ТЕМА 1.2. АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

В данной теме рассматриваются следующие вопросы:

- шинная архитектура.
- компоненты архитектуры (процессор, память, внешние устройства), их назначение и взаимодействие;
- задачи операционной системы по управлению и организации работы компьютера;
- система прерываний.

Лекции – 2 часа, лабораторные занятия – 2 часа, самостоятельная работа – 2 часа.

Экзаменационные вопросы:

- Компоненты архитектуры вычислительных систем, их назначение и взаимодействие
- Система прерываний

1.2.1. Шинная архитектура

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 содержащими 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).

Наличие интеллектуальных контроллеров внешних устройств стало важной отличительной чертой машин третьего и четвертого поколения.

Контроллер можно рассматривать как специализированный процессор, управляющий работой "вверенного ему" внешнего устройства по специальным встроенным программам обмена. Такой процессор имеет собственную систему команд. Например, контроллер на гибких магнитных дисках (дисковода) умеет позиционировать головку на нужную дорожку диска, читать или записывать сектор, форматировать дорожку и т.п. Результаты выполнения каждой операции заносятся во внутренние регистры памяти контроллера, которые могут быть в дальнейшем прочитаны центральным процессором накопителя.

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. Последний получает возможность "заниматься своим делом", т.е. выполнять программу дальше (если по данной задаче до завершения обмена ничего сделать нельзя, то можно в это время решать другую...).

Для связи между отдельными функциональными узлами ЭВМ используется **общая шина** (англ. bus, часто ее называют **магистралью**).

Шина — это несколько проводников, соединяющих несколько устройств. Шины можно разделить на категории в соответствии с выполняемыми функциями. Они могут быть внутренними по отношению к процессору и служить для передачи данных в АЛУ и из АЛУ, а могут быть внешними по отношению к процессору и связывать процессор с памятью или устройствами ввода-вывода. Каждый тип шины обладает определенными свойствами и к каждому из них предъявляются определенные требования.

Первые персональные компьютеры имели одну внешнюю шину, которая называлась системной. Она состояла из нескольких медных проводов (от 50 до 100), которые встраивались в материнскую плату. На материнской плате на одинаковых расстояниях друг от друга находились разъемы для микросхем памяти и устройств ввода-вывода. Современные персональные компьютеры обычно содержат специальную шину между центральным процессором и памятью и по крайней мере еще одну шину для устройств ввода-вывода (рис. 1.2.1).

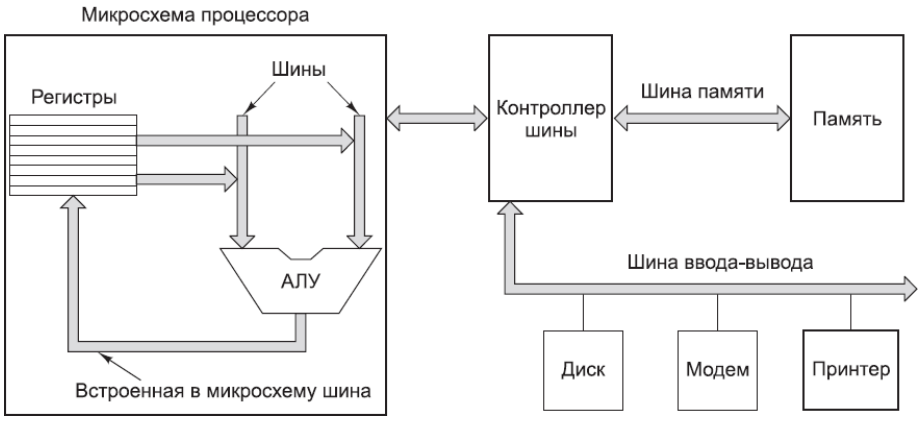

Рис. 1.2.1. Компьютерная система с несколькими шинами

Любая шина состоит из трех частей:

- шина данных, по которой передается информация;
- шина адреса, определяющая, кому передаются данные;
- шина управления, регулирующая процесс обмена информацией.

Также может присутствовать шина питания.

Рис. 1.2.2. Структура общей шины компьютера

Шина данных

По этой шине данные передаются между различными устройствами. Разрядность шины данных определяется разрядностью процессора, т. е. количеством двоичных разрядов, которые процессор обрабатывает за один такт. Со времени создания первого персонального компьютера (1975 г.), разрядность шины данных увеличилась с 8 до 64 бит.

Шина адреса

Каждая ячейка оперативной памяти имеет свой адрес. Адрес передается по адресной шине. Разрядность шины адреса определяет адресное пространство процессора, т. е. количество ячеек оперативной памяти, которые могут иметь уникальные адреса. Количество адресуемых ячеек памяти равно 2^n, где n – разрядность шины адреса.

В первых персональных компьютерах разрядность шины адреса составляла 16 бит, в современных персональных компьютерах разрядность шины адреса составляет 64 бита.

Шина управления

По шине управления передаются сигналы, определяющие характер обмена информацией по магистрали. Сигналы управления определяют, какую операцию - считывание или запись информации из памяти — нужно производить, синхронизируют обмен информацией между устройствами и т. д.

Устройство, управляющее магистралью — обычно центральное процессорное устройство (ЦПУ) или, проще говоря, процессор.

Некоторые устройства, соединенные с шиной, являются **активными** и могут инициировать передачу информации по шине, тогда как другие являются **пассивными** и ждут запросов. Активное устройство называется **задающим**, пассивное — **подчиненным**. Роли могут меняться в зависимости от задач. Например, когда центральный процессор требует от контроллера диска считать или записать блок информации, центральный процессор действует как задающее устройство, а контроллер диска — как подчиненное. Контроллер диска может действовать как задающее устройство, когда он командует памяти принять данные, которые считал с диска.

Отметим, что существуют модели компьютеров, у которых шины данных и адреса для экономии объединены. У таких машин сначала на шину выставляется адрес, а затем через некоторое время данные; для какой именно цели используется шина в данный момент, определяется сигналами на шине управления.

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.

Так как 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.

На практике структура с единственной общей шиной применяется только для ЭВМ с небольшим количеством внешних устройств. 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.

Примеры внутренних компьютерных шин: ISA, PCI.

Примеры внешних компьютерных шин: IDE, USB, SCSI, SATA.

1.2.2. Компоненты архитектуры (процессор, память, внешние устройства), их назначение и взаимодействие

Основы учения об архитектуре вычислительных машин заложил выдающийся американский математик Джон фон Нейман (1903-1957). Вместе с коллегами он продемонстрировал преимущества двоичной системы для технической реализации, удобство и простоту выполнения в ней арифметических и логических операций [1].

