов не зарезервирована, пакетам, вероятно, придется ждать, чтобы быть отправленными. Это вводит задержку очереди и скопление, если много пакетов послано в одно время. С другой стороны, нет никакой опасности застать сигнал «занято» и не иметь возможности использовать сеть. Таким образом, скопление происходит в разное время при коммутации каналов (во время установки) и пакетной коммутации (когда пакеты посланы). Если канал был зарезервирован для отдельного пользователя и нет никакого трафика, его пропускная способность потрачена впустую. Она не может использоваться для другого трафика. Пакетная коммутация не тратит впустую пропускную способность и таким образом более эффективна с системной точки зрения. Понимание этого выбора крайне важно для понимания различия между коммутацией каналов и пакетной коммутацией. Выбор между гарантируемым сервисом и тратой ресурсов против не гарантированного сервиса и не траты ресурсов. Пакетная коммутация более устойчива к сбоям. На самом деле, именно это свойство стало причиной изобретения данного метода. Если, например, выходит из строя один из коммутаторов, то все линии, подключенные к нему, также выходят из строя. Но при коммутации пакетов данные могут быть отправлены в обход «умершего» коммутатора. Наконец, еще одним различием между двумя способами коммутации является политика оплаты услуг. Системы с коммутацией каналов традиционно взимают плату за 186 Глава 2. Физический уровень расстояние передачи и время на линии. В мобильных телефонах расстояние роли не играет (кроме международных звонков), а время играет не очень значительную роль (ну, например, тариф с 2000 бесплатных минут дороже, чем тариф с 1000 бесплатных минут, причем иногда звонки в ночное время и в выходные являются льготными). В случае коммутации пакетов время на линии вообще не принимается в расчет, однако иногда взимается плата за трафик. С обычных пользователей провайдеры иногда берут просто ежемесячную абонентскую плату, поскольку это проще для обеих сторон, однако магистральные транспортные службы взимают с местных провайдеров плату за объем трафика

## Традиционно телефонные сети использовали схему с коммутацией каналов, чтобы обеспечить высококачественные телефонные звонки, а компьютерные сети использовали пакетную коммутацию для простоты и эффективности. Однако есть известные исключения. Некоторые более старые компьютерные сети имели внутреннюю схему коммутации каналов (например, X.25), а некоторые более новые телефонные сети используют пакетную коммутацию в технологии IP-телефонии. Для пользователя это выглядит точно так же, как стандартный телефонный звонок, но в сети происходит коммутация пакетов голосовых данных. Этот подход позволил возникнуть рынку дешевых международных звонков с помощью телефонных карточек, хотя, возможно, с более низким качеством звонка, чем у должностных лиц.

## 

## **Коммутация сообщений.**

## Коммутация сообщений была преобладающим методом передачи данных в 1960 – 1970 гг. и до сих пор широко используется в некоторых областях (в электронной почте, электронных новостях, телеконференциях, телесеминарах). Как и все методы коммутации с промежуточным хранением, технологии коммутации сообщений относятся к технологии типа «запомнить и послать». Кроме того эта технология обычно предусматривает отношения «главный-подчиненный». Коммутатор (коммутационная ЭВМ) в центре коммутации сообщений (ЦКС) выполняет регистрацию и выбор при управлении входящими и выходящими потоками. Данные через коммутатор могут передаваться на очень высокой скорости с соответствующим определенным уровнем приоритета. Высокоприоритетные сообщения могут задерживаться не более короткое время по сравнению с низкоприоритетными.

## Необходимо отметить, что при коммутации сообщений сообщение, независимо от его длинны целиком сохраняет свою первоначальную целостность как единичный объект при прохождении его от одного пункта к другому вплоть до пункта назначения. Более того, транзитный узел не может начинать дальнейшею передачу части сообщения, если оно еще принимается. По своему влиянию на задержки это равносильно низкому уровню использования ресурсов сети.

## ***Недостатки*** метода коммутации сообщений:

## · необходимость реализации достаточно серьезных требований к емкости буферных ЗУ в узлах связи;

## · недостаточная возможность в реализации диалогового режима и работы в реальном масштабе времени при передачи данных;

## · выход из строя всей сети при отказе коммутатора;

## · коммутатор сообщений является потенциально «узким» местом по пропускной способности;

## · каналы передачи данных используются менее эффективно по сравнению с другими методами коммутаций с промежутками хранения.

## Преимущества метода:

## · отсутствие необходимости заблаговременного установления сквозного канала между абонентами;

## · возможность формирования маршрута из отдельных участков с различной пропускной способностью;

## · возможность сглаживания пиковых нагрузок путем запоминания низкоприоритетных сообщений;

## · отсутствие потерь запросов на обслуживание

## реализация различных систем обслуживания с учетом их приоритетов.

## 

## 13. **Электронная почта. FTP. Протокол эмуляции терминала Telnet**

## Электро́нная по́чта (англ. email, e-mail [iˈmeɪl], от англ. electronic mail) — технология и служба по пересылке и получению электронных сообщений (называемых «письма», «электронные письма» или «сообщения») между пользователями компьютерной сети (в том числе — Интернета)[1].

## Электронная почта по составу элементов и принципу работы практически повторяет систему обычной (бумажной) почты, заимствуя как термины (почта, письмо, конверт, вложение, ящик, доставка и другие), так и характерные особенности — простоту использования, задержки передачи сообщений, достаточную надёжность и, в то же время, отсутствие гарантии доставки.

## Достоинствами электронной почты являются: легко воспринимаемые и запоминаемые человеком адреса вида имя\_пользователя@имя\_домена (например somebody@example.com); возможность передачи как простого текста, так и форматированного, а также произвольных файлов (текстовые документы, медиафайлы, программы, архивы и т. д.[1]); независимость серверов (в общем случае они обращаются друг к другу непосредственно); достаточно высокая надёжность доставки сообщения; простота использования человеком и программами, высокая скорость передачи сообщений. Недостатки электронной почты: наличие такого явления, как спам (массовые рекламные и вирусные рассылки); возможные задержки доставки сообщения (до нескольких суток); ограничения на размер одного сообщения и на общий размер сообщений в почтовом ящике (персональные для пользователей).

## FTP расшифровывается как File Transfer Protocol — протокол передачи файлов. Он отли-чается от других протоколов тем, что если в процессе передачи возникает какая-то ошиб-ка, то процесс останавливается и выводится сообщение для пользователя. Если ошибок не было, значит, пользователь получил именно тот файл, который нужен, в целости и без недостающих элементов.

## По FTP-протоколу можно скачивать что угодно: фильмы, музыку, документы, программы, драйверы и картинки. Сейчас многие производители железа выкладывают драйверы от устройств на FTP-серверы, чтобы их могли скачать все желающие.

## В корпоративной среде FTP используется для организации локального хранилища внут-ренних документов и файлов для работы. Например, там могут храниться видеолекции или архивные сканы документов. Ещё FTP позволяет загружать свои файлы на сервер, чтобы их мог скачать любой желающий.

## Программисты иногда используют такие серверы для обмена файлами и для бэкапов кода, хотя многие для этого предпочитают GIT. Про него ещё поговорим отдельно.

## Клиент и сервер

## Для работы по FTP нужны двое: FTP-сервер и FTP-клиент. Что делает сервер:

## · обеспечивает доступ по логину и паролю к нужным файлам;

## · показывает пользователю только те файлы и папки, которые он может просматривать или загружать в них;

## · следит за качеством передачи и смотрит, чтобы не было ошибок;

## · управляет параметрами соединения в пассивном режиме.

## Так как FTP пришёл к нам из времён UNIX-систем, то любое соединение требует логина и пароля. Если у пользователя его нет, сервер его не пропустит. Но чтобы сделать файлы доступными для всех, используют анонимный режим. В нём логином будет слово anonymous, а паролем — любой адрес электронной почты. Современные браузеры умеют сами заходить на анонимные FTP-серверы и подставлять почту. Со стороны это выглядит так, как будто никакого логина и пароля нет, но они есть.

## Когда запускается FTP-сервер, ему говорят: «Уважаемый сервер, вот список файлов и папок, которые нужно показывать на сервере. Если к тебе постучится пользователь с таким-то логином и паролем, то покажи ему всё, а если с вот таким логином — то дай ему одну только эту папку. Анонимов не пускать». Ещё один обязательный параметр — адрес сервера и порт, по которому будет идти передача файлов.

## Чтобы подключиться к серверу, нужна специальная программа, их ещё называют FTP-клиентами. Для каждой операционной системы есть много своих клиентов, например, FileZilla или CuteFTP. Те, кто работает в Linux-подобных системах, часто используют командную строку.Ну и казалось бы — что мешает передавать файлы, как обычно, через сайты по протоколу HTTP? Полно же сайтов и форумов, на которых лежат файлы, и их можно спокойно скачать.

## А разница вот в чем:

## · Для FTP не нужен сайт, то есть веб-интерфейс. Не нужно запускать веб-сервер, настраи-вать шаблоны вывода списка файлов и поднимать отдельную программу, которая будет нам отдавать эти файлы (типа Вордпресса). FTP — это как доступ к удаленной папке: ты сразу видишь файлы и можешь их качать, без посредников. А в вебе нужна какая-то про-грамма, которая «нарисует» тебе файловую систему и поставит ссылки на файлы.

## · В FTP уже реализованы вопросы авторизации и прав. А в вебе их нужно создавать: напри-мер, ставить тот же Вордпресс и к нему прикручивать плагины с системой доступа. Или настраивать Apache, генерировать ключи доступа, раскладывать конфигурационные фай-лы по папкам — это гораздо менее элегантно, чем настройка FTP.

