НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»
Кафедра информатики и процессов управления (№17)

Курс «Современные операционные системы»

Лекция 3

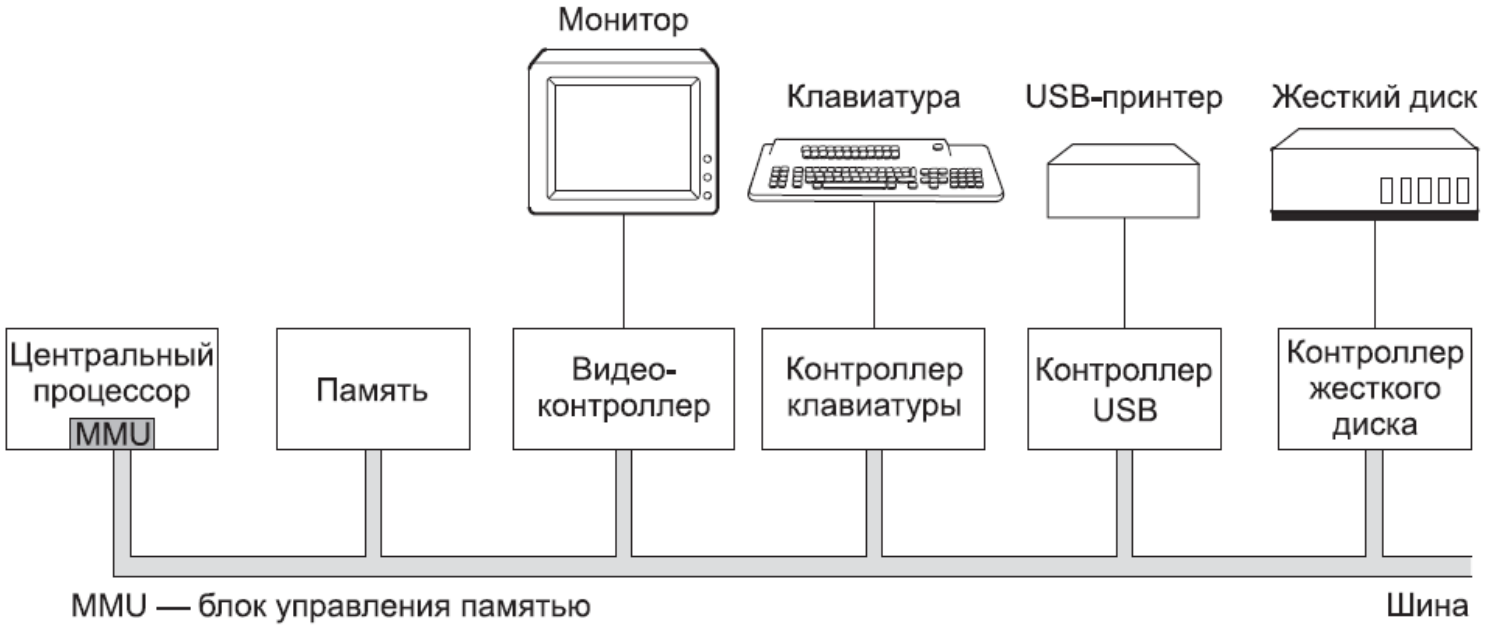
Аппаратное обеспечение компьютера

Москва
2016

Содержание

1. Аппаратное обеспечение компьютера

1.1. Модель простого ПК



1.2. Процессор

Регистры:

- **Общего назначения** применяются для хранения переменных и промежуточных результатов.
- **Счетчик команд** содержит адрес ячейки памяти со следующей выбираемой командой. После выборки этой команды счетчик команд обновляется, переставляя указатель на следующую команду.
- **Указатель стека** ссылается на вершину текущего стека в памяти. Стек содержит по одному фрейму (области данных) для каждой процедуры, в которую уже вошла, но из которой еще не вышла программа. В стековом фрейме процедуры хранятся ее входные параметры, а также локальные и временные переменные, не содержащиеся в регистрах.
- **PSW** (Program Status Word – слово состояния программы) содержит биты кода условия, устанавливаемые инструкциями сравнения, а также биты управления приоритетом центрального процессора, режимом (*пользовательским* или *ядра*) и другие служебные биты. Регистр PSW играет важную роль в системных вызовах и операциях ввода-вывода.