Еще одной поистине революционной идеей, значение которой трудно переоценить, является предложенный Нейманом принцип "хранимой программы". Первоначально программа задавалась путем установки перемычек на специальной коммутационной панели. Это было весьма трудоемким занятием: например, для изменения программы "ЭНИАК" требовалось несколько дней (в то время как собственно расчет не мог продолжаться более нескольких минут — выходили из строя лампы). Нейман первым догадался, что программа может также храниться в виде набора нулей и единиц, причем в той же самой памяти, что и обрабатываемые ей числа. Отсутствие принципиальной разницы между программой и данными дало возможность ЭВМ самой формировать для себя программу в соответствии с результатами вычислений.

Фон Нейман не только выдвинул основополагающие принципы логического устройства ЭВМ, но и предложил ее структуру. Классическая архитектура ЭВМ, построенная по принципу фон Неймана (фон-неймановская архитектура) и реализованная в вычислительных машинах двух (трех) поколений, представлена ниже (рис. 1.2.3) и содержит следующие основные блоки:

- арифметическо-логическое устройство (АЛУ), выполняющее арифметические и логические операции;
- управляющее устройство (УУ), организующее процесс выполнения программ;
- внешнее запоминающее устройство (ВЗУ), или память, для хранения программ и данных;
- оперативное запоминающее устройство (ОЗУ);
- устройства ввода и вывода информации (УВВ).

Рис. 1.2.3. Классическая архитектура ЭВМ, построенная по принципу фон Неймана

Как ранее было описано, существенное повышение быстродействия процессора привело к появлению специализированных контроллеров и общей шины в компьютерах третьего поколения.

Современный компьютер в типичной конфигурации включает в себя:

- Системный блок
  - блок питания
  - корпус
  - материнская плата
  - процессор
  - жесткий диск
  - оперативная память
  - видеоадаптер
  - звуковой адаптер
  - приводы дисков
  - шлейфы, кабели для подключения и связи устройств между собой и т.д.
- Монитор
- Клавиатура
- Мышь
- Периферийные устройства

Процессоры

Центральный процессор — это «мозг» компьютера [2]. Он выбирает команды из памяти и выполняет их. Обычный цикл работы центрального процессора выглядит так: выборка из памяти первой команды, ее декодирование для определения ее типа и операндов, выполнение этой команды, а затем выборка, декодирование и выполнение последующих

команд. Этот цикл повторяется до тех пор, пока не закончится программа. Таким образом программы выполняются.

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

Среди специальных регистров процессора стоит выделить **счетчик команд**, который содержит адрес ячейки памяти со следующей выбираемой командой, указатель стека, который ссылается на вершину текущего стека в памяти и **слово состояния программы** — PSW (Program Status Word). Каждый раз при переключении задач содержимое регистров предыдущей задачи выгружается в оперативную память и загружается содержимое регистров для следующей задачи. Этот процесс называется переключением контекста.

Для повышения производительности процессоров их разработчики давно отказались от простой модели извлечения, декодирования и выполнения одной команды за один цикл. Многие современные процессоры способны одновременно выполнять более одной команды. Подобная организация работы называется **конвейером**. Процессоры с **суперскалярной** архитектурой имеют несколько исполнительных блоков, которые могут выполнять команды не в порядке их следования, что хоть и может повысить быстродействие, но усложняет операционную систему.

Большинство центральных процессоров имеют два режима работы: режим ядра и пользовательский режим (режим пользователя). Обычно режимом управляет специальный бит в слове состояния программы — PSW. При работе в режиме ядра процессор может выполнять любые команды из своего набора и использовать любые возможности аппаратуры. Пользовательские программы всегда работают в режиме пользователя, который допускает выполнение только подмножества команд и дает доступ к определенному подмножеству возможностей аппаратуры. Как правило, в пользовательском режиме запрещены все команды, касающиеся операций ввода-вывода и защиты памяти.

Эмпирический закон Мура гласит, что количество транзисторов на одном кристалле удваивается каждые 18 месяцев. Закон Мура соблюдался в течение трех десятилетий и, как ожидается, будет соблюдаться как минимум еще одно десятилетие. После этого число атомов в транзисторе станет настолько мало, что дальнейшему уменьшению размеров транзистора воспрепятствует усиливающаяся роль законов квантовой механики (отставание уже заметно). Увеличение компьютеров позволяет повышать функциональность микросхем, в том числе и процессоров. Увеличился объем кэш-памяти, была реализована многопоточность (hyperthreading у Intel). Но настоящую революцию произвели многоядерные процессоры, которые содержат несколько полноценных процессоров. Современные графические процессоры содержат сотни или тысячи микроядер (рендеров), которые позволяют эффективно проводить параллельные вычисления не только графических примитивов, но генерацию криптовалюты.

Наиболее популярные производители процессоров для компьютеров сегодня – это Intel и AMD. Между их процессорами есть совместимость на уровне базового набора команд, но есть и собственные наборы команд (рис. 1.2.4).