## · В FTP можно разрешить или запретить отдельным пользователям загружать файлы на FTP-сервер. В вебе загрузка файлов от пользователя на сервер — это на порядок более сложная задача.

## Протокол эмуляции терминала Telnet.

## С помощью этого протокола пользователь сети Internet может работать на удаленном компьютере. Связь устанавливается при обращении к Telnet-программе командой

## telnet: <имя базы данных или системы каталогов>

## или <имя удаленного компьютера S>

## После установления связи все, что пользователь набирает на клавиатуре своего компьютера, передается в S, а содержимое экрана S отображается на экране пользователя. Для возвращения в свой компьютер (т.е. в командный режим клиентской программы Тelnet) нужно нажать соответствующую клавишу (Ctrl-). Примерами команд в клиентской программе могут служить: установление связи (open), возвращение в командный режим (close), завершение работы (quit). Передача сообщений при работе с Telnet осуществляется с помощью средств FTP.

## Telnet должен иметь возможность работать в условиях разных аппаратных платформ клиента и сервера. Это требование выполняется через промежуточный виртуальный терминал (аналогично SQL сервису в ODBC). В терминале зафиксирована интерпретация различных символов управления, поскольку их разновидностей не так уж много.

## Необходимо предусматривать выход из блокировок, возникающих, например, вследствие зацикливания процесса на сервере; он осуществляется очисткой серверного буфера.

## 

## 14. **IP-адреса и маски подсети Расчет и построение сетей по маске. Сетевые IPv6-адреса**

IP-адрес — уникальный сетевой адрес узла в компьютерной сети, построенной на основе стека протоколов TCP/IP.

Теоретически IPv4-адресов может быть: 232 = 210\*210\*210\*22 = 1024\*1024\*1024\*4 ≈ 1000\*1000\*1000\*4 = 4 млрд. Всего 4 миллиарда. Но дальше будет рассмотрено, сколько из них не используется, грубо говоря, съедается. Как записывается IPv4-адрес? Он состоит из четырёх октетов и записывается в десятичном представлении без начальных нулей, октеты разделяются точками: например, "192.168.11.10".

Маска подсети — битовая маска для определения по IP-адресу адреса подсети и адреса узла (хоста, компьютера, устройства) этой подсети. В отличие от IP-адреса маска подсети не является частью IP-пакета.

Маска подсети — это тоже 32-бита. Но, в отличие от IP-адреса, нули и единицы в ней не могут чередоваться. Всегда сначала идут единицы, потом нули.

· Не может быть маски 120.22.123.12=01111000.00010110.01111011.00001100.

· Но может быть маска 255.255.248.0=11111111.11111111.11111000.00000000. В 6-й версии IP-адрес (IPv6) является 128-битным. Внутри адреса разделителем является двоеточие. Ведущие нули допускается в записи опускать. Нулевые группы, идущие подряд, могут быть опущены, вместо них ставится двойное двоеточие. Более одного такого пропуска в адресе не допускается.

Групповые IPv6-адреса аналогичны групповым IPv4-адресам. Как вы помните, групповой адрес используется для отправки одного пакета по одному или нескольким адресам назначения (группе многоадресной рассылки). Групповые IPv6-адреса имеют префикс FF00::/8.

Примечание: Групповые адреса могут быть только адресами назначения, а не адресами источника.

Существует два типа групповых IPv6-адресов:

· Присвоенный групповой адрес.

· Групповой адрес для поиска узла.’

Сейчас, почти сорок лет спустя, IP составляет основу современной сети Интернет. Мы уже говорили о нескольких типах устройств для соединения сетей. Среди них повторители, концентраторы, мосты, коммутаторы и шлюзы. Повторители и концентраторы просто переносят биты с одного кабеля на другой. Чаще всего это аналоговые устройства, ничего не смыслящие в протоколах вышележащих уровней. Мосты и коммутаторы работают на канальном уровне. Они могут использоваться для построения сетей, осуществляя по ходу дела минимальные преобразования протоколов, например, между сетями Ethernet со скоростями 10, 100 и 1000 Мбит/с. В этом разделе в центре нашего внимания будут устройства сетевого взаимодействия, работающие на сетевом уровне, то есть маршрутизаторы. Шлюзы, являющиеся высокоуровневыми интеркоммуникационными устройствами, будут рассмотрены позднее

## 15. **Подуровень управления доступом к среде: Протоколы коллективного доступа(симплексные). Протоколы без столкновений Протоколы множественного доступа с контролем несущей (CSMA)/CD.**

· Протоколы коллективного доступа:

Система ALOHA-чистая и дискретная (Норман Абрамсон и ученые Гавайского университета)

Найденное решение основывалось на использовании радиосистемы ближнего радиуса действия. Терминал каждого пользователя передавал кадры на центральный компьютер в пределах общей полосы частот. Также присутствовал простой и элегантный метод решения проблемы распределения каналов. Хотя в работе Абрамсона, получившей название системы ALOHA, использовалась широковещательная радиосвязь со стационарными передатчиками, основная идея применима к любой системе, в которой независимые пользователи соревнуются за право использования одного общего канала.

Чистая система ALOHA-В основе системы ALOHA лежит простая идея: разрешить пользователям передачу, как только у них появляются данные для отсылки. Конечно, при этом будут столкновения, и столкнувшиеся кадры будут разрушены. Отправителям необходимо уметь обнаруживать такие ситуации. В системе ALOHA, после того как каждая станция отправляет свой кадр центральному компьютеру, этот компьютер рассылает полученный кадр на все остальные станции. Отправитель прослушивает широковещательную передачу, чтобы понять, насколько успешной была передача. В других системах, таких как проводные локальные сети, у отправителя может быть возможность распознавать коллизии во время передачи. Если кадр был уничтожен, отправитель просто выжидает некоторое случайное время и пытается переслать этот кадр снова. Время ожидания должно быть случайным. В противном случае, при равных фиксированных интервалах времени ожидания коллизии будут повторяться снова и снова. Системы, в которых несколько пользователей использует один общий канал таким способом, что время от времени возникают конфликты, называются системами с конкуренцией. Кадр не пострадает от коллизии в том случае, если в течение интервала времени его передачи не будет послано больше ни одного кадра.

Дискретная система ALOHA - Робертс опубликовал описание метода, позволяющего удвоить производительность систем ALOHA. Его предложение заключалось в разделении времени на дискретные интервалы, называемые слотами (или тактами), соответствующие времени одного кадра. При таком подходе пользователи должны согласиться с определенными временными ограничениями. Одним из способов достижения синхронизации является установка специальной станции, испускающей синхронизирующий сигнал в начале каждого интервала. В системе Робертса станция не может начинать передачу сразу после ввода пользователем строки. Вместо этого она должна дождаться начала нового такта. Уязвимый временной интервал теперь становится в два раза короче. Поскольку число попыток передачи для одного кадра экспоненциально зависит от количества попыток передачи в единицу времени, небольшое увеличение нагрузки в канале может сильно снизить его производительность.

· Протоколы множественного доступа с контролем несущей:

Протоколы, в которых станции прослушивают среду передачи данных и действуют в соответствии с этим, называются протоколами с контролем несущей.

Настойчивый и ненастойчивый CSMA:

Первый протокол с опросом несущей называется 1-настойчивый протокол CSMA (Carrier-Sense Multiple Access — множественный доступ с контролем несущей). Когда у станции появляются данные для передачи, она сначала прослушивает канал, проверяя, свободен он или занят. Если канал бездействует, то станция отправляет данные. В противном случае, когда канал занят, станция ждет, пока он освободится. Затем станция передает кадр. Если происходит столкновение, станция ждет в течение случайного интервала времени, затем снова прослушивает канал и, если он свободен, пытается передать кадр еще раз. Такой протокол называется протоколом CSMA с настойчивостью 1, так как станция передает кадр с вероятностью 1, как только обнаружит, что канал свободен. Существует небольшая вероятность того, что как только станция начнет передачу, друга я станция также окажется готовой к передаче и опросит канал. Если сигнал от первой станции еще не успел достичь второй станции, вторая станция решит, что канал свободен, и также начнет передачу, результатом чего будет коллизия. Чем больше время распространения сигнала, тем выше вероятность столкновений и ниже производительность протокола.

Вторым протоколом с опросом несущей является ненастойчивый протокол CSMA. В данном протоколе предпринята попытка сдержать стремление станций начинать передачу, как только освобождается канал. Прежде чем начать передачу, станция опрашивает канал. Если никто не передает в данный момент по каналу,

станция начинает передачу сама. Однако если канал занят, станция не ждет освобождения канала, постоянно прослушивая его и пытаясь захватить сразу, как только он освободится. Вместо этого станция ждет в течение случайного интервала времени, а затем снова прослушивает линию.

Третий протокол- это протокол CSMA с настойчивостью p. Он применяется в дискретных каналах и работает следующим образом. Когда станция готова передавать, она опрашивает канал. Если канал свободен, она с вероятностью p начинает передачу. С вероятностью q = 1 – p она отказывается от передачи и ждет начала следующего такта. Этот процесс повторяется до тех пор, пока кадр не будет передан или какая-либо другая станция не начнет передачу. В последнем случае станция ведет себя так же, как в случае столкновения. Она ждет в течение случайного интервала времени, после чего начинает все снова. Если при первом прослушивании канала он оказывается занят, станция ждет следующего интервала времени, после чего применяется тот же алгоритм.