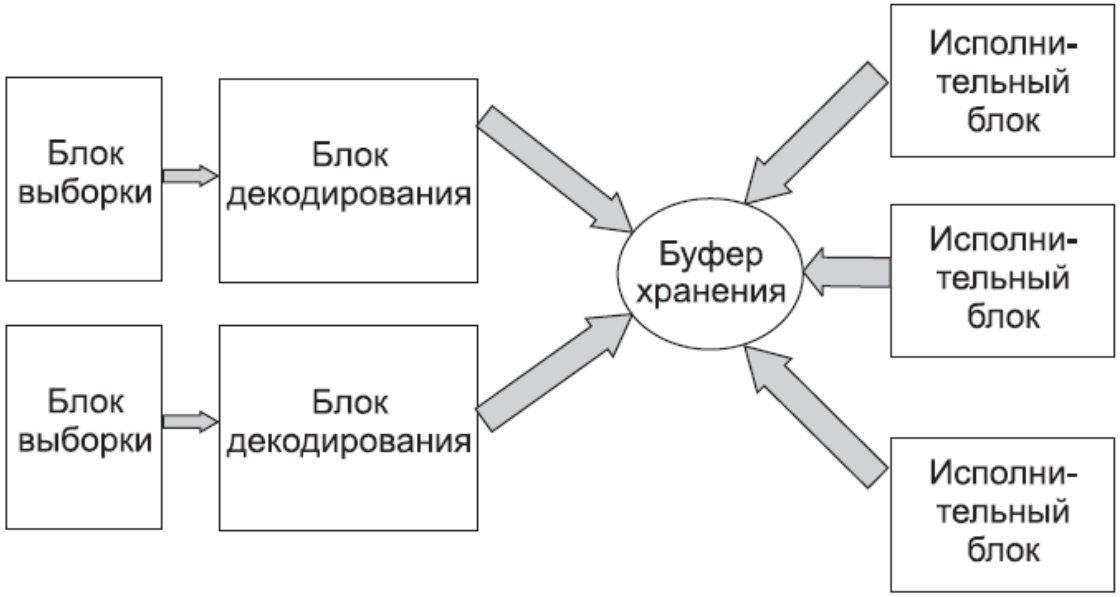
Для повышения производительности современные процессоры способны одновременно выполнять более одной команды. Например, у процессора могут быть отдельные блоки для выборки, декодирования и выполнения команд, тогда во время выполнения команды n он сможет декодировать команду n+1 и осуществлять выборку команды n+2. Подобная организация работы называется ***конвейером***.

Конвейер с тремя стадиями обработки.

При разработке компиляторов и операционных систем конвейеры – основная сложность.

Более совершенной конструкцией обладает ***суперскалярный*** процессор, он имеет несколько исполнительных блоков, например: один — для целочисленной арифметики, другой — для арифметики чисел с плавающей точкой, третий — для логических операций.

Одновременно выбираются две и более команды, которые декодируются и помещаются в буфер хранения, в котором ожидают возможности своего выполнения. Как только исполнительный блок становится доступен, он обращается к буферу хранения за командой, которую может выполнить, и если такая команда имеется, извлекает ее из буфера, а затем выполняет. В результате команды программы часто выполняются не в порядке их следования. При этом обеспечение совпадения конечного результата с тем, который получился бы при последовательном выполнении команд, возлагается в основном на аппаратуру.



Суперскалярный процессор.

Для получения услуг от операционной системы пользовательская программа должна осуществить ***системный вызов***, который перехватывается внутри ядра и вызывает операционную систему. Инструкция перехвата ***TRAP*** («ловушка») осуществляет переключение из пользовательского режима в режим ядра и запускает операционную систему. Когда обработка вызова будет завершена, управление возвращается пользовательской программе и выполняется команда, которая следует за системным вызовом.

1.3. Многопоточность и многоядерность

Дублирование на одном кристалле процессора не только функциональных блоков, но и части управляющей логики называется ***многопоточностью***, или ***гиперпоточностью*** (hyperthreading по версии Intel). Она впервые была использована в Pentium 4 и стала неотъемлемой принадлежностью процессора x86 и ряда других процессоров, включая SPARC, Power5, Intel Xeon, а также процессоры семейства Intel Core.

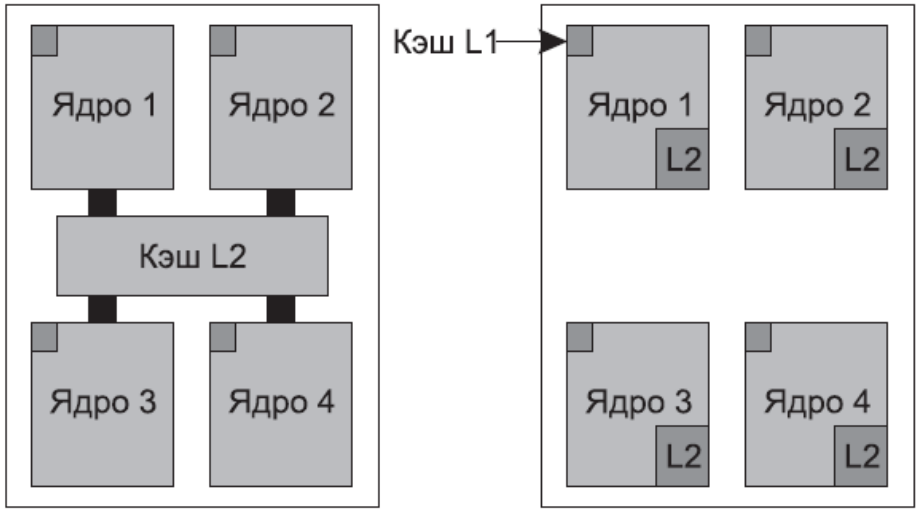
Эта технология позволяет процессору сохранять состояние двух различных потоков и переключаться между ними за наносекунды (***поток*** является разновидностью легковесного процесса – выполняющейся программы) Например, если одному из процессов нужно прочитать слово из памяти (что занимает несколько тактов), многопоточный процессор может переключиться на другой поток.

Многопоточность не предлагает настоящей параллельной обработки данных. Одновременно работает только один процесс, но время переключения между процессами сведено до наносекунд.

Многопоточность оказывает влияние на операционную систему, поскольку каждый поток выступает перед ней как отдельный центральный процессор.

Процессоры, имеющие на одном кристалле два, четыре, восемь и более полноценных процессоров, или ***ядер*** называются ***многоядерными***, для них потребуется многопроцессорная ОС.