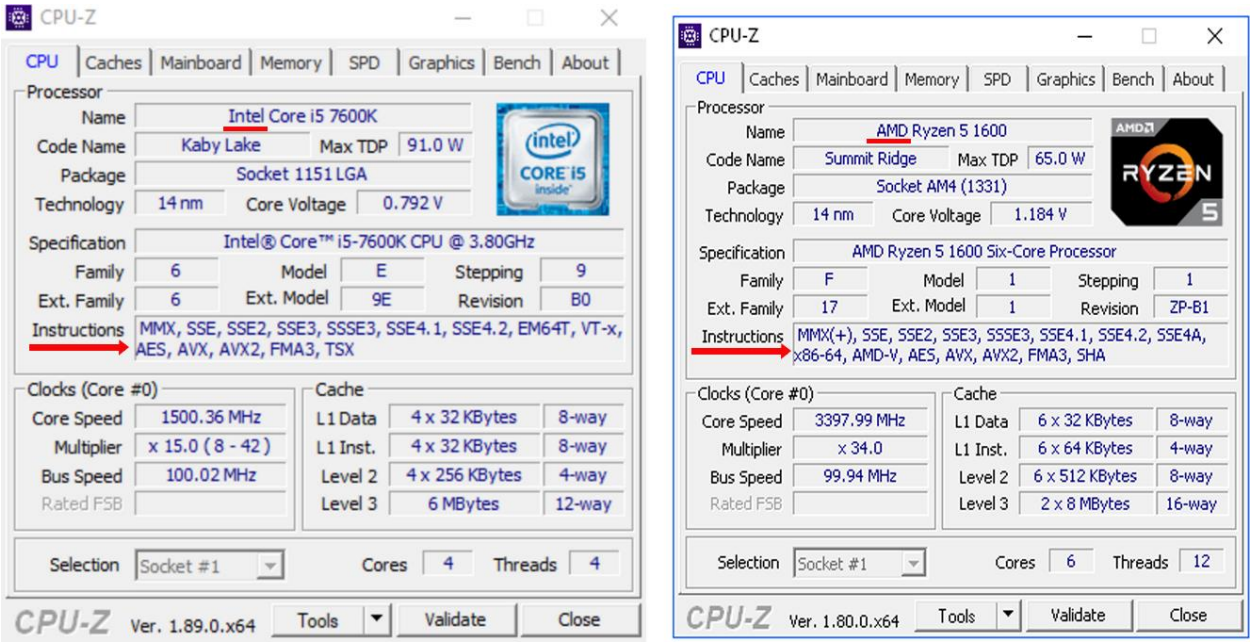

Рис. 1.2.4. Сравнение наборов команд у процессоров Intel и AMD

Есть несколько архитектур процессоров с различными наборами команд. Аббревиатура CISC обозначает Complex Instruction Set Computer, а RISC — Reduced Instruction Set Computer [3].

В конце 1970, когда началась разработка CISC-процессоров, память была очень дорогой. Компиляторы тоже были плохие, а люди писали на ассемблере. Так как память была дорогой, люди искали способ минимизировать использование памяти. Одно из таких решений — использовать сложные инструкции процессора, которые делают много действий. Это также помогло программистам на ассемблере, так как они смогли писать более простые программы, ведь всегда найдется инструкция, которая выполняет то, что нужно.

Через некоторое время это стало сложным. Проектирование декодеров для таких команд стало существенной проблемой. Изначально ее решили с помощью микрокода. Для каждой инструкции из набора создается подпрограмма, которая состоит из простых инструкций и хранится в специальной памяти внутри микропроцессора. Таким образом, процессор содержит небольшой набор простых инструкций. На их основе можно создать множество сложных инструкций из набора команд с помощью добавления подпрограмм в микрокод.

Микрокод хранится в ROM-памяти (Read-Only Memory), которая значительно дешевле оперативной памяти. Следовательно, уменьшение использования оперативной памяти через увеличение использования постоянной памяти — выгодный компромисс.

Какое-то время все выглядело очень хорошо. Но со временем начались проблемы, связанные с подпрограммами в микрокоде. В них появились ошибки. Исправление ошибки в микрокоде в разы сложнее, чем в обычной программе. Нельзя получить доступ к этому коду и протестировать его как обычную программу.

Разработчики стали думать, что существует другой, более простой способ справиться с этими трудностями.

Потом оперативная память стала дешеветь, компиляторы стали лучше, а большинство разработчиков перестало писать на ассемблере. Эти технологические изменения спровоцировали появление философии RISC. Сперва люди анализировали программы и заметили, что большинство сложных инструкций CISC не используются большинством программистов. Разработчики компиляторов затруднялись в выборе правильной сложной

инструкции. Вместо этого они предпочли использовать комбинацию нескольких простых инструкций для решения проблемы.

Идея RISC заключается в замене сложных инструкций на комбинацию простых. Так не придется заниматься сложной отладкой микрокода. Вместо этого разработчики компилятора будут решать возникающие проблемы. Есть разногласия в том, как понимать слово «сокращенный» (reduced) в аббревиатуре RISC. Люди думают, что сокращено количество инструкций. Но более правильная интерпретация — это уменьшение сложности команд.

Одним из аргументов RISC было то, что люди перестали писать ассемблерный код, поэтому необходимо создать набор инструкций, удобный для компиляторов. Архитектура RISC оптимизирована для компиляторов, но не для людей. Поэтому RISC-код может быть непростым для человека. И при использовании RISC часто нужно писать больше команд, чем в случае CISC.

Еще одна основная идея RISC — это конвейеризация. Инструкции разделены на этапы, каждый из которых выполняется примерно одинаковое количество времени.

Если рассмотреть ARM RISC-процессор, то мы обнаружим пятиступенчатый конвейер инструкций.

- **(Fetch) Извлечение** инструкции из памяти и увеличение счетчика команд, чтобы извлечь следующую инструкцию в следующем такте.
- **(Decode) Декодирование** инструкции — определение, что эта инструкция делает. То есть активация необходимых для выполнения этой инструкции частей микропроцессора.
- **(Execute) Выполнение** включает использование арифметико-логического устройства (АЛУ) или совершение сдвиговых операций.
- **(Memory) Доступ к памяти**, если необходимо. Это то, что делает инструкция load.
- **(Write Back) Запись результатов** в соответствующий регистр.

Инструкции ARM состоят из секций, каждая из которых работает с одним из этих этапов, а выполнение этапа обычно занимает один такт. То есть инструкции ARM очень удобно конвейеризировать. Более того, каждая инструкция имеет одинаковый размер, то есть этап Fetch знает, где будет располагаться следующая инструкция, и ему не нужно проводить декодирование.

С инструкциями CISC все не так просто. Они могут быть разной длины. То есть без декодирования фрагмента инструкции нельзя узнать ее размер и где располагается следующая инструкция.

Вторая проблема CISC — сложность инструкций. Многократный доступ к памяти и выполнение множества вещей не позволяют легко разделить инструкцию CISC на отдельные части, которые можно выполнять поэтапно.

Конвейеризация — это особенность, которая позволила первым RISC-процессорам на голову обогнать своих конкурентов в тестах производительности.

Характеристики RISC [4]:

1. Более простые инструкции, следовательно, простое декодирование инструкций.
2. Инструкция состоит из одного слова.
3. Для выполнения инструкции требуется один такт.
4. Большее количество универсальных регистров.
5. Простые режимы адресации.
6. Меньше типов данных.
7. Конвейер может быть реализован.

Характеристики CISC:

1. Сложные инструкции, следовательно, сложное декодирование инструкций.

2. Инструкции обычно имеют размер, превышающий одно слово.

3. Для выполнения инструкции может потребоваться более одного такта.

4. Меньше количества регистров общего назначения, поскольку операции могут выполняться в оперативной памяти.