Протокол CSMA с обнаружением коллизий:

Протокол CSMA/CD (Carrier-Sense Multiple Access with Collision Detection — множественный доступ с контролем несущей и обнаружением коллизий) является основой чрезвычайно популярных ЛВС Ethernet. В момент времени одна из станций закончила передачу кадра. Все остальные станции, готовые к передаче, теперь могут попытаться передать свои кадры. Если две станции или более одновременно начнут передачу, то произойдет столкновение. Обнаружив коллизию, станция прекращает передачу, ждет случайный период времени, после чего пытается снова, при условии, что к этому моменту не начала передачу другая станция. Таким образом, наша модель протокола CSMA/CD будет состоять из чередования периодов конкуренции и передачи, а также периодов простоя канала (когда все станции молчат). Различие между CSMA/CD и дискретной системой ALOHA состоит в том, что в первом случае за слотом, в течение которого передачу осуществляет только одна станция (то есть когда канал захвачен), следует передача оставшейся части кадра. Это позволит значительно улучшить производительность, если время кадра будет намного больше времени распространения сигнала по каналу.

· Протоколы без столкновений:

Протокол битовой карты:

В первом протоколе без столкновений, называющемся основным методом битовой карты (basic bit-map method), каждый период конкуренции состоит ровно из N временных интервалов. Если у станции 0 есть кадр для передачи, она передает единичный бит во время 0-го интервала. Другим станциям не разрешается передача в это время. Во время интервала 1 станция 1 также сообщает, есть ли у нее кадр для передачи, передавая бит 1 или 0. В результате к окончанию интервала N все N станций знают, кто хочет передавать. В этот момент они начинают передачу в соответствии со своим порядком номеров. Поскольку все знают, чья очередь передавать, столкновений нет. Протоколы, в которых намерение передавать объявляется всем перед самой передачей, называются протоколами с резервированием (reservation protocols), так как они заранее резервируют канал для определенной станции, предотвращая коллизии.

Передача маркера:

Этот способ основан на передаче небольшого сообщения, называемого маркером (token), от одной станции к следующей в том же самом заранее определенном порядке. Маркер представляет собой разрешение на отправку. Если на станции в очереди находится кадр, готовый к пересылке, и станция получает маркер, она имеет право отправить кадр, прежде чем передавать маркер следующей станции. Если кадров для отправки нет, то она просто передает маркер. В протоколе маркерного кольца (token ring) для определения порядка, в котором станции отправляют данные, используется топология сети. Станции подключены одна к другой, образуя простое кольцо. Таким образом, передача маркера заключается в получении его с одного направления и пересылке в противоположном. Кадры передаются в том же направлении, что и маркер. Они путешествуют по кольцу, проходя по всем станциям, которые оказываются на их пути. Однако для того чтобы кадр не циркулировал вечно (как маркер), какая-то станция должна извлечь его из кольца. Это может быть либо первоначальный отправитель (если кадр прошел полный цикл), либо станция-получатель. Канал, соединяющий станции, может иметь форму одной длинной шины. Станции просто пересылают маркер по шине соседям в предопределенном порядке. Наличие маркера позволяет станции использовать шину для отправки одного кадра, как и раньше. Такой протокол называется маркерной шиной (token bus

Двоичный обратный отсчет:

Используя двоичный адрес станции, можно улучшить эффективность канала. Станция, желающая занять канал, объявляет свой адрес в виде битовой строки, начиная со старшего бита. Предполагается, что все адреса станций имеют одинаковую длину. Будучи отправленными одновременно, биты адреса в каждой позиции

логически складываются (логическое ИЛИ) средствами канала. Это - протокол с двоичным обратным отсчетом (binary countdown). Предполагается, что задержки распространения сигнала пренебрежимо малы, поэтому станции слышат утверждаемые номера практически мгновенно. Во избежание конфликтов следует применить правило арбитража: как только станция с 0 в старшем бите адреса видит, что в суммарном адресе этот 0 заменился единицей, она сдается и ждет следующего цикла. Данный метод предполагает, что приоритет станции напрямую зависит от ее номера. В некоторых случаях такое жесткое правило может играть положительную, в некоторых - отрицательную роль.

## 16. **IP, IP4**

Internet Protocol или IP— маршрутизируемый протокол сетевого уровня модели стека протоколов TCP/IP. Благодаря протоколу IP существует нынешняя сеть Интернет, поскольку именно этот протокол стал связующей нитью между разрозненными компьютерными сетями во всем мире. На данный момент нам нужно выделить два ключевых понятия в протоколе IP: минимально единицей измерения данных здесь является IP-пакет (о единицах измерения в компьютерных сетях), который чаще всего инкапсулируется в Ethernet кадр, а также каждый узел на сетевом уровне в модели TCP/IP должен иметь IP-адрес.

IPv4, цифра четыре здесь не означает четыре октета в IP-адресе, просто так случайно вышло. Больше всего нас интересует IP-адрес, под него в IPv4 выделено четыре байта или октета, как известно, в байте 8 бит, то есть восемь двоичных значений (0 или 1), следовательно, максимально возможное десятичное число равно 255, минимально допустимое 0. Думаю, приводить примеры IP-адресов для IPv4 не нужно, вам они уже знакомы. Более детальное описание у нас будет в отдельной теме.

Обмен информацией в IPv4 происходит при помощи IP-пакетов, у данной версии протокола этот пакет делится на два больших поля: поле данных, в котором переносится полезная информация и заголовок, в котором заложен весь функционал протокола, заголовок пакета IPv4 содержит 14 полей, тринадцать из которых обязательные и одно опциональное. Вообще, протокол IPv4 дает нам в распоряжение 2 в 32 степени IP-адресов или же 4 294 967 296. Но этих адресов уже начинает не хватать, дело все в том, что этот протокол был разработан 1980-ом году, тогда это число выглядело ужасающе большим, сейчас во времена интернет-чайников и тостеров со встроенным Wi-Fi этого пространства начинает не хватать.

Осознание того, что 4.2 млрд адресов не хватит начало приходить в 90-ых годах, а в 1996 году появился протокол IPv6, который должен когда-нибудь заменить IPv4. Дело всё в том, что под IP-адрес в IPv6 выделено 128 бит или 16 байт, а это уже совсем другая история. Теперь давайте попытаемся немного по сравнивать IPv4 и IPv6. Во-первых, IPv6 делает такую технологию, как NAT в текущих условиях бесполезной (для кого-то это плюс, а для кого-то это минус, поскольку NAT не только позволяет «перебивать» много частных IP-адресов в один публичный, но является первой линией защиты вашей компьютерной сети).

Маршрутизация в чистом своем виде (а есть и нечистые маршрутизации, которые очень ускоряют процесс обработки пакетов роутером) в IPv6 стала быстрее, чем в IPv4 даже несмотря на то, что адрес IPv6 значительно больше, дело все в том, что количество полей в IPv6 стало меньше, хотя сам заголовок оказался несколько длиннее, также немного изменен алгоритм обработки IPv6 пакетов маршрутизатором. Сейчас не будем далее углубляться в IPv6, а лучше перейдем к следующим темам, в которых мы разберемся с IPv4.

## 

## 

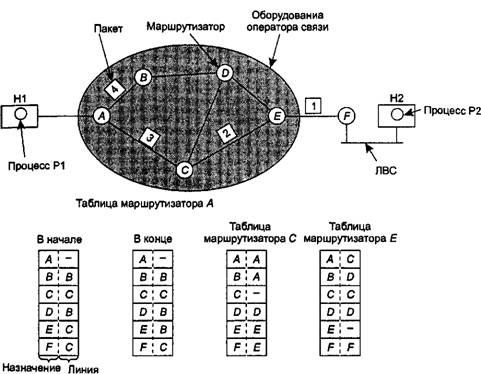
## **17. Реализация сервиса без установления соединения**

Рассмотрев два класса сервисов, которые сетевой уровень может предоставлять своим пользователям, можно перейти к обсуждению устройства этого уровня. Возможны два варианта в зависимости от типа сервиса. Если предоставляется сервис без установления соединения, пакеты внедряются в сеть по отдельности, и их маршруты рассчитываются независимо. При этом никакой предварительной настройки не требуется. В этом случае пакеты часто называют дейтаграммами (datagrams), по аналогии с телеграммами, а сети соответственно дейтаграммными (datagram network). При использовании сервиса, ориентированного на соединение, весь путь от маршрутизатора-отправителя до маршрутизатора-получателя должен быть установлен до начала передачи каких-либо пакетов данных. Такое соединение называется виртуальным каналом (VC, Virtual Circuit), по аналогии с физическими каналами, устанавливаемыми в телефонной системе. Сеть при этом называется сетью виртуального канала (virtual-circuit network). В этом разделе мы обсудим дейтаграммные сети; в следующем разделе — сети виртуального канала.

Рассмотрим принцип работы дейтаграммных сетей. Пусть процесс P1 (рис. 5.2) хочет послать длинное сообщение для Р2. Он передает свое послание транспортному уровню, сообщает ему о том, что доставить данные необходимо процессу Р2, выполняющемуся на хосте Н2. Код транспортного уровня исполняется на хосте Н1; более того, обычно он является частью операционной системы. Заголовок транспортного уровня вставляется в начало сообщения, и в таком виде оно передается на сетевой уровень. Обычно это просто еще одна процедура операционной системы.