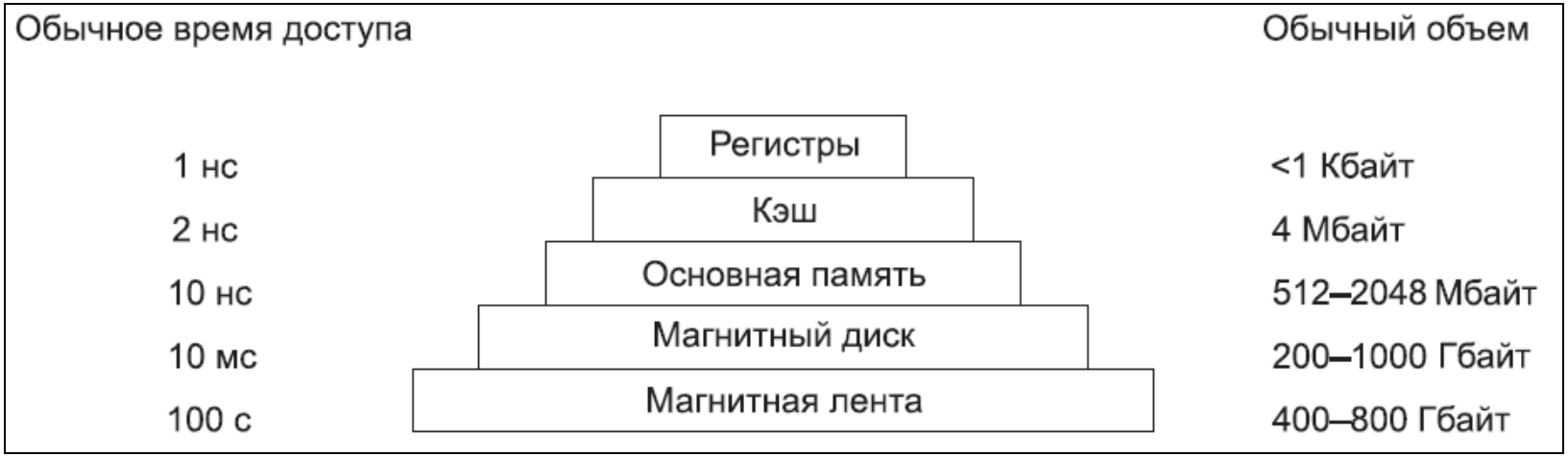
Современные графические процессоры (Graphics Processing Unit, GPU) содержат на своих кристаллах тысячи крохотных ядер. Они очень хорошо подходят для множества небольших производимых параллельно вычислений, но для последовательных задач не годятся. К тому же их трудно программировать.



Четырехядерные процессоры с кэш-памятью второго уровня L2:
Intel (с общей L2) AMD (с раздельной L2)

1.4. Память

Память должна работать быстрее, чем производится выполнение одной инструкции, чтобы работа центрального процессора не замедлялась обращениями к памяти. Но этого не в состоянии обеспечить никакая современная технология. Поэтому система памяти создается в виде иерархии уровней.



Иерархия памяти.

Внутренние **регистры** обычно предоставляют возможность для хранения 32×32 бит для 32-разрядного процессора или 64×64 бит — для 64-разрядного. В обоих случаях этот объем не превышает 1 Кбайт.

Затем следует **кэш-память**, которая управляется аппаратурой, она разделяется на кэш-строки, обычно по 64 байт, с адресами от 0 до 63 в *кэш-строке 0*, адресами от 64 до 127 в *кэш-строке 1* и т. д. Наиболее интенсивно используемые кэш-строки оперативной памяти сохраняются в высокоскоростной кэш-памяти. Обычно результативное обращение к кэшу занимает по времени два такта. Отсутствие слова в кэш-памяти вынуждает обращаться к оперативной памяти, что приводит к существенной потере времени.

Кэш-память из-за своей высокой стоимости ограничена в объеме. Некоторые машины имеют два или даже три уровня кэша, причем каждый из последующих медленнее и объемнее предыдущего.

Первый уровень, или **кэш L1**, всегда является частью самого процессора и обычно подает декодированные команды в процессорный механизм исполнения команд. У многих процессоров есть и второй кэш L1 для тех слов данных, которые используются особенно интенсивно. Обычно каждый из кэшей L1 имеет объем 16 Кбайт.

Вдобавок к этому кэшу процессоры часто оснащаются вторым уровнем кэш-памяти, **кэш L2** и содержит несколько Мбайт недавно использованных слов памяти. Доступ к кэшу первого уровня осуществляется без задержек, а доступ к кэшу второго уровня требует задержки в один или два такта.

Роль основной памяти играет **оперативная память**, которую называют оперативным запоминающим устройством (**ОЗУ**), или памятью с произвольным доступом (Random Access Memory (**RAM**)).

Дополнительно к оперативной памяти компьютеры оснащены небольшой по объему неизменяемой памятью с произвольным доступом — постоянным запоминающим устройством (ПЗУ), оно же память, предназначенная только для чтения (Read Only Memory (ROM)).

В отличие от ОЗУ она не утрачивает своего содержимого при отключении питания, то есть является энергонезависимой. ПЗУ программируется на предприятии-изготовителе и впоследствии не подлежит изменению, она характеризуется высоким быстродействием и дешевизной.