5. Сложные режимы адресации.

6. Дополнительные типы данных.

Пример. Предположим, нам нужно сложить два 8-битных числа:

CISC-подход: для этого будет одна команда или инструкция, например ADD, которая выполнит задачу.

RISC-подход: здесь программист напишет первую команду загрузки для загрузки данных в регистры, затем он будет использовать подходящий оператор, а затем сохранит результат в нужном месте.

Преимущества RISC:

Более простые инструкции. RISC-процессоры используют меньший набор простых инструкций, что упрощает их декодирование и быстрое выполнение. Это приводит к ускорению времени обработки.

Более быстрое выполнение: поскольку процессоры RISC имеют более простой набор команд, они могут выполнять инструкции быстрее, чем процессоры CISC.

Более низкое энергопотребление: процессоры RISC потребляют меньше энергии, чем процессоры CISC, что делает их идеальными для портативных устройств.

Недостатки RISC:

Требуется больше инструкций: процессорам RISC требуется больше инструкций для выполнения сложных задач, чем процессорам CISC.

Увеличение использования памяти: процессорам RISC требуется больше памяти для хранения дополнительных инструкций, необходимых для выполнения сложных задач.

Более высокая стоимость. Разработка и производство процессоров RISC может быть дороже, чем процессоров CISC.

Преимущества CISC:

Уменьшенный размер кода: процессоры CISC используют сложные инструкции, которые могут выполнять несколько операций, что уменьшает объем кода, необходимого для выполнения задачи.

Более эффективное использование памяти. Поскольку инструкции CISC более сложны, для выполнения сложных задач требуется меньше инструкций, что может привести к более эффективному использованию памяти.

Широкое использование: процессоры CISC используются дольше, чем процессоры RISC, поэтому они имеют большую базу пользователей и более доступное программное обеспечение.

Недостатки CISC:

Медленное выполнение: процессорам CISC требуется больше времени для выполнения инструкций, поскольку они имеют более сложные инструкции и требуют больше времени для их декодирования.

Более сложная конструкция: процессоры CISC имеют более сложные наборы команд, что усложняет их проектирование и производство.

Более высокое энергопотребление: процессоры CISC потребляют больше энергии, чем процессоры RISC, из-за более сложных наборов команд.

RISC-V (RISC-Five), как и RISC и CISC, представляет собой архитектуру набора команд (ISA). Однако это открытая и бесплатная ISA. Это означает, что каждый может внести свой вклад в его развитие, и его использование не будет стоить ни копейки. Это важно, поскольку позволяет мелким производителям устройств создавать оборудование без необходимости платить роялти [5].

RISC-V спроектирован так, чтобы иметь небольшую фиксированную ISA вместе с модульными фиксированными стандартными расширениями, которые хорошо работают с большей частью кода. Это оставляет достаточно места для расширений для конкретных приложений, позволяющих создавать собственные процессоры для конкретных рабочих нагрузок.

Преимущества RISC-V по сравнению с традиционной архитектурой RISC и CISC включают:

1. **Гибкость**. Возможность настройки процессора позволяет инженерам настраивать наборы микросхем большими, маленькими, мощными или легкими в зависимости от конкретных требований устройства.

2. **Инновации**. Компании могут внедрять минимальный набор инструкций и использовать собственные и определенные расширения для создания процессоров для передовых инструментов.

3. **Снижение затрат и ускорение вывода на рынок**. Повторное использование интеллектуальной собственности с открытым исходным кодом помогает снизить стоимость разработки и позволяет компаниям быстрее выводить свои разработки на рынок.

Память

Второй основной составляющей любого компьютера является память [2]. В идеале память должна быть максимально быстрой (работать быстрее, чем производится выполнение одной инструкции, чтобы работа центрального процессора не замедлялась обращениями к памяти), довольно большой и чрезвычайно дешевой. Никакая современная технология не в состоянии удовлетворить все эти требования, поэтому используется другой подход. Система памяти создается в виде иерархии уровней (рис. 1.2.5). Верхние уровни обладают более высоким быстродействием, меньшим объемом и более высокой удельной стоимостью хранения одного бита информации, чем нижние уровни, иногда в миллиарды и более раз.

Рис. 1.2.5. Типичная иерархия памяти. Приведенные значения весьма приблизительны

Верхний уровень состоит из внутренних регистров процессора. Они выполнены по той же технологии, что и сам процессор, и поэтому не уступают ему в быстродействии. Следовательно, к ним нет и задержек доступа. В общем случае разрядность внутренних регистров процессора совпадает с разрядностью процессора.

Затем следует кэш-память, которая управляется главным образом аппаратурой. Оперативная память 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, находящейся внутри процессора или очень близко к нему. Когда программе нужно считать слово из памяти, аппаратура кэша проверяет, нет ли нужной строки в кэш-памяти. Если строка в ней имеется, то происходит результативное обращение к кэш-памяти (cache hit — кэш-попадание), запрос удовлетворяется за счет кэш-памяти без отправки запроса по шине к оперативной 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. Доступ к кэшу первого уровня осуществляется без задержек, а доступ к кэшу второго уровня требует задержки в один или два такта.

Следующей в иерархии, изображенной на рис. 1.2.5, идет оперативная память. Это главная рабочая область системы памяти машины. Оперативную память часто называют оперативным запоминающим устройством (ОЗУ), или памятью с произвольным доступом (Random Access Memory (RAM)). В настоящее время блоки памяти имеют объем от сотен мегабайт до нескольких гигабайт, и этот объем стремительно растет. Все запросы процессора, которые не могут быть удовлетворены кэш-памятью, направляются к оперативной памяти.

Дополнительно к оперативной памяти многие 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. Эта разновидность памяти характеризуется высоким быстродействием и дешевизной. На некоторых компьютерах в ПЗУ размещается начальный загрузчик, используемый для их запуска. Такой же памятью, предназначенной для осуществления низкоуровневого управления устройством, оснащаются некоторые контроллеры ввода-вывода.

Существуют также 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. При этом они обладают еще одним дополнительным свойством — возможностью исправлять ошибки в содержащихся в них программах путем перезаписи занимаемых ими участков памяти.

Флеш-память также обычно используется как носитель информации в портативных электронных устройствах. По быстродействию флеш-память занимает промежуточное положение между ОЗУ и диском. Также, в отличие от дисковой памяти, если флеш-память стирается или перезаписывается слишком часто, она приходит в негодность.