Предположим, что в нашем примере сообщение в четыре раза длиннее максимального размера пакета, поэтому сетевой уровень должен разбить его на четыре пакета (1, 2, 3 и 4) и послать их все поочередно на маршрутизатор А с использованием какогонибудь протокола двухточечного соединения, например PPP. Здесь вступает в игру интернет-провайдер. Каждый маршрутизатор имеет свою внутреннюю таблицу, по которой он определяет дальнейший путь пакета при каждом из возможных адресов назначения. Каждая запись таблицы состоит из двух полей: пункт назначения (адресат) и выходящая линия для данного адресата. Во втором поле могут использоваться только линии, непосредственно соединенные с данным маршрутизатором. Так, например, на рис. 5.2 у маршрутизатора А имеются только две исходящие линии — ведущие к В и к С, — поэтому все входящие пакеты должны пересылаться на какой-то из этих двух маршрутизаторов, даже если они не являются адресатами. Изначальная таблица маршрутизации А показана на рисунке под соответствующей надписью.

В маршрутизаторе A пакеты 1, 2 и 3, поступившие на вход, кратковременно сохраняются для верификации контрольной суммы. Затем в соответствии с таблицей А каждый пакет пересылается по исходящему соединению на маршрутизатор С с использованием нового кадра. После этого пакет 1 уходит на Е, откуда доставляется на маршрутизатор локальной сети, F. Когда он прибывает на F, он передается внутри кадра по ЛВС на хост Н2. Пакеты 2 и 3 следуют по тому же маршруту.



Однако с пакетом 4 связана несколько иная история. После прибытия на А он пересылается на маршрутизатор В, несмотря на то что адресом назначения является F, как и у первых трех пакетов. По каким-то своим причинам маршрутизатор А решил послать пакет 4 по новому маршруту. Может быть, это стало следствием затора где-то на линии ACE, возникшего при пересылке трех пакетов, в результате чего маршрутизатор решил обновить свою таблицу (на рисунке показана под надписью «В конце»). Алгоритм, управляющий таблицами маршрутизации и принимающий решения, называется алгоритмом маршрутизации (routing algorithm). Именно изучению алгоритмов маршрутизации будет уделено основное внимание в этой главе. Как мы увидим, существует несколько типов таких алгоритмов.

IP (Internet Protocol, «межсетевой протокол»), составляющий основу всей сети Интернет, является наиболее ярким примером сетевого сервиса без установления соединения. Каждый пакет содержит IP-адрес назначения, с помощью которого маршрутизатор осуществляет индивидуальную отправку пакета. В пакетах IPv4 используются адреса длиной 32 бита, а в IPv6 — 128 бит. Более подробно о протоколах IP мы поговорим далее в этой главе.

Алгоритмы маршрутизации

Основная функция сетевого уровня заключается в выборе маршрута для пакетов от начальной до конечной точки. В большинстве сетей пакетам приходится проходить через несколько маршрутизаторов. Единственным исключением являются широковещательные сети, но даже в них маршрутизация является важным вопросом, если отправитель и получатель находятся в разных сегментах сети. Алгоритмы выбора маршрутов и используемые ими структуры данных являются значительной областью при проектировании сетевого уровня.

Алгоритм маршрутизации реализуется той частью программного обеспечения сетевого уровня, которая отвечает за выбор выходной линии для отправки пришедшего пакета. Если сеть использует дейтаграммную службу, выбор маршрута для каждого пакета должен производиться заново, так как оптимальный маршрут мог

измениться. Если сеть использует виртуальные каналы, маршрут выбирается только при создании нового виртуального канала. После этого все информационные пакеты следуют по установленному маршруту. Последний случай иногда называют сеансовой маршрутизацией (session routing), так как маршрут остается в силе на протяжении всего сеанса связи (например, все время, пока вы подключены к сети VPN).

Полезно понимать разницу между маршрутизацией, при которой системе приходится делать выбор определенного маршрута следования, и пересылкой — действием, происходящим при получении пакета. Можно представить себе маршрутизатор как устройство, в котором функционируют два процесса. Один из них обрабатывает приходящие пакеты и выбирает для них по таблице маршрутизации исходящую линию. Такой процесс называется пересылкой (forwarding). Второй процесс отвечает за заполнение и обновление таблиц маршрутизации. Именно здесь в игру вступает алгоритм маршрутизации.

Вне зависимости от того, отдельно ли выбираются маршруты для каждого отправляемого пакета или же только один раз для соединения, желательно, чтобы алгоритм выбора маршрута обладал определенными свойствами — корректностью, простотой, надежностью, устойчивостью, справедливостью и эффективностью. Корректность и простота вряд ли требуют комментариев, а вот потребность в надежности не столь очевидна с первого взгляда. Во время работы большой сети постоянно происходят какие-то отказы аппаратуры и изменения топологии. Алгоритм маршрутизации должен уметь справляться с изменениями топологии и трафика без необходимости прекращения всех задач на всех хостах. Представьте себе, что было бы, если бы сеть перезагружалась при каждой поломке маршрутизатора!

Алгоритм маршрутизации должен также обладать устойчивостью. Существуют алгоритмы выбора маршрута, никогда не сходящиеся к фиксированному набору путей, независимо от того, как долго они работают. Устойчивый алгоритм должен достигать состояния равновесия и оставаться в нем. Но он также должен быстро находить этот набор путей, так как соединение может быть прервано до того, как будет достигнуто равновесие.

Такие цели, как справедливость и эффективность, могут показаться очевидными — вряд ли кто-нибудь станет возражать против них, — однако они зачастую оказываются взаимоисключающими. Для примера рассмотрим ситуацию, показанную на рис. 5.4. Предположим, что трафик между станциями A и A', B и B', а также C и C' настолько интенсивный, что горизонтальные линии связи оказываются полностью насыщенными. Чтобы максимизировать общий поток данных, трафик между станциями X и X' следовало бы совсем отключить. Однако станции X и X', скорее всего, имеют другую точку зрения по данному вопросу. Очевидно, необходим компромисс между справедливым выделением трафика всем станциям и оптимальным использованием канала в глобальном смысле.

РИСУНОК НЕ СМОГ СКАЧАТЬ

Прежде чем пытаться искать приемлемое соотношение справедливости и эффективности, следует решить, что именно мы будем стремиться оптимизировать. Для увеличения эффективности передачи данных по сети можно попробовать минимизировать среднее время задержки или увеличить общую пропускную способность сети. Однако эти цели также противоречат друг другу, поскольку работа любой системы с очередями вблизи максимума производительности предполагает долгое стояние в очередях. В качестве компромисса многие сети стараются

минимизировать расстояние, которое должен пройти пакет, или просто снизить количество пересылок для каждого пакета. В обоих случаях время прохождения каждого пакета по сети снижается, в результате чего улучшается пропускная способность всей сети.

Алгоритмы выбора маршрута можно разбить на два основных класса: неадаптивные и адаптивные. Неадаптивные алгоритмы не учитывают при выборе маршрута топологию и текущее состояние сети и не измеряют трафик на линиях. Вместо этого выбор маршрута для каждой пары станций производится заранее, в автономном режиме, и список маршрутов загружается в маршрутизаторы во время загрузки сети. Такая процедура иногда называется статической маршрутизацией (static routing). Поскольку статическая маршрутизация не реагирует на сбои, она, как правило, используется в тех случаях, когда выбор маршрута очевиден. Например, маршрутизатор F на рис. 5.3 должен отправлять пакеты, передаваемые по сети, на маршрутизатор E независимо от конечного адреса назначения.

Адаптивные алгоритмы, напротив, изменяют решение о выборе маршрутов при изменении топологии и также иногда в зависимости от загруженности линий. Эти динамические алгоритмы маршрутизации (dynamic routing algorithms) отличаются источниками получения информации (такие источники могут быть, например, локальными, если это соседние маршрутизаторы, либо глобальными, если это вообще все маршрутизаторы сети), моментами изменения маршрутов (например, при изменении топологии или через определенные равные интервалы времени при изменении нагрузки) и данными, использующимися для оптимизации (расстояние, количество транзитных участков или ожидаемое время пересылки).

В следующих разделах мы обсудим различные алгоритмы маршрутизации. Помимо отправки пакета от источника к месту назначения такие алгоритмы предусматривают модель предоставления информации. Иногда требуется отправить пакет на несколько адресов из заданного списка, на все такие адреса или на один из них. Все алгоритмы, о которых мы будем здесь говорить, принимают решения на основании топологии; вопрос о возможности учета интенсивности передачи данных мы оставим до раздела 5.3.

5.2. Алгоритмы маршрутизации . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 392 5.2.1. Принцип оптимальности маршрута . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 394 5.2.2. Алгоритм нахождения кратчайшего пути . . . . . . . . . . . . . . . . . . . . . . . . . . 395 5.2.3. Заливка . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 398 5.2.4. Маршрутизация по вектору расстояний . . . . . . . . . . . . . . . . . . . . . . . . . . . 399 5.2.5. Маршрутизация с учетом состояния линий . . . . . . . . . . . . . . . . . . . . . . . . 403 5.2.6. Иерархическая маршрутизация . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 409 5.2.7. Широковещательная маршрутизация . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 411 5.2.8. Многоадресная рассылка . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 413 5.2.9. Произвольная маршрутизация . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 416 5.2.10. Алгоритмы маршрутизации для мобильных хостов . . . . . . . . . . . . . . . . 417 5.2.11. Маршрутизация в произвольных сетях . . . . . . . . . . . . . . . . . . . . . . . . . . . 420

## 

## 18. **Транспортный уровень.**

Конечная цель транспортного уровня заключается в предоставлении эффективных, надежных и экономичных услуг (сервисов) передачи данных своим пользователям, которыми обычно являются процессы прикладного уровня. Для достижения этой цели транспортный уровень пользуется услугами, предоставляемыми сетевым уровнем. Программа и/или аппаратура, выполняющая работу транспортного уровня, называется транспортной подсистемой или транспортным объектом (transport entity). Транспортная подсистема может располагаться в ядре операционной системы, в библиотечном модуле, загруженном сетевым приложением, в отдельном пользовательском процессе или даже в сетевой интерфейсной плате. Первые два варианта чаще всего встречаются в сети Интернет.