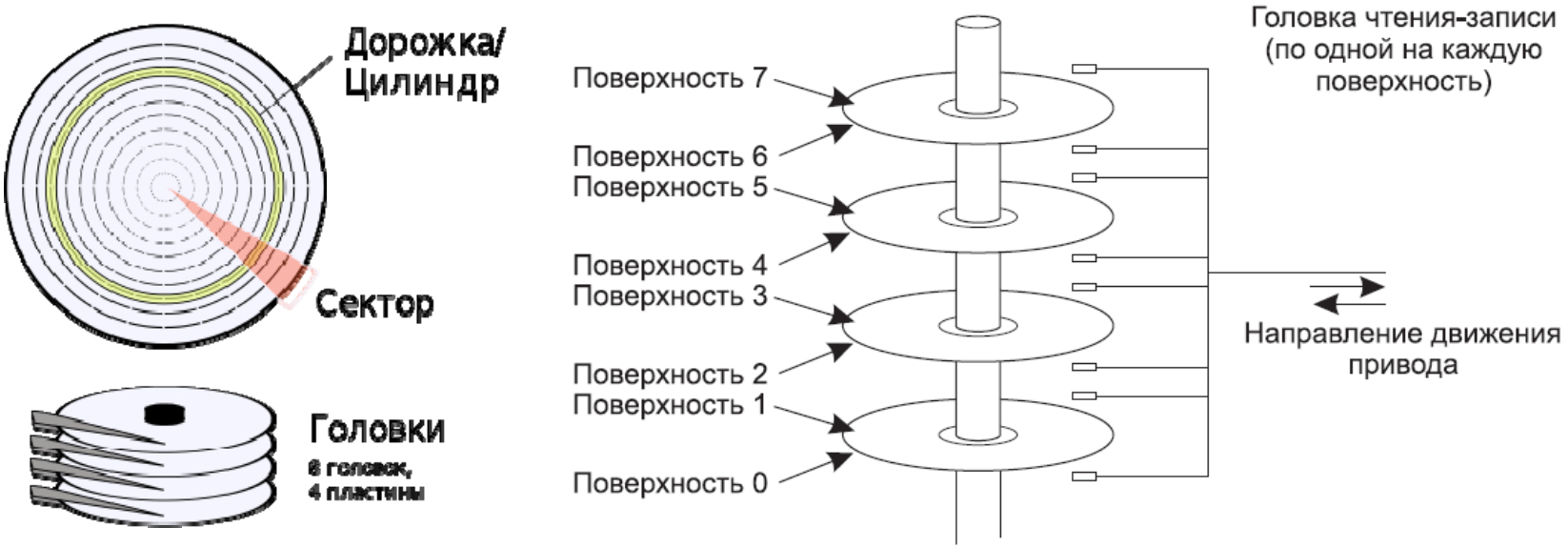
На некоторых компьютерах в ПЗУ размещается начальный загрузчик (**BIOS**), используемый для их запуска. Такой же памятью, предназначенной для осуществления низкоуровневого управления устройством, оснащаются некоторые контроллеры ввода-вывода.

Другие разновидности энергонезависимой памяти, которые в отличие от ПЗУ могут стираться и перезаписываться, — электрически стираемые программируемые постоянные запоминающие устройства (ЭСППЗУ), они же **EEPROM** (Electrically Erasable PROM), и **флеш-память**. Однако запись в них занимает на несколько порядков больше времени, чем запись в ОЗУ, поэтому они используются для тех же целей, что и ПЗУ.

1.5. Жесткий диск

Жесткий диск состоит из одной или нескольких металлических пластин, вращающихся со скоростью 5400, 7200, 10 800 и более оборотов в минуту.

Дорожка – кольцеобразный участок концентрической окружности на поверхности диска в заданной позиции привода. Совокупность всех дорожек одинакового радиуса в заданной позиции привода – ***цилиндр***. Каждая дорожка поделена на определенное количество ***секторов***, обычно по 512 байт.



Перемещение привода с одного цилиндра на другой занимает около 1 мс. Перемещение к произвольно выбранному цилиндру обычно занимает от 5 до 10 мс. Ожидание, когда нужный сектор попадет под головку, – от 5 до 10 мс. Операция чтения или записи со скоростью от 50 Мбайт/с до 160 Мбайт/с.

Два основных способа адресации секторов на диске:

- цилиндр-головка-сектор (англ. cylinder-head-sector, **CHS**),
- линейная адресация блоков (англ. linear block addressing, **LBA**).

LBA – ограничение размера диска обусловлено лишь разрядностью LBA. В настоящее время для задания номера блока используется 48 бит, что дает возможность адресовать 2^{48} блоков (128 Пбайт).

LBA (c,h,s) = (c·H + h)·S + s – 1

где **c** — номер цилиндра,
h — номер головки,
s — номер сектора,
H — число головок,
S — число секторов на дорожке.

Твердотельные накопители — **SSD** (Solid State Disks) не имеют движущихся частей, данные хранятся во флеш-памяти.

Схема ***виртуальной памяти*** дает возможность запускать программы, превышающие по объему физическую память компьютера, за счет помещения их на диск и использования оперативной памяти как некой разновидности кэша для наиболее интенсивно исполняемых частей. При этом адрес, сгенерированный программой, конвертируется в физический адрес в ОЗУ. Такое отображение адресов осуществляется частью процессора – блоком управления памятью **MMU** (Memory Management Unit), или **диспетчером памяти**.

1.6. Устройства ввода-вывода

Устройства ввода-вывода обычно состоят из двух компонентов: самого устройства и его ***контроллера***, который представляет собой микросхему или набор микросхем, управляющих устройством на физическом уровне.