Еще одна разновидность памяти — CMOS-память, которая является энергозависимой. Во многих компьютерах CMOS-память используется для хранения текущих даты и времени. CMOS-память и схема электронных часов, отвечающая за отсчет времени, получают питание от миниатюрной батарейки (или аккумулятора), поэтому значение текущего времени исправно обновляется, даже если компьютер отсоединен от внешнего источника питания. CMOS-память также может хранить параметры конфигурации, указывающие, например, с какого диска системе следует загружаться. Потребление энергии CMOS-

памятью настолько низкое, что батарейки, установленной на заводе-изготовителе, часто хватает на несколько лет работы.

Диски

Магнитный жесткий диск состоит из одной или нескольких металлических пластин, вращающихся со скоростью 5400, 7200, 10 000 или 15 000 оборотов в минуту. Информация записывается на диск в виде последовательности концентрических окружностей. В каждой заданной позиции привода каждая из головок может считывать кольцеобразный участок, называемый дорожкой (рис. 1.2.6). Из совокупности всех дорожек в заданной позиции привода составляется цилиндр. Каждая дорожка поделена на определенное количество секторов, обычно по 512 байт на сектор. На современных дисках внешние цилиндры содержат больше секторов, чем внутренние [6].

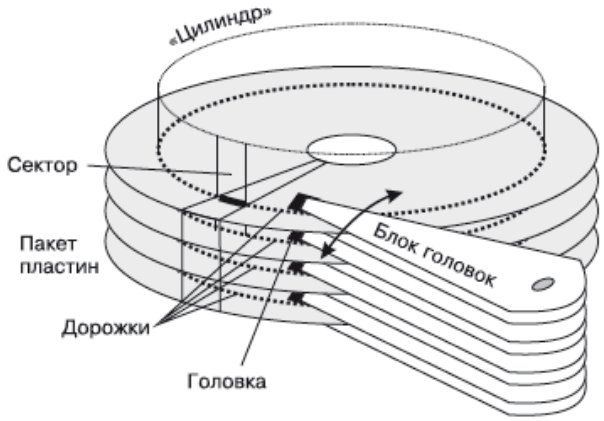

Рис. 1.2.6. Цилиндры, головки и сектора [7]

Бурный рост объема жестких дисков вынудил повысить плотность записи информации. При этом размер сектора в 512 байтов приводит к неэффективному использованию поверхности пластины, поэтому появились диски с размером сектора 4 KiB. Для совместимости со старым программным обеспечением такие диски обычно имеют режим эмуляции с размером сектора 512 байтов, а в дисковых утилитах можно встретить обозначения 512, 512e и 4K.

> *Примечание. 1000 и 1024 близки, но не равны. Во избежание путаницы принято использовать привычные приставки кило-, мега-, гига и т. д. только как десятичные. Соответствующие им двоичные приставки киби-, меби-, гиби- и т. д. Например, диск с объемом 4 TB (четыре терабайта) виден операционной системой как диск объемом 3,7 TiB.*

С целью дальнейшей увеличения плотности записи дорожки стали частично накладывать друг на друга. Такие диски эффективны при последовательной записи данных и редком их чтении (например, системы видеорегистрации). При черепичной магнитной записи (SMR — Shingled Magnetic Recording) на жесткий диск дорожки размещаются друг над другом, подобно черепице на крыше (рис. 1.2.7) [8]. Это позволяет повысить плотность записи. Увеличивается количество дорожек на дюйм (TPI). При перпендикулярной магнитной записи (PMR), используемой в большинстве современных дисков, данные размещаются на параллельных дорожках. Увеличение TPI путем уменьшения расстояния между дорожками благодаря технологии SMR открывает огромный простор для роста емкости жестких дисков. Конечный продукт физически выглядит и ведет себя как обычный жесткий диск с PMR, но при этом имеет большую емкость. Аббревиатура CMR (Conventional Magnetic Recording, обычная магнитная запись) является антонимом к SMR и по факту эквивалентна PMR.

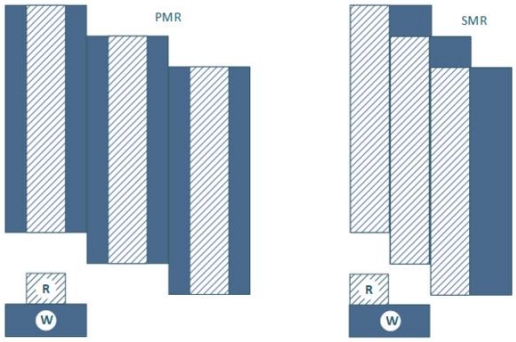

Рис. 1.2.7. Перпендикулярная и черепичная магнитная запись

Для простоты изложения, к жестким дискам также отнесем твердотельные накопители (SSD — Solid State Disks), которые хранят данных во флэш-памяти и не имеют движущихся частей. У них то же назначение, что и у магнитных дисков, и с точки зрения операционной системы выглядят так же. SSD-диски обладают бóльшим быстродействием, чем магнитные, но и стоимость таких дисков выше. Гибридные диски используют обе технологии.

SSD-диски могут в одной ячейке хранить несколько бит данных. Чем больше бит данных в одной ячейке, тем выше емкость, ниже цена и меньше количество циклов перезаписи. SLC-ячейки выдерживают до 100000 циклов перезаписи, но хранят в одной ячейке ровно один бит. MLC — 2 бита на ячейку, TLC — 3 бита на ячейку, QLC — 4 бита на ячейку, PLC — 5 битов на ячейку.

По интерфейсу подключения современные диски можно разделить на SAS, SATA, PCIe (NVMe) и M.2

Как правило, SATA используется в домашних рабочих станциях, второй – в серверах, поэтому технологии между собой не конкурируют, отвечая разным требованиям.

SAS (Serial Attached SCSI) – последовательный интерфейс подключения устройств хранения данных, разработанный на основе параллельного SCSI для исполнения того же набора команд. Используется преимущественно в серверных системах.

SATA (Serial ATA) – последовательный интерфейс обмена данными, базирующийся на основе параллельного PATA (IDE). Применяется в домашних, офисных, мультимедийных ПК и ноутбуках.