Транспортные протоколы Интернета: UDP В Интернете нашли применение два основных протокола транспортного уровня, один из которых требует установления соединения, другой — нет. Эти протоколы дополняют друг друга. Протоколом без установления соединения является UDP. Он не делает практически ничего, кроме отправки пакетов между приложениями, позволяя последним надстраивать свои собственные протоколы. TCP, напротив, является протоколом с установлением соединения. В его задачи входит практически все. Он устанавливает соединения и обеспечивает надежность сети, выполняя повторную передачу данных, а также осуществляет управление потоком данных и контроль перегрузки — и все это от лица приложений, которые его используют.Транспортные протоколы Интернета: UDP 575 В следующих разделах мы изучим UDP и TCP. Так как UDP проще, его мы изучим первым. Мы также рассмотрим два практических применения UDP. Поскольку протокол транспортного уровня UDP обычно запускается в операционной системе, а протоколы, использующие UDP, — в пространстве пользователя, эти применения вполне можно считать приложениями. Однако методы, которые в них используются, оказываются полезными для многих приложений и их лучше считать частью транспортного сервиса. Поэтому о них мы тоже поговорим.

Стр 574 https://vk.com/doc4262442\_326138355?hash=0d044433066e15d76c&dl=914dea340c3dc6e096

Среди набора протоколов Интернета есть транспортный протокол без установления соединения, UDP (User Datagram Protocol — протокол передачи дейтаграмм пользователя). UDP позволяет приложениям отправлять инкапсулированные IP-дейтаграммы без установления соединений. UDP описан в RFC 768.

Транспортные протоколы Интернета: TCP UDP является простым протоколом и имеет очень важную область применения. В первую очередь, это клиент-серверные взаимодействия и мультимедиа. Тем не менее большинству интернет-приложений требуется надежная, последовательная передача. UDP не удовлетворяет этим требованиям, поэтому необходим иной протокол. Такой протокол называется TCP, и он является рабочей лошадкой Интернета. Ниже мы рассмотрим его детально. Протокол TCP (Transmission Control Protocol — протокол управления передачей) был специально разработан для обеспечения надежного сквозного байтового потока по ненадежной интерсети. Объединенная сеть отличается от отдельной сети тем, что ее различные участки могут обладать сильно различающейся топологией, пропускной способностью, значениями времени задержки, размерами пакетов и другими параметрами. При разработке TCP основное внимание уделялось способности протокола адаптироваться к свойствам объединенной сети и отказоустойчивости при возникновении различных проблем. СТР 586 https://vk.com/doc4262442\_326138355?hash=0d044433066e15d76c&dl=914dea340c3dc6e096

Транспортные протоколы реального масштаба времени Клиент-серверный удаленный вызов процедур — это та область, в которой UDP применяется очень широко. Еще одной такой областью являются мультимедийные приложения реального времени. В частности, с ростом популярности интернетрадио, интернет-телефонии, музыки и видео по заказу, видеоконференций и других мультимедийных приложений стало понятно, что все они пытаются заново изобрести велосипед, используя более или менее одинаковые транспортные протоколы реального времени. Вскоре пришли к мысли о том, что было бы здорово иметь один общий транспортный протокол для мультимедийных приложений. Так появился на свет протокол RTP (Real-Time Transport Protocol — транспортный протокол реального масштаба времени). Он описан в RFC 3550 и ныне широко используется для мультимедийных приложений. Мы рассмотрим два аспекта транспортировки данных в реальном времени. Первый из них — протокол RTP, использующийся для пакетной передачи аудио и видео. Второй аспект — обработка данных, необходимая для воспроизведения (в основном со стороны получателя). Эти функции входят в стек протоколов, изображенный на рис. 6.26. RTP обычно работает поверх UDP (в операционной системе). Делается это так. Мультимедийное приложение может состоять из нескольких аудио-, видео-, текстовых и некоторых других потоков. Они прописываются в библиотеке RTP, которая, как и само приложение, находится в пользовательском пространстве. Библиотека уплотняет потоки и помещает их в пакеты RTP, которые, в свою очередь, отправляются в сокет. На другом конце сокета (в операционной системе) генерируются UDP-пакеты, в которые помещаются RTP-пакеты. Они передаются протоколу IP для отправки по каналу (например, Ethernet). На хосте-получателе происходит обратный процесс. В конечном итоге мультимедийное приложение получает данные из RTP-библиотеки. Оно отвечает за воспроизведение мультимедиа

## 19. **Алгоритмы борьбы с перегрузкой**

Подходы к борьбе с перегрузкой

Наличие перегрузки означает, что нагрузка на сеть (временно) превышает возможности (сетевых) ресурсов. Соответственно, существует два возможных решения:

увеличить ресурсы или снизить нагрузку.

Самый простой способ избежать перегрузки — построить такую сеть, которая

лучше всего соответствует передаваемому по ней трафику. Если часть пути, по которому обычно пересылаются большие объемы данных, обладает низкой пропускной

способностью, возникновение перегрузки очень вероятно. Иногда при возникновении

перегрузки возможно динамическое добавление ресурсов, например подключение

свободных маршрутизаторов или резервных линий (обеспечивающих отказоустойчивость системы), или же приобретение дополнительной пропускной способности

на свободном рынке. Как правило, наиболее загруженные связи и маршрутизаторы

обновляются при первой возможности. Этот процесс называется обеспечением

(provisioning) и использует временную шкалу месяцев, основываясь на долгосрочных

данных о динамике трафика.

Чтобы максимально эффективно использовать пропускную способность сети,

маршруты могут строиться в соответствии со специальными схемами трафика, которые меняются в течение суток по мере того, как пользователи просыпаются и ложатся

спать в различных временных зонах. Можно, например, отвести трафик от загруженных линий, изменив весовые коэффициенты кратчайшего пути. Некоторые радиостанции используют вертолеты, чтобы получить сведения о перегрузках на дорогах и передать их радиослушателям, которые благодаря этому смогут объехать «горячие точки».

Это называется маршрутизацией с учетом состояния трафика (traffic-aware routing).

В таком случае полезно также разделять трафик, направляя его по разным линиям.

Алгоритмы борьбы с перегрузкой

Но иногда увеличение пропускной способности оказывается невозможным. Тогда

единственным средством борьбы с перегрузкой является снижение нагрузки. В сети

виртуальных каналов новые соединения могут быть отклонены, если они приведут

к перегрузке сети. Это называется управлением доступом (admission control).

Возможен и более сложный вариант: когда перегрузка неизбежна, сеть может послать сообщение обратной связи тому отправителю, чей трафик вызывает проблему.

Сеть может попросить отправителя уменьшить трафик или же сделать это сама.

Этот подход (регулирование трафика — traffic throttling) требует ответа на два

вопроса: как обнаружить зарождающуюся перегрузку и как сообщить об этом отправителю, который должен замедлить трафик. Решить первую проблему можно, если

маршрутизаторы будут следить за средней нагрузкой на сеть, временем ожидания

в очереди и числом утерянных пакетов. В любом случае увеличение значений параметров сигнализирует о приближении к перегрузке.

Для решения второй проблемы необходимо, чтобы маршрутизаторы входили

в петлю обратной связи с отправителями. Чтобы данная схема работала, необходимо

тщательно настроить временные параметры. Если каждый раз, когда два пакета приходят одновременно, какой-нибудь нервный маршрутизатор будет кричать «Стоп!»,

а простояв без работы 20 мкс, он же будет давать команду «Давай!», система будет находиться в состоянии постоянных незатухающих колебаний. С другой стороны, если

маршрутизатор для большей надежности станет ждать 30 минут, прежде чем что-либо

сообщить, то механизм борьбы с перегрузкой будет реагировать слишком медленно,

чтобы приносить вообще какую-либо пользу. Своевременная доставка сообщений

обратной связи — не такая уж простая задача. Кроме того, необходимо добиться того,

чтобы маршрутизаторы отправляли больше сообщений, если сеть уже перегружена.

Наконец, если все остальные методы не работают, сеть вынуждена удалить пакеты,

которые она не может доставить. Для этого используется общий термин сброс нагрузки (load shedding). Если правильно выбрать удаляемые пакеты, можно предотвратить

затор.

Сброс нагрузки

Первую стратегию (старое лучше нового) часто называют винной стратегией,

а вторую (новое лучше старого) — молочной стратегией, так как большинство людей

предпочтут пить свежее молоко и выдержанное вино, а не наоборот.

Более разумные алгоритмы сброса нагрузки требуют участия отправителей. В качестве примера можно привести пакеты, содержащие сведения о маршрутизации. Эти

пакеты значительно важнее обычных пакетов с данными, поскольку они устанавливают маршруты; если они будут утеряны, может пострадать связность сети. Другой

пример — алгоритмы сжатия видеосигнала (например, MPEG), которые периодически

посылают полный кадр, а последующие кадры представляют собой карты изменений

относительно последнего полного кадра. В таком случае потеря пакета, содержащего

разностный сигнал, не так страшна, как потеря полного кадра, так как от этого полного

кадра зависят последующие пакеты.