Контроллер принимает от операционной системы команды и выполняет их. Задачей контроллера является предоставление ОС простого (но не упрощенного) интерфейса. Поскольку интерфейс устройства скрыт его контроллером, ОС видит только интерфейс контроллера, который может существенно отличаться от интерфейса самого устройства.

Программа, предназначенная для общения с контроллером, выдачи ему команды и получения поступающих от него ответов, называется ***драйвером*** устройства. Каждый производитель контроллеров должен поставлять вместе с ними драйверы для каждой поддерживаемой ОС.

Для использования драйвер нужно поместить в ОС, предоставив ему тем самым возможность работать в режиме ядра. Вообще драйверы могут работать и не в режиме ядра, но подавляющее большинство драйверов по-прежнему запускается ниже границы ядра.

Существует три способа установки драйвера в ядро.

1) Заново скомпоновать ядро вместе с новым драйвером и затем перезагрузить систему (устаревшие UNIX-системы).
2) Создать в специальном файле ОС запись, сообщающую ей о том, что требуется, и затем перезагрузить систему. Во время загрузки ОС сама находит нужные ей драйверы и загружает их (Windows).
3) Динамическая загрузка драйверов — ОС может принимать новые драйверы в процессе работы и оперативно устанавливать их, не требуя для этого перезагрузки.

Внешние устройства, работающие по принципу «горячего подключения», например, с интерфейсами USB и IEEE 1394, всегда нуждаются в динамически загружаемых драйверах.

В каждом контроллере для связи с ним имеется небольшое количество регистров. Чтобы активизировать контроллер, драйвер получает команду от ОС, затем переводит ее в соответствующие значения для записи в регистры устройства.

Из совокупности всех регистров устройств формируется ***пространство портов ввода-вывода*** (I/O port space), в котором каждый регистр имеет адрес порта. Для такой схемы в режиме ядра доступны специальные команды ввода-вывода, позволяющие драйверам читать и записывать данные в регистры.

Ввод и вывод данных можно делать тремя различными способами.

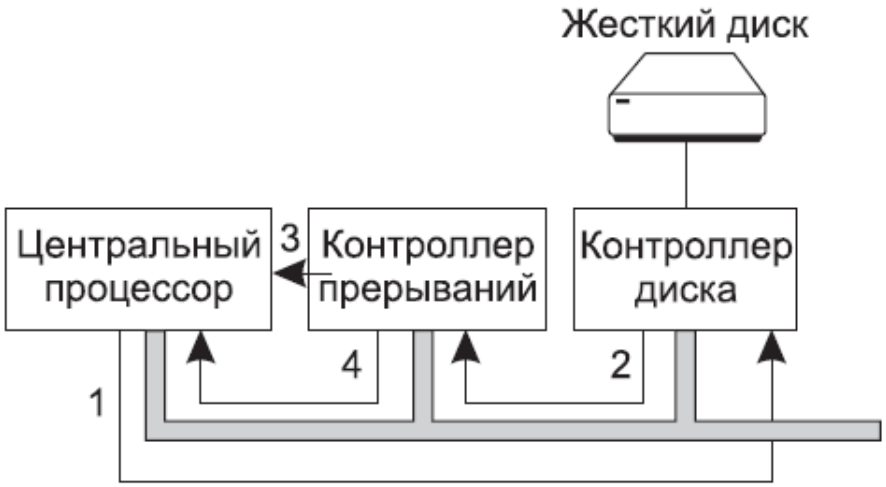
1) Активное ожидание (ожидание готовности).

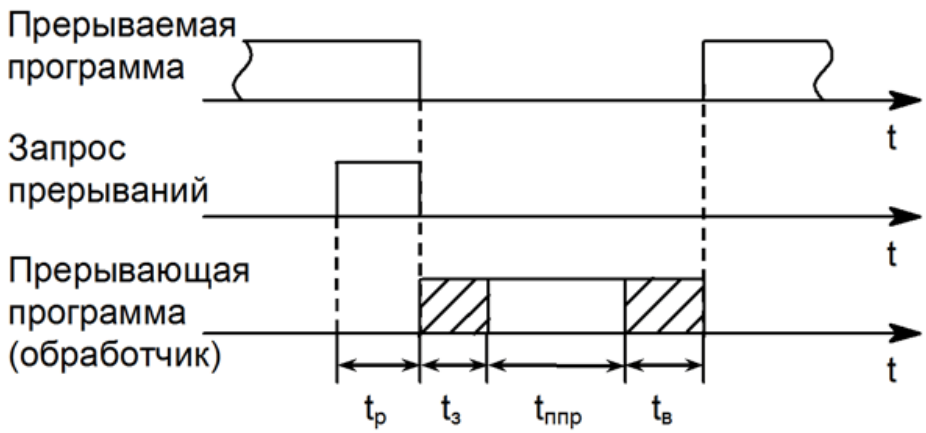
Пользовательская программа производит ***системный вызов***, который передается ядром в процедуру вызова соответствующего драйвера. После этого драйвер приступает к процессу ввода-вывода. В это время он выполняет очень короткий цикл, постоянно опрашивая устройство и отслеживая завершение операции (обычно занятость устройства определяется состоянием специального бита). По завершении операции ввода-вывода драйвер помещает данные (если таковые имеются) туда, куда требуется, и возвращает управление. Затем операционная система возвращает управление вызывающей программе.

Недостаток: процессор загружен опросом устройства об окончании работы.

2) Прерывания.