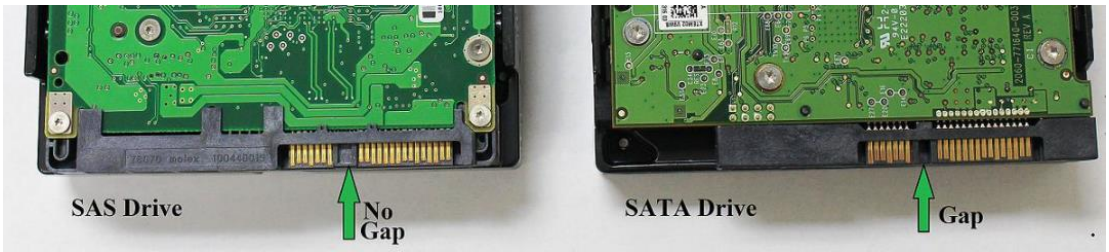

Рис. 1.2.8. Сравнение разъема SAS и SATA

Шины PCI и PCIe

Еще можно найти компьютеры, использующие шину PCI (Peripheral Component Interconnect), разработанную компанией Intel. Существует много различных конфигураций шины PCI. Наиболее типичная из них показана на рис. 1.2.9. В такой конфигурации центральный процессор взаимодействует с контроллером памяти по выделенному высокоскоростному соединению. Таким образом, контроллер соединяется с памятью непосредственно, то есть передача данных между центральным процессором и памятью происходит не через шину PCI. Другие периферийные устройства подсоединяются прямо к шине PCI. Машина такого типа обычно содержит 2 или 3 пустых

разъема PCI, чтобы покупатели имели возможность подключать карты PCI для новых периферийных устройств).

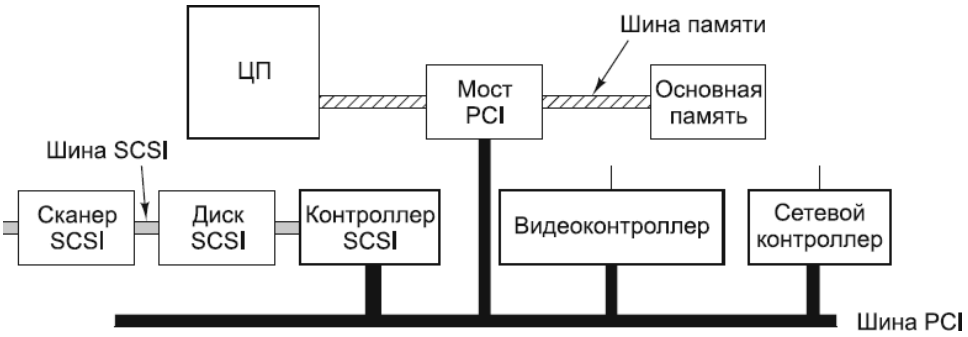

Рис. 1.2.9. Персональный компьютер с шиной PCI (контроллер SCSI является PCI-устройством)

Сегодня шина PCI 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.

Если шина PCI представляла собой обновленную версию старой шины ISA с более высокой скоростью и разрядностью параллельно передаваемых данных, PCIe представляет кардинальное изменение по сравнению с шиной PCI. Собственно, это вообще не шина, а одноранговая сеть, использующая разрядно-последовательные линии и коммутацию пакетов. Архитектура PCIe изображена на рис. 1.2.10.

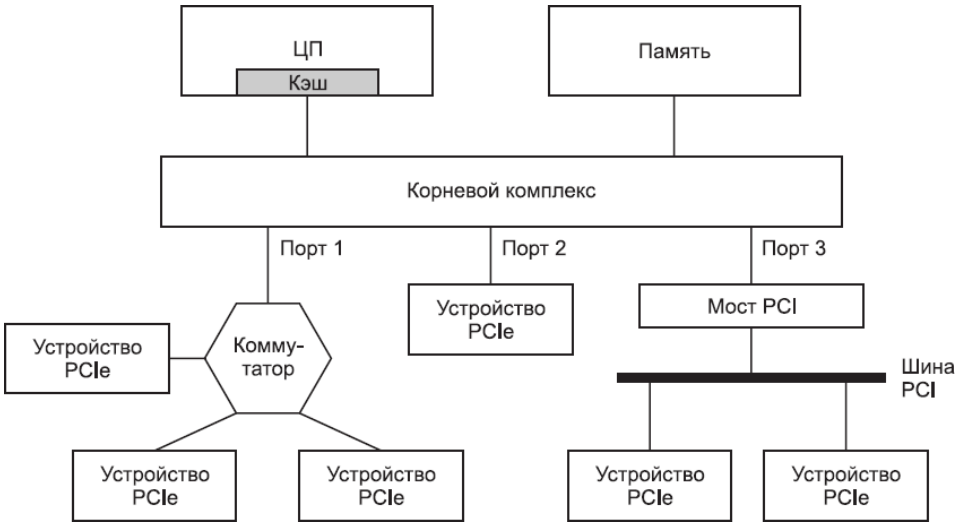

Рис. 1.2.10. Архитектура системы PCIe с тремя портами PCI

Во-первых, в шине PCIe соединения между устройствами являются последовательными, то есть имеют разрядность в один бит вместо 8, 16, 32 или 64 бит. Это позволяет значительно увеличить пропускную способность шины. Например, шины PCI работают на максимальной тактовой частоте 66 МГц. При передаче 64 бит за такт скорость передачи данных составляет 528 Мбайт/с. При тактовой частоте 8 Гбит/с, даже в случае последовательной передачи, скорость передачи по шине PCIe составляет 1 Гбайт/с.

Устройство может иметь до 32 проводных пар, называемых трактами (lanes) или дорожками. На большинстве материнских плат имеется 16-трактовый разъем для графической карты, что для PCIe 3.0 обеспечивает пропускную способность в 16 Гбайт/с — примерно в 30 раз больше, чем у графических карт PCI.

Во-вторых, все взаимодействия являются одноранговыми. Когда процессор хочет обратиться к устройству, он отправляет этому устройству пакет и обычно получает ответ. Пакет проходит через корневой комплекс на материнской плате, а затем передается устройству — как правило, через коммутатор (или для устройств PCI — через мост PCI).

Шина USB

Первоначально шина USB была разработана для низкоскоростных устройств (клавиатур, мышей, фотоаппаратов, сканеров, цифровых телефонов и т. д.) [2]. Общая пропускная способность первой версии шины (USB 1.0) составляла 1,5 Мбит/с. Версия 1.1 работает на скорости 12 Мбит/с, что было вполне достаточно для принтеров, цифровых камер и многих других устройств.

Версия 2.0 поддерживает устройства со скоростью до 480 Мбит/с, что позволило подключать внешние диски, веб-камеры высокого разрешения и сетевые интерфейсы. В версии USB 3.0 скорость возросла до 5 Гбит/с. USB 3.2 поддерживает скорость до 20 Гбит/с. Последняя версия стандарта, USB4, поддерживает скорость до 40 Гбит/с.