Для реализации интеллектуальной стратегии выбрасывания части информации

приложения должны помечать свои пакеты, сообщая сети об их важности. Тогда

маршрутизаторы смогут сначала выбросить пакеты наименее важного класса, затем

следующего за ним и т. д.

Конечно, при отсутствии стимула все будут помечать свои пакеты не иначе как

ОЧЕНЬ ВАЖНО — НИ В КОЕМ СЛУЧАЕ НЕ ВЫБРАСЫВАТЬ. Предотвращение неоправданного использования таких отметок часто достигается за счет сетевых

ресурсов и денежных средств. Например, сеть может разрешить отправителям пересылать пакеты с большей скоростью, чем указано в договоре на предоставление услуг,

если пакет будет помечаться как низкоприоритетный. Такая стратегия весьма удачна,

поскольку более эффективно использует свободные ресурсы, разрешая хостам пользоваться ими, пока это никому не мешает, но не закрепляя за ними этого права.

## 

## 20. **Определение локальной сети. (недостатки сети, абонент, сервер, клиент). Виды локальных вычислительных сетей**

Локальная вычислительная сеть представляет собой систему распределенной обработки данных, охватывающую небольшую территорию (диаметром до 10 км) внутри учреждений, НИИ, вузов, банков, офисов и т.п., это система взаимосвязанных и распределенных на фиксированной территории средств передачи и обработки информации, ориентированных на коллективное использование общесетевых ресурсов — аппаратных, информационных, программных. ЛВС можно рассматривать как коммуникационную систему, которая поддерживает в пределах одного здания или некоторой ограниченной территории один или несколько высокоскоростных каналов передачи информации, предоставляемых подключенным абонентским системам (АС) для кратковременного использования.

Серверы сети — это аппаратно-программные системы, выполняющие функции управления распределением сетевых ресурсов общего доступа, которые могут работать и как обычная абонентская система. В качестве аппаратной части сервера используются достаточно мощный ПК, мини-ЭВМ, большая ЭВМ или компьютер, спроектированный специально как сервер. В ЛВС может быть несколько различных серверов для управления сетевыми ресурсами, однако всегда имеется один (или более) файл-сервер (сервер баз данных) для управления внешними ЗУ общего доступа и организации распределенных баз данных (РБД). Рабочие станции и серверы соединяются с кабелем коммуникационной подсети с помощью интерфейсных плат — сетевых адаптеров (СА). Основные функции СА: организация приема

(передачи) данных из (в) PC, согласование скорости приема (передачи) информации (буферизация), формирование пакета данных, параллельно-последовательное преобразование (конвертирование), кодирование (декодирование) данных, проверка правильности передачи, установление соединения с требуемым абонентом сети, организация собственно обмена данными. В ряде случаев перечень функций С А существенно увеличивается, и тогда они строятся на основе микропроцессоров и встроенных модемов. В ЛВС в качестве кабельных передающих сред используются витая пара, коаксиальный кабель и оптоволоконный кабель.

ВИДЫ ЛВС

Одноранговая локальная сеть

Одноранговую локальную сеть применяют для объединения небольшого количество компьютеров (до 10 штук). При одноранговой локальной сети, каждый пользователь своего компьютера принимает решение о доступе к данным для других пользователей сети. Такую ЛВС еще называют равноправной.

Локальная сеть на основе сервера

Это более распрастраненный вид ЛВС, более производительный и надежный. Сервером может служить как обычный компьютер, так и специальный, характеристики и програмное обеспечение которого предназначены специально для этих целей. Сервер может выполнять массу функций: хранить в себе данные пользователей ЛВС, назначать права и ограничивать доступ пользователям, при передачи сообщений определять оптимальные маршруты и многое другое.

Топология локальных сетей

Под топологией вычислительной сети понимается способ соединения ее отдельных компонентов (компьютеров, серверов, принтеров и т.д.). Различают три основные топологии:

o топология типа звезда;

o топология типа кольцо;

o топология типа общая шина.

При использовании топологии типа звезда информация между клиентами сети передается через единый центральный узел. В качестве центрального узла может выступать сервер или специальное устройство - концентратор (Hub).

Однако помимо достоинств у данной топологии есть и недостатки:

1. Низкая надежность, так как надежность всей сети определяется надежностью центрального узла. Если центральный компьютер выйдет из строя, то работа всей сети прекратится.

2. Высокие затраты на подключение компьютеров, так как к каждому новому абоненту необходимо ввести отдельную линию.

При топологии типа кольцо все компьютеры подключаются к линии, замкнутой в кольцо. Сигналы передаются по кольцу в одном направлении и проходят через каждый компьютер.

При топологии типа кольцо все компьютеры подключаются к линии, замкнутой в кольцо. Сигналы передаются по кольцу в одном направлении и проходят через каждый компьютер.

К недостаткам данной топологии относятся:

1. Низкая надежность сети, так как отказ любого компьютера влечет за собой отказ всей системы.

2. Для подключения нового клиента необходимо отключить работу сети.

3. При большом количестве клиентов скорость работы в сети замедляется, так как вся информация проходит через каждый компьютер, а их возможности ограничены.

4. Общая производительность сети определяется производи¬тельностью самого медленного компьютера.

При топологии типа общая шина все клиенты подключены к общему каналу передачи данных. При этом они могут непосредственно вступать в контакт с любым компьютером, имеющимся в сети.

К недостаткам топологии типа общая шина относятся:

1. Низкая скорость передачи данных, т.к. вся информация циркулирует по одному каналу (шине).

2. Быстродействие сети зависит от числа подключенных компьютеров. Чем больше компьютеров подключено к сети, тем медленнее идет передача информации от одного компьютера к другому.

3. Для сетей, построенных на основе данной топологии, характерна низкая безопасность, так как информация на каждом компьютере может быть доступна с любого другого компьютера.

Абонент (узел, хост, станция) — это устройство, подключенное к сети и активно участвующее в информационном обмене.

Чаще всего абонентом (узлом) сети является компьютер, но абонентом также может быть, например, сетевой принтер или другое периферийное устройство, имеющее возможность напрямую подключаться к сети.

Клиентом называется абонент сети, который только использует сетевые ресурсы, но сам свои ресурсы в сеть не отдает, то есть сеть его обслуживает, а он ей только пользуется.

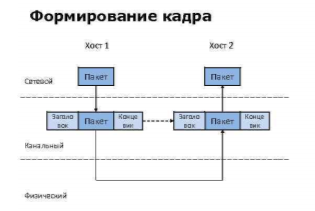
Компьютер-клиент также часто называют рабочей станцией.

l В принципе каждый компьютер может быть одновременно как клиентом, так и сервером.

Под сервером и клиентом часто понимают также не сами компьютеры, а работающие на них программные приложения.

В этом случае то приложение, которое только отдает ресурс в сеть, является сервером, а то приложение, которое только пользуется сетевыми ресурсами — клиентом.

## 21. **Ключевые аспекты организации уровня передачи данных, формирование кадра**



o Задача разбиения потока битов на отдельные кадры не решается простым временным интервалом.

o Методы маркировки начала и конца кадра:

1. Подсчёт количества символов.

2. Использование сигнальных байтов с системным заполнением.

3. Использование стартовых и стоповых бит с битовым

заполнением.

4. Использование запрещённых сигналов физического уровня.

Ключевые аспекты организации уровня передачи данных.

Уровень передачи данных должен выполнять ряд специфических функций. К ним относятся:

1. Обеспечение строго очередного служебного интерфейса для сетевого уровня.

2. Обработка ошибок передачи данных.

3. Управление потоком данных,исключающее затопление медленных приемников быстрыми передачами.

## 22. **Реализация сервиса без установления соединения. Алгоритмы маршрутизации**

Рассмотрев два класса сервисов, которые сетевой уровень может предоставлять своим пользователям, можно перейти к обсуждению устройства этого уровня. Возможны два варианта в зависимости от типа сервиса. Если предоставляется сервис без установления соединения, пакеты внедряются в сеть по отдельности, и их маршруты рассчитываются независимо. При этом никакой предварительной настройки не требуется. В этом случае пакеты часто называют дейтаграммами (datagrams), по аналогии с телеграммами, а сети соответственно дейтаграммными (datagram network). При использовании сервиса, ориентированного на соединение, весь путь от маршрутизатора-отправителя до маршрутизатора-получателя должен быть установлен до начала передачи каких-либо пакетов данных. Такое соединение называется виртуальным каналом (VC, Virtual Circuit), по аналогии с физическими каналами, устанавливаемыми в телефонной системе. Сеть при этом называется сетью виртуального канала (virtual-circuit network). В этом разделе мы обсудим дейтаграммные сети; в следующем разделе — сети виртуального канала. Рассмотрим принцип работы дейтаграммных сетей. Пусть процесс P1

хочет послать длинное сообщение для Р2. Он передает свое послание транспортному

уровню, сообщает ему о том, что доставить данные необходимо процессу Р2, выполняющемуся на хосте Н2. Код транспортного уровня исполняется на хосте Н1; более того, обычно он является частью операционной системы. Заголовок транспортного уровня вставляется в начало сообщения, и в таком виде оно передается на сетевой уровень. Обычно это просто еще одна процедура операционной системы. Предположим, что в нашем примере сообщение в четыре раза длиннее максимального размера пакета, поэтому сетевой уровень должен разбить его на четыре пакета и послать их все поочередно на маршрутизатор А с использованием какого-нибудь протокола двухточечного соединения, например PPP. Здесь вступает в игру интернет-провайдер. Каждый маршрутизатор имеет свою внутреннюю таблицу, по которой он определяет дальнейший путь пакета при каждом из возможных адресов назначения. Каждая запись таблицы состоит из двух полей: пункт назначения (адресат) и выходящая линия для данного адресата. Во втором поле могут использоваться только линии, непосредственно соединенные с данным маршрутизатором. Алгоритм, управляющий таблицами маршрутизации и принимающий решения, называется алгоритмом маршрутизации (routing algorithm). Именно изучению алгоритмов маршрутизации будет уделено основное внимание в этой главе. Как мы увидим, существует несколько типов таких алгоритмов.