Драйвер запускает устройство и просит его выдать ***прерывание*** по окончании выполнения команды (завершении ввода или вывода данных). Сразу после этого драйвер возвращает управление. Затем ОС блокирует вызывающую программу, если это необходимо, и переходит к выполнению других задач. Когда контроллер обнаруживает окончание передачи данных, он генерирует прерывание, чтобы просигнализировать о завершении операции.





Запуск устройства и получение прерывания:

1 – драйвер передает команду контроллеру, записывая информацию в его регистры, затем контроллер запускает само устройство;

2 – контроллер устройства завершает чтение или запись заданного ему количества байтов и выставляет сигнал для контроллера прерываний;

3 – если контроллер прерываний готов принять прерывание (а он может быть и не готов к этому, если обрабатывает прерывание с более высоким приоритетом), то он выставляет сигнал на контакте процессора, информируя его о завершении операции;

4 – контроллер прерываний выставляет номер устройства на шину, чтобы процессор мог его считать и узнать, какое устройство только что завершило работу (поскольку одновременно могут работать сразу несколько устройств).

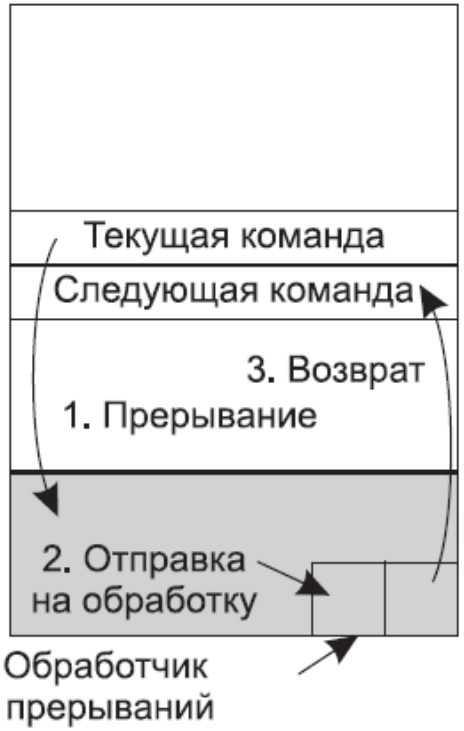
$t_р$ – реакция системы на прерывание;
$t_з$ – запоминание состояния программы;
$t_{ппр}$ – обработка прерывания;
$t_в$ – восстановление состояния программы.

Обработка прерывания:

1. Как только центральный процессор решит принять прерывание, содержимое счетчика команд и слова состояния программы помещаются, как правило, в текущий стек и процессор переключается в режим ядра. Номер устройства может быть использован как индекс части памяти, используемой для поиска адреса ***обработчика прерываний*** (interrupt handler) данного устройства. Эта часть памяти называется ***вектором прерываний***.

2. Когда обработчик прерываний (являющийся частью драйвера устройства, выдающего запрос на прерывание) начинает свою работу, он извлекает помещенные в стек содержимое счетчика команд и слова состояния программы и сохраняет их, а затем опрашивает устройство для определения его состояния.

3. После завершения обработки прерывания обработчик возвращает управление ранее работавшей пользовательской программе — на первую же еще не выполненную команду.



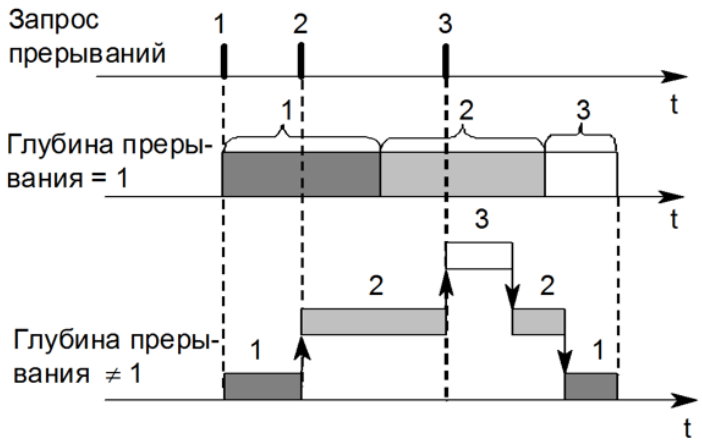
3) Прямой доступа к памяти (DMA).

Используется специальный контроллер ***прямого доступа к памяти*** (Direct Memory Access (**DMA**)), который может управлять потоком битов между оперативной памятью и контроллерами без постоянного вмешательства центрального процессора. Центральный процессор осуществляет настройку контроллера DMA, сообщая ему, сколько байтов следует передать, какое устройство и адреса памяти задействовать и в каком направлении передать данные, а затем дает ему возможность действовать самостоятельно. Когда контроллер DMA завершает работу, он выдает запрос на прерывание.

Приоритеты прерывний.

Процессор обладает возможностью запрещать прерывания с последующим их разрешением. Если за время запрещения прерываний завершится работа сразу нескольких устройств, контроллер решает, какое из них должно быть обработано первым, полагаясь обычно на ***приоритеты***, назначенные каждому устройству. Все остальные устройства ожидают своей очереди.

Если во время обработки прерывания поступает запрос на прерывание с более высоким приоритетом, то управление передается обработчику прерывания более высокого приоритета, при этом работа обработчика прерывания с более низким уровнем приоритета приостанавливается. Возникает **вложенность** прерываний. Максимальное число программ, которые могут приостанавливать друг друга называется **глубиной** прерываний.