Шина USB состоит из корневого хаба (root hub), который подключается к главной шине. Этот корневой хаб (часто называемый корневым концентратором) содержит разъемы для кабелей, которые могут подсоединяться к устройствам ввода-вывода или к дополнительным хабам, чтобы увеличить количество разъемов. Таким образом, топология шины USB представляет собой дерево с корнем в корневом хабе, который находится внутри компьютера. Коннекторы кабеля со стороны устройства отличаются от коннекторов со стороны хаба, чтобы пользователь случайно не подсоединил кабель другой стороной.

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.

Видеокарты

Видеокарта – устройство, преобразующее цифровую информацию в форму, пригодную для дальнейшего вывода на экран монитора. Обычно видеокарта выполнена в виде печатной платы (плата расширения) и вставляется в слот расширения материнской платы, универсальный либо специализированный (AGP, PCI Express), но может быть реализована и на системной плате.

Видеокарты не ограничиваются простым выводом изображения, они имеют встроенный графический процессор, который может производить дополнительную обработку, снимая эту задачу с центрального процессора компьютера.

Также имеет место тенденция использовать вычислительные возможности графического процессора для решения неграфических задач (например, добычи криптовалюты).

Современные графические процессоры очень эффективно обрабатывают и отображают компьютерную графику, благодаря специализированной конвейерной архитектуре они намного эффективнее в обработке графической информации, чем типичный центральный процессор.

Графический процессор в современных видеоадаптерах применяется в качестве ускорителя трёхмерной графики.

Графический процессор изначально создавался как многопоточная структура с множеством ядер и предназначался для обработки компьютерной графики, поэтому рассчитан на массивные параллельные вычисления.

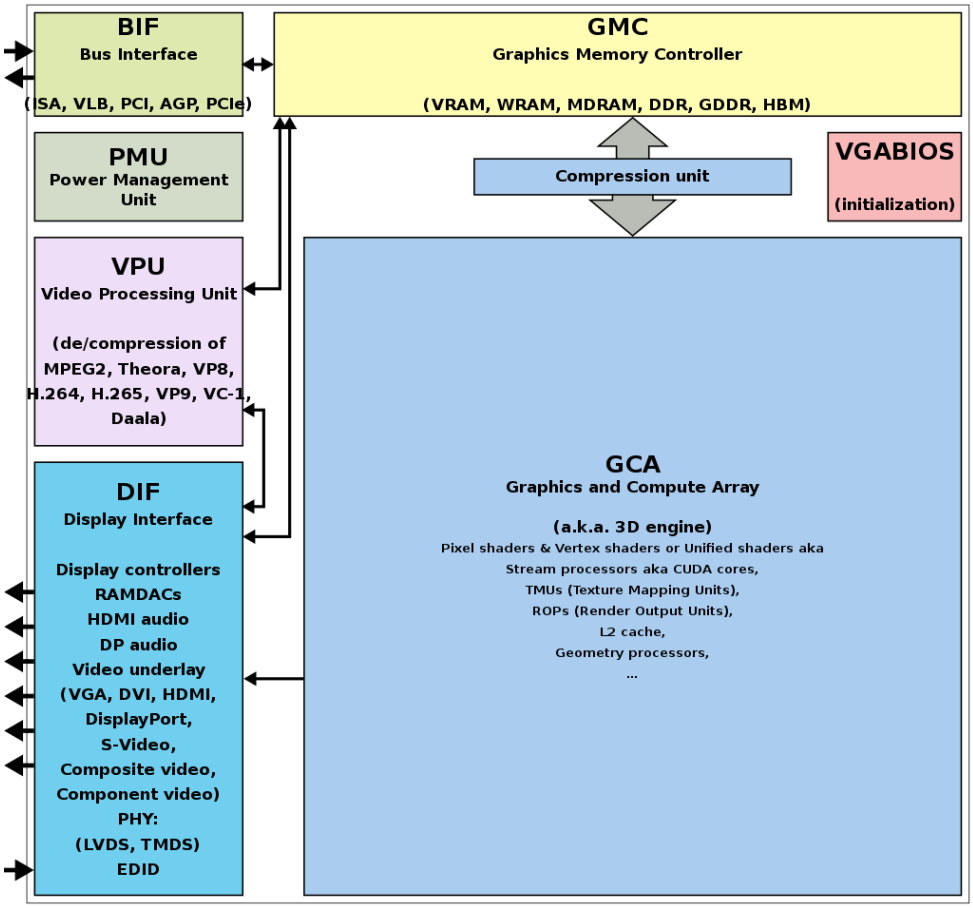

Рис. 1.2.11. Блок-схема графического процессора

Разъём HDMI меньше по размеру, чем DVI, а также поддерживает передачу многоканальных цифровых аудиосигналов.

При помощи пассивного переходника порт DVI совместим с HDMI, но только в пределах режимов, которым достаточно single-link DVI.

Максимальное разрешение DVI 2560×1600

DisplayPort имеет пропускную способность вдвое большую, чем Dual-Link DVI, низкое напряжение питания (3,3 В вместо 5 В, используемых в DVI и HDMI); поддерживает максимальное разрешение 16K (15360 × 8640)

Thunderbolt комбинирует интерфейсы PCI Express (PCIe) и DisplayPort (DP) в одном кабеле.

Для подключения монитора к видеокарте можно использовать как аналоговые, так и цифровые интерфейсы. Разъемы VGA характерного синего цвета еще можно встретить на старом оборудовании. Но для использования высокого разрешения на мониторах нужно использовать цифровые интерфейсы (рис 1.2.12).