IP (Internet Protocol, «межсетевой протокол»), составляющий основу всей сети

Интернет, является наиболее ярким примером сетевого сервиса без установления

соединения. Каждый пакет содержит IP-адрес назначения, с помощью которого

маршрутизатор осуществляет индивидуальную отправку пакета. В пакетах IPv4 используются адреса длиной 32 бита, а в IPv6 — 128 бит.

Алгоритмы маршрутизации

Основная функция сетевого уровня заключается в выборе маршрута для пакетов от

начальной до конечной точки. В большинстве сетей пакетам приходится проходить

через несколько маршрутизаторов. Единственным исключением являются широковещательные сети, но даже в них маршрутизация является важным вопросом, если отправитель и получатель находятся в разных сегментах сети. Алгоритмы выбора

маршрутов и используемые ими структуры данных являются значительной областью

при проектировании сетевого уровня. Алгоритм маршрутизации реализуется той частью программного обеспечения сетевого уровня, которая отвечает за выбор выходной линии для отправки пришедшего пакета. Если сеть использует дейтаграммную службу, выбор маршрута для каждого пакета должен производиться заново, так как оптимальный маршрут мог измениться. Если сеть использует виртуальные каналы, маршрут выбирается только при создании нового виртуального канала. После этого все информационные пакеты следуют по установленному маршруту. Последний случай иногда называют сеансовой маршрутизацией (session routing), так как маршрут остается в силе на протяжении всего сеанса связи (например, все время, пока вы подключены к сети VPN). Полезно понимать разницу между маршрутизацией, при которой системе приходится делать выбор определенного маршрута следования, и пересылкой — действием, происходящим при получении пакета. Можно представить себе маршрутизатор как устройство, в котором функционируют два процесса. Один из них обрабатывает приходящие пакеты и выбирает для них по таблице маршрутизации исходящую линию. Такой процесс называется пересылкой (forwarding). Второй процесс отвечает за заполнение и обновление таблиц маршрутизации. Именно здесь в игру вступает алгоритм маршрутизации. Вне зависимости от того, отдельно ли выбираются маршруты для каждого отправляемого пакета или же только один раз для соединения, желательно, чтобы алгоритм выбора маршрута обладал определенными свойствами — корректностью, простотой, надежностью, устойчивостью, справедливостью и эффективностью. Корректность и простота вряд ли требуют комментариев, а вот потребность в надежности не столь очевидна с первого взгляда. Во время работы большой сети постоянно происходят какие-то отказы аппаратуры и изменения топологии. Алгоритм маршрутизации должен уметь справляться с изменениями топологии и трафика без необходимости прекращения всех задач на всех хостах. Представьте себе, что было бы, если бы сеть перезагружалась при каждой поломке маршрутизатора! Алгоритм маршрутизации должен также обладать устойчивостью. Существуют алгоритмы выбора маршрута, никогда не сходящиеся к фиксированному набору путей, независимо от того, как долго они работают. Устойчивый алгоритм должен достигать состояния равновесия и оставаться в нем. Но он также должен быстро находить этот набор путей, так как соединение может быть прервано до того, как будет достигнуто равновесие. Такие цели, как справедливость и эффективность, могут показаться очевидными — вряд ли кто-нибудь станет возражать против них, — однако они зачастую оказываются взаимоисключающими. Для увеличения эффективности передачи данных по сети можно попробовать минимизировать среднее время задержки или увеличить общую пропускную способность сети. Однако эти цели также противоречат друг другу, поскольку работа любой системы с очередями вблизи максимума производительности предполагает долгое стояние в очередях. В качестве компромисса многие сети стараются минимизировать расстояние, которое должен пройти пакет, или просто снизить количество пересылок для каждого пакета. В обоих случаях время прохождения каждого пакета по сети снижается, в результате чего улучшается пропускная способность всей сети. Алгоритмы выбора маршрута можно разбить на два основных класса: неадаптивные и адаптивные. **Неадаптивные алгоритмы** не учитывают при выборе маршрута топологию и текущее состояние сети и не измеряют трафик на линиях. Вместо этого выбор маршрута для каждой пары станций производится заранее, в автономном режиме, и список маршрутов загружается в маршрутизаторы во время загрузки сети. Такая процедура иногда называется статической маршрутизацией (static routing). Поскольку статическая маршрутизация не реагирует на сбои, она, как правило, используется в тех случаях, когда выбор маршрута очевиден. **Адаптивные алгоритмы**, напротив, изменяют решение о выборе маршрутов при изменении топологии и также иногда в зависимости от загруженности линий. Эти динамические алгоритмы маршрутизации (dynamic routing algorithms) отличаются источниками получения информации (такие источники могут быть, например, локальными, если это соседние маршрутизаторы, либо глобальными, если это вообще все маршрутизаторы сети), моментами изменения маршрутов (например, при изменении топологии или через определенные равные интервалы времени при изменении нагрузки) и данными, использующимися для оптимизации (расстояние, количество транзитных участков или ожидаемое время пересылки). В следующих разделах мы обсудим различные алгоритмы маршрутизации. Помимо отправки пакета от источника к месту назначения такие алгоритмы предусматривают модель предоставления информации. Иногда требуется отправить пакет на несколько адресов из заданного списка, на все такие адреса или на один из них.

## 23. **Службы и протоколы на основе соединений и службы без установления соединений**.

Уровни могут предлагать вышестоящим уровням услуги двух типов: с наличием или отсутствием установления соединения. В этом разделе мы рассмотрим, что означает каждый из этих типов и в чем состоит разница между ними.

Типичным примером сервиса с установлением соединения является телефонная связь. Чтобы поговорить с кем-нибудь, необходимо поднять трубку, набрать номер, а после окончания разговора положить трубку. Нечто подобное происходит и в компьютерных сетях: при использовании сервиса с установлением соединения абонент сперва устанавливает соединение, а после окончания сеанса разрывает его. Это напоминает трубу: биты сообщения влетают в один ее конец, а вылетают с другого. В большинстве случаев не возникает путаницы с последовательностью передачи этих битов.

В некоторых случаях перед началом передачи отправляющая и получающая машины обмениваются приветствиями, отсылая друг другу приемлемые параметры соединения: максимальный размер сообщения, необходимое качество сервиса и др. В большинстве случаев одна из сторон посылает запрос, а другая его принимает, отвергает или же выставляет встречные условия.

Что касается сервисов без установления соединения, то типичный пример такой технологии – почтовые системы. Каждое письмо содержит полный адрес назначения и проходит по некоему маршруту, который совершенно не зависит от других писем. Обычно то письмо, которое отправлено раньше, в место назначения приходит раньше.

Каждая служба характеризуется качеством обслуживания. Некоторые службы являются надежными в том смысле, что они никогда не теряют данных. Обычно надежная служба реализуется при помощи подтверждений, посылаемых получателем в ответ на каждое принятое сообщение, так что отправитель знает, дошло очередное сообщение или нет. Процесс пересылки подтверждений требует некоторых накладных расходов и снижает пропускную способность канала. Впрочем, подобные затраты обычно не очень велики и окупаются, хотя иногда могут быть нежелательными.

Типичным примером необходимости надежной службы на основе соединений является пересылка файлов. Владелец файла хочет быть уверенным, что все биты файла прибыли без искажений и в том же порядке, в котором были отправлены. Вряд ли кто-нибудь отдаст предпочтение службе, которая случайным образом искажает информацию, даже если передача происходит значительно быстрее.

Надежные службы на основе соединений бывают двух типов: последовательности сообщений и байтовые потоки. В первом варианте сохраняются границы между сообщениями. Когда посылаются два сообщения размером по 1 Кбайт, то они прибывают в виде двух сообщений размером по 1 Кбайт и никогда – как одно двух-килобайтное сообщение. При втором варианте связь представляет собой просто поток байтов, без разделения на отдельные сообщения. Когда 2048 байт прибывают к получателю, то нет никакой возможности определить, было это одно сообщение длиной 2 Кбайт, два сообщения длиной 1 Кбайт или же 2048 однобайтных сообщений. Если страницы книги посылаются по сети фотонаборной машине в виде отдельных сообщений, то, возможно, необходимо сохранить границы между сообщениями. С другой стороны, при регистрации с удаленного терминала в системе разделения времени вполне достаточно потока байтов с терминального компьютера.

Как уже упоминалось ранее, существуют системы, для которых задержки, связанные с пересылкой подтверждений, неприемлемы. В качестве примера такой системы можно назвать цифровую голосовую связь. В данном случае предпочтительнее допустить шумы на линии или искаженные слова, нежели большие паузы, вызванные отсылкой подтверждений и повторной передачей блоков данных. Аналогично, при проведении видеоконференции отдельные неправильные пикселы окажутся меньшей проблемой, нежели дергающиеся и останавливающиеся кадры.