1.7. Шина

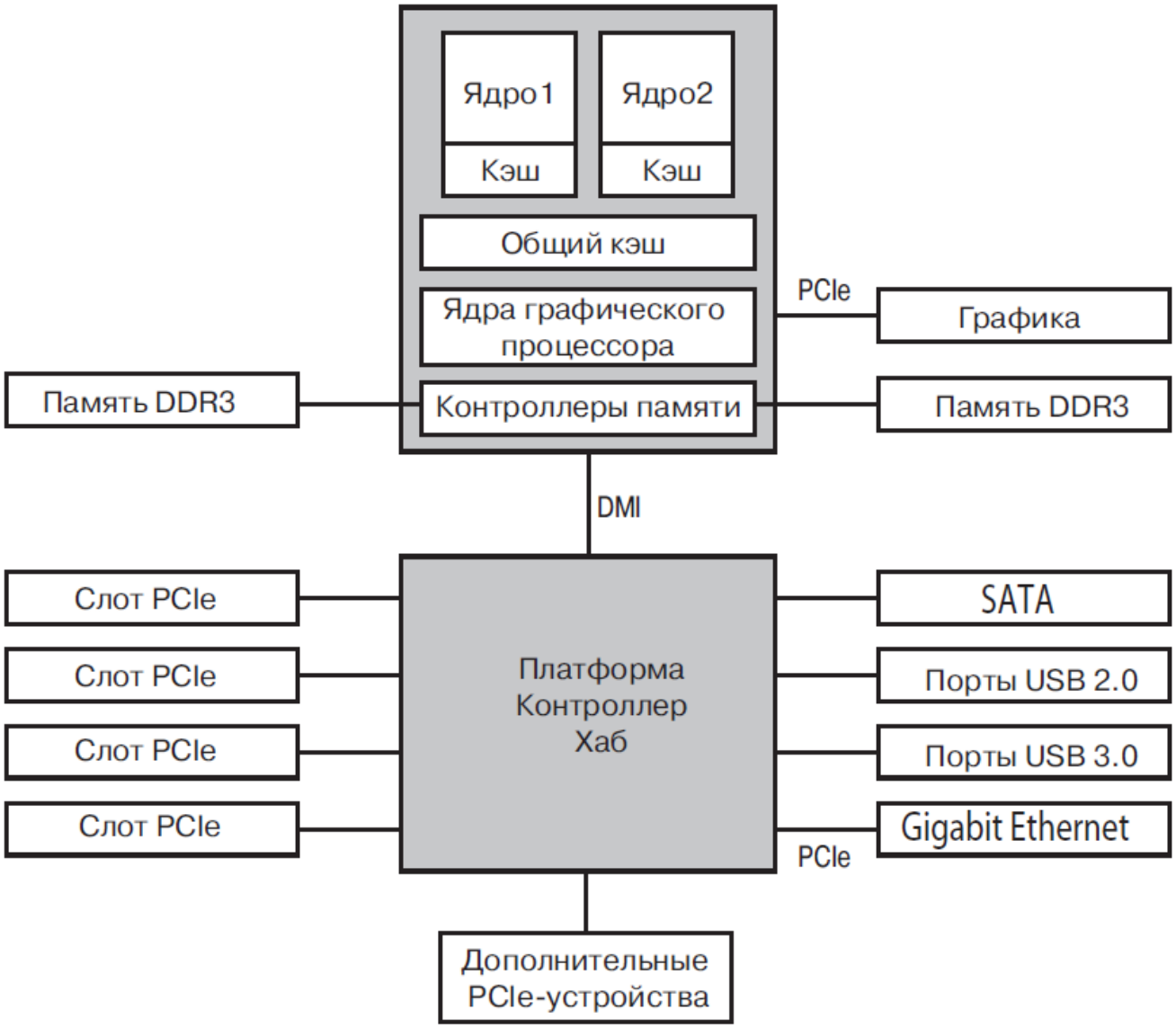
Со временем возможности единой шины по обеспечению всех процессов обмена данными достигли своего предела. Появились дополнительные шины как для более быстродействующих устройств ввода-вывода, так и для обмена данными между процессором и памятью.

У современной массовой x86-система имеется множество шин, например, шина кэш-памяти, шина памяти, шины PCIe, PCI, USB, SATA и DMI. Основная шина – PCI (Peripheral Component Interconnect — интерфейс периферийных устройств) пришла на замену исходной шине ISA (Industry Standard Architecture — стандартная промышленная архитектура). Она имеет архитектуру параллельной шины, что предполагает отправку каждого слова данных по нескольким проводникам. В обычных шинах PCI одно 32-разрядное число отправляется по 32 параллельным проводникам.

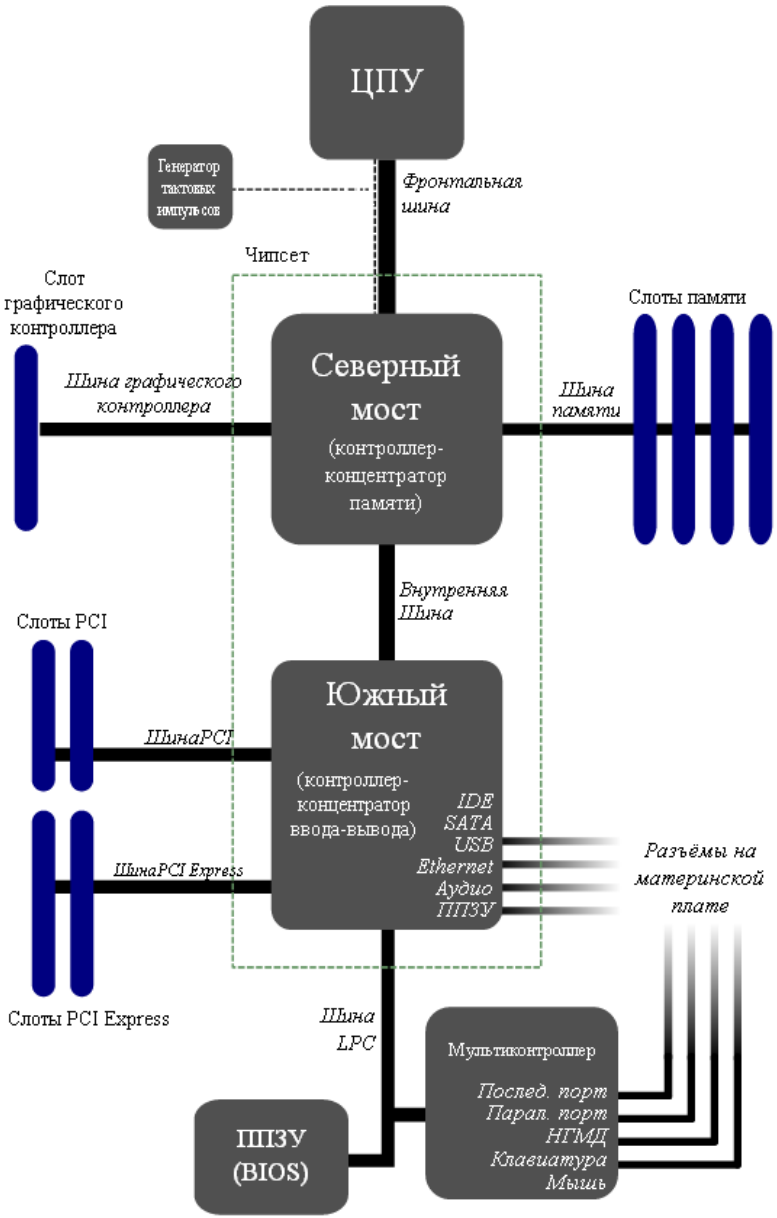
Шина PCIe, разработанная Intel, использует выделенные непосредственные соединения типа «точка — точка» используя последовательную архитектуру. Сообщения передаются по одному соединению – дорожке (lane). Но параллелизм, поскольку параллельно могут действовать сразу несколько дорожек.

Центральный процессор общается с памятью через быструю шину **DDR3** (DDR3 SDRAM – double-data-rate three synchronous dynamic random access memory) — синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных, третье поколение), со внешним графическим устройством — через шину PCIe (PCI Express), а со всеми остальными устройствами — через концентратор по шине DMI (Direct Media Interface — интерфейс непосредственной передачи данных).

Концентратор в свою очередь соединяет все другие устройства, используя для обмена данными с USB-устройствами универсальную последовательную шину, для обмена данными с жесткими дисками и DVD-приводами — шину SATA и для передачи Ethernet-кадров — шину PCIe.



Структура системы семейства x86



Чипсет на материнской плате

Шина USB (Universal Serial Bus — универсальная последовательная шина) была разработана для подключения к компьютеру устройств ввода-вывода. У USB 3.0 с скорость передачи данных – 5 Гбит/с. USB является централизованной шиной, в которой главное (корневое) устройство опрашивает устройства ввода-вывода каждую миллисекунду, чтобы узнать, есть ли у них данные для передачи.

SCSI (Small Computer System Interface — интерфейс малых вычислительных систем) является высокоскоростной шиной, предназначенной для высокопроизводительных дисков, сканеров и других устройств, нуждающихся в значительной пропускной способности. Скорость передачи данных может достигать 640 Мбайт/с.

ОС должна знать о том, какие периферийные устройства подключены к компьютеру, и сконфигурировать эти устройства. Это требование заставило корпорации Intel и Microsoft разработать для PC-совместимых компьютеров систему, называемую ***plug and play*** (подключи и работай). Технология plug and play заставляет систему автоматически собирать информацию об устройствах ввода-вывода, централизованно присваивая уровни запросов на прерывания и адреса ввода-вывода, а затем сообщать каждой карте, какие значения ей присвоены.

1.8. Загрузка компьютера

На материнской плате находится программа, которая называется базовой системой ввода-вывода — ***BIOS*** (Basic Input Output System). BIOS содержит низкоуровневое программное обеспечение ввода-вывода, включая процедуры считывания состояния клавиатуры, вывода информации на экран и осуществления дискового ввода-вывода.

При начальной загрузке компьютера BIOS начинает работать первой. BIOS определяет устройство, с которого будет вестись загрузка, по очереди проверив устройства из списка, сохраненного в CMOS-памяти. С загрузочного устройства в память считывается первый сектор, а затем выполняется записанная в нем программа. Обычно эта программа проверяет таблицу разделов, которая находится в конце загрузочного сектора, чтобы определить, какой из разделов имеет статус активного. Затем из этого раздела считывается вторичный загрузчик, который в свою очередь считывает из активного раздела и запускает ОС.

После этого ОС запрашивает BIOS, чтобы получить информацию о конфигурации компьютера. Как только в ее распоряжении окажутся все драйверы устройств, ОС загружает их в ядро. Затем она инициализирует свои таблицы, создает все необходимые ей фоновые процессы и запускает программу входа в ОС или графический интерфейс пользователя.