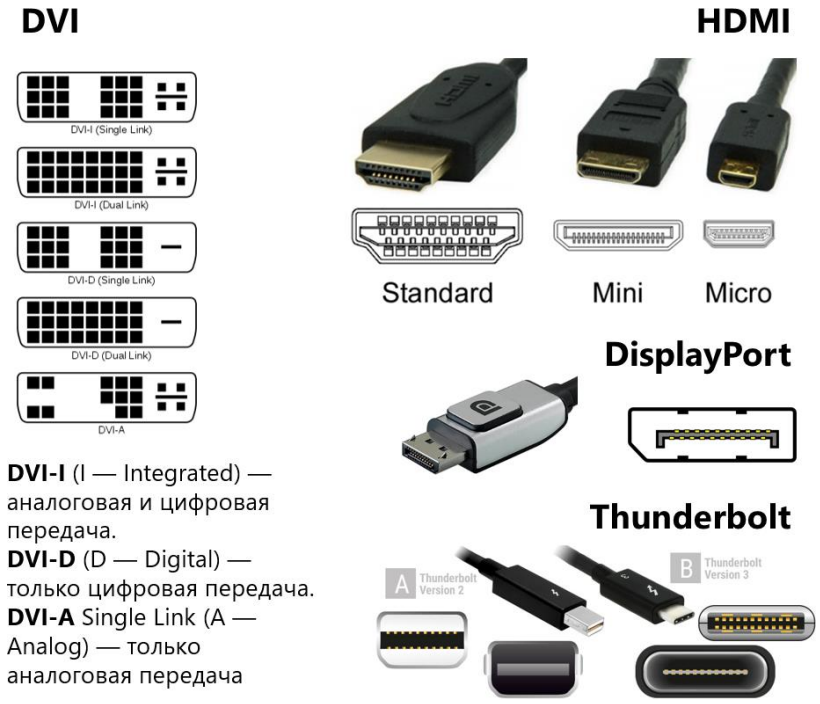

Рис 1.2.12. Цифровые интерфейсы видеокарт

В компьютере видеокарта подключается с одной из шин компьютера. В современных компьютерах это обычно PCI Express, хотя в старых компьютерах использовались и другие шины.

Шина **ISA** (Industry Standart Architecture) имела разрядность 8/16 бит и работала на частоте 8 МГц. Пиковая пропускная способность составляла чуть больше 5,5 МиБ/с. Этого более чем хватало для отображения текстовой информации и игр с 16-цветной графикой.

Шина **VLB** (VESA Local Bus) имела разрядность 32 бит и работала на внешней тактовой частоте процессора 80486 (25 МГц до 50 МГц). Пиковая пропускная способность около 130 МиБ/с.

Шина **PCI** (Periferal Component Interconnect) имела разрядность 32/32 бит и работала на частоте 33 МГц, появилась на материнских платах для процессоров Pentium. Пиковая пропускная способность 133 МиБ/с (сравнимо с VLB). Но она была удобнее и, в конце концов, вытеснила шину VLB и на материнских платах для процессоров класса 486.

Шина **AGP** (Accelerated Graphics Port — ускоренный графический порт). Разрядность шины AGP составляет 32 бит, рабочая частота — 66 МГц. Пиковая пропускная способность в режиме 1x — 266 МиБ/с.

PCI Express – последовательный интерфейс (расширение шины PCI). Пропускная способность может достигать нескольких десятков Гб/с.

Хотя существует можество различных производителей видеокарт, все они используют графические процессоры одного из производителей: Nvidia, AMD или Intel. Также многие микропроцессоры Intel и AMD имеют встроенный графический контроллер. Встроенная графика позволяет построить компьютер без отдельных плат видеоадаптеров, что сокращает стоимость и энергопотребление систем. Данное решение обычно используется в ноутбуках и настольных компьютерах нижней ценовой категории, а также для бизнес-компьютеров, для которых не требуется высокий уровень производительности графической подсистемы. У графического ядра процессора нет отдельной видеопамяти и под эти нужды система выделяет определённый объём из оперативной памяти.

Intel HD Graphics — семейство интегрированных графических процессоров, используемых в процессорах компании Intel. До создания Intel HD Graphics интегрированная графика Intel была встроена в северный мост материнской платы. Это были серии Intel Extreme Graphics и Intel Graphics Media Accelerator. В рамках Platform

Controller Hub северный мост был ликвидирован, а часть его функциональности, в том числе графический процессор, была интегрирована в центральный процессор.

Гибридные процессоры AMD — это процессоры со встроенной графикой. То есть под крышкой процессора помимо самого вычислительного кристалла расположено графическое ядро выполняющее функции видеокарты. Такому процессору не нужна отдельная видеокарта для вывода изображения, а подключение к монитору осуществляется через разъём на материнской плате.

Существует несколько API для программирования графических приложений.

Direct3D — это интерфейс программирования графических приложений (API) для Microsoft Windows. Direct3D, являющийся частью DirectX, используется для рендеринга трехмерной графики в приложениях, где важна производительность, например в играх. Direct3D использует аппаратное ускорение, если оно доступно на видеокарте, что позволяет аппаратно ускорять весь конвейер 3D-рендеринга или даже только частичное ускорение. Текущая версия 12.0

OpenCL (Open Computing Language) — фреймворк для написания компьютерных программ, связанных с параллельными вычислениями на различных графических и центральных процессорах, а также FPGA. В OpenCL входят язык программирования, который основан на стандарте языка программирования Си C99, и API. OpenCL обеспечивает параллелизм на уровне инструкций и на уровне данных и является осуществлением техники GPGPU. OpenCL является полностью открытым стандартом, его использование не облагается лицензионными отчислениями.

OpenGL (Open Graphics Library) — спецификация, определяющая платформонезависимый (независимый от языка программирования) программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику. Разрабатывается в США и Европе, имеет тип лицензий GNU-/EU/.

Mantle — спецификация низкоуровневого API, разработанная компанией AMD в качестве альтернативы Direct3D и OpenGL

Vulkan — кроссплатформенный API для 2D- и 3D-графики, потомок Mantle, использует промежуточный двоичный формат SPIR-V, что позволяет компилировать шейдеры на этапе разработки

1.2.3. Задачи операционной системы по управлению и организации работы компьютера

С точки зрения программиста, операционная система — это программа, добавляющая ряд команд и функций к командам и функциям, выполняемым аппаратным обеспечением [6]. Выделим три важные группы таких функций.

1. **Виртуальная память**. Механизм виртуальной памяти используется многими операционными системами. Она позволяет создать впечатление, будто у машины больше памяти, чем есть на самом деле.

2. **Файловый ввод-вывод**. Это понятие более высокого уровня, чем команды ввода-вывода, которые мы рассматривали в предыдущей главе.

3. **Параллелизм** (как организовано одновременное выполнение нескольких процессов, обмен информацией и синхронизация). Понятие процесса является очень важным, и мы подробно рассмотрим его далее в этой главе. Под процессом можно понимать работающую программу и всю информацию об ее состоянии (памяти, регистрах, счетчике команд, вводе-выводе и т. д.).

Если рассматривать более подробно, то самыми важными задачами управления компьютерным оборудованием, осуществляемыми