Не все приложения требуют установки соединения. Например, при рассылке рекламы по электронной почте установка связи для пересылки каждого отдельного сообщения нежелательна. Также не требуется в этом случае и 100 процентная надежность, особенно, если это существенно увеличит стоимость. Все, что нужно, – это способ переслать сообщение с высокой вероятностью его получения, но без гарантии. Ненадежная (то есть без подтверждений) служба без установления соединения часто называется службой дейтаграмм, или дейтаграммной службой – по аналогии с телеграфной службой, также не предоставляющей подтверждений отправителю.

В других ситуациях бывает желательно не устанавливать соединение для пересылки коротких сообщений, но надежность, тем не менее, существенна. Такая служба называется службой дейтаграмм с подтверждениями. Она подобна отправке заказного письма с подтверждением получения. Получив подтверждение, отправитель уверен, что письмо доставлено адресату, а не потеряно по дороге.

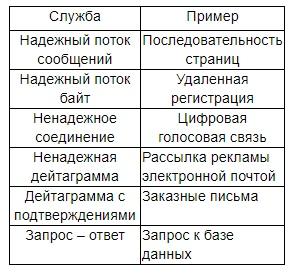
Ориентированная на соединение

Без установления соединения

Кроме того, существует служба запросов и ответов, в которой отправитель посылает дейтаграммы, содержащие запросы, и получает ответы от получателя. Например, к данной категории можно отнести вопрос к библиотеке о том, где говорят по-уйгурски. Обычно модель запросов и ответов применяется для реализации общения в модели «клиент-сервер»: клиент посылает запрос, а сервер отвечает на него. Обсуждавшийся ранее типы служб сведены в таблицу на рис. 1.13.

Служба

Концепция использования ненадежной связи поначалу может показаться несколько странной. В самом деле, почему это может возникать такая ситуация, когда выгоднее предпочесть ненадежную связь надежной? Во-первых, надежное соединение (в том смысле, который был оговорен ранее, то есть с подтверждением) не всегда можно установить. Скажем, Ethernet не является «надежным» средством коммуникации. Пакеты при передаче могут искажаться, но решать эту проблему должны протоколы более высоких уровней. Во-вторых, задержки, связанные с отсылкой подтверждения, в некоторых случаях неприемлемы, особенно при передаче мультимедиа в реальном времени. Именно благодаря этим факторам продолжают сосуществовать надежные и ненадежные соединения.



## 

## **24. Метод коммутации пакетов с ожиданием**

Прежде чем начать подробное рассмотрение сетевого уровня, необходимо восстановить в памяти окружение, в котором ему приходится функционировать. Оно показано на рис. 5.1. Основными компонентами сети являются устройства интернет-провай-дера (маршрутизаторы, соединенные линиями связи), показанные внутри затененного овала, а также устройства, принадлежащие клиенту и показанные вне овала. Хост H1 напрямую соединен с одним из маршрутизаторовинтернет-провайдераА (как, например, домашний компьютер, подключенный к DSL-модему).ХостH2, на- против, находится в ЛВС (например, офисной сети Ethernet) с маршрутизаторомF, принадлежащим клиенту, который с ним работает. Этот маршрутизатор связывается синтернет-провайдеромпо выделенной линии. Мы показалиF вне овала, потому что он не принадлежитинтернет-провайдеру.Однако в контексте данной главы мы будем считать маршрутизаторы клиента частью сетиинтернет-провайдера,поскольку в них применяются те же самые алгоритмы, что и в маршрутизаторахинтернет-провайдеров(а основным предметом рассмотрения будут именно алгоритмы). Система работает следующим образом. Хост, у которого есть пакет для передачи, посылает его на ближайший маршрутизатор либо в своей ЛВС, либо по двухточечному соединению интернет-провайдеру. Там пакет хранится до тех пор, пока не будет принят целиком и не пройдет полную обработку, включая верификацию контрольной суммы. Затем он передается по цепочке маршрутизаторов, которая в итоге приводит к пункту назначения. Такой механизм называется коммутацией пакетов с ожиданием(store-and-forward),и мы уже рассматривали его в предыдущих главах.

Сервисы, предоставляемые транспортному уровню

Сетевой уровень предоставляет транспортному уровню сервисы через интерфейс между сетевым и транспортным уровнями. Важным вопросом является именно то, какой именно вид сервиса сетевой уровень предоставляет транспортному .Разработка таких сервисов требует особой аккуратности и при этом необходимо учитывать следующее: ÊСервисы сетевого уровня не должны зависеть от технологии маршрутизатора. ÊТранспортный уровень должен быть независим от количества, типа и топологии присутствующих подсетей с маршрутизаторами. ÊСетевые адреса, доступные транспортному уровню, должны использовать единую систему нумерации, даже между локальными и глобальными сетями.

## 25. **Определение локальных сетей и их топология. Стандарт IEEE 802.x**

Локальные сети - частные сети, размещающиеся, как правило, в одном здании или на территории какой-либо организации. Используют для объединения PC (компьютеров), рабочих станций в офисах компании или предприятия бытовой электроники для предоставления совместного доступа к ресурсам (например - принтерам) и обмена информацией.

Локальные сети делятся на 2 вида:

Одноранговые локальные сети – сети, где все компьютеры равноправны: каждый из компьютеров может быть и сервером, и клиентом. Пользователь каждого из компьютеров сам решает, какие ресурсы будут предоставлены в общее пользование и кому.

Локальные сети с цетрализованным управлением. В сетях с централизованным управлением политика безопасности общая для всех пользователей сети.

Топология компьютерной сети - физическое расположение компьютеров сети друг относительно друга и способ соединения их линиями связи.

Топология локальных сетей:

Общая шина - все клиенты подключены к общему каналу передачи данных.

Преимущества:

-Вся информация находится в сети и доступна каждому компьютеру.

-Рабочие станции можно подключать независимо друг от друга. Т.е. при подключении нового абонента нет необходимости останавливать передачу информации в сети.

-Дешевизна, так как отсутствуют затраты на прокладку дополнительных линий при подключении нового клиента.

-Сеть обладает высокой надежностью, т.к. работоспособность сети не зависит от работоспособности отдельных компьютеров.

Недостатки:

-Низкая скорость передачи данных, т.к. вся информация циркулирует по одному каналу (шине).

-Быстродействие сети зависит от числа подключенных компьютеров. Чем больше компьютеров подключено к сети, тем медленнее идет передача информации от одного компьютера к другому.

-Низкая безопасность, так как информация на каждом компьютере может быть доступна с любого другого компьютера.

Звезда: активная и пассивная - информация между клиентами сети передается через единый центральный узел.

Преимущества:

-Высокое быстродействие сети, так как общая производительность сети зависит только от производительности центрального узла.

-Отсутствие столкновения передаваемых данных, так как данные между рабочей станцией и сервером передаются по отдельному каналу, не затрагивая другие компьютеры.

Недостатки:

-Низкая надежность, так как надежность всей сети определяется надежностью центрального узла. -Если центральный компьютер выйдет из строя, то работа всей сети прекратится.

-Высокие затраты на подключение компьютеров, так как к каждому новому абоненту необходимо ввести отдельную линию.

Кольцо - все компьютеры подключаются к линии, замкнутой в кольцо. Сигналы передаются по кольцу в одном направлении и проходят через каждый компьютер.

Преимущества:

-Пересылка сообщений является очень эффективной, т.к. можно отправлять несколько сообщений друг за другом по кольцу. Т.е. компьютер, отправив первое сообщение, может отправлять за ним следующее сообщение, не дожидаясь, когда первое достигнет адресата.

-Протяженность сети может быть значительной. Т.е. компьютеры могут подключаться к друг к другу на значительных расстояниях, без использования специальных усилителей сигнала.

Недостатки:

-Низкая надежность сети, так как отказ любого компьютера влечет за собой отказ всей системы.

-Для подключения нового клиента необходимо отключить работу сети.

-При большом количестве клиентов скорость работы в сети замедляется, так как вся информация проходит через каждый компьютер, а их возможности ограничены.

-Общая производительность сети определяется производительностью самого медленного компьютера.

IEEE 802 — группа стандартов семейства IEEE, касающихся локальных вычислительных сетей (LAN) и сетей мегаполисов (MAN). Был принят в феврале 1980 года.

Службы и протоколы, указанные в IEEE 802, находятся на двух нижних уровнях (канальный и физический) семиуровневой сетевой модели OSI. Фактически IEEE 802 разделяет канальный уровень OSI на два подуровня — Media Access Control (MAC) и Logical Link Control (LLC). Таким образом, уровни располагаются в следующем виде:

-Канальный уровень

---Подуровень LLC - отвечает за передачу с различной степенью надежности кадров данных между узлами, а также реализует функции интерфейса с прилегающим к нему сетевым уровнем. Именно уровень LLC принимает запрос от сетевого про­токола на выполнение транспортной операции канального уровня с тем или иным качеством.

---Подуровень MAC - обеспечивает корректное совместное использование общей среды, предоставляя ее в соответствии с определенным ал­горитмом в распоряжение того или иного узла сети.

-Физический уровень

Рабочие группы:

Стандарт IEEE 802.1 – стандарт управления сетевыми устройствами и их взаимодействием.

Стандарт IEEE 802.3 – стандарт Ethernet – наиболее распространенный тип проводной локальной сети. Топология – пассивная звезда (т.е. с коммутатором).

Стандарт IEEE 802.11 (Wi-Fi)– стандарт беспроводных сетей.

Стандарт IEEE 802.15 – стандарт персональной сети.

А также стандарты от 802.15 до 802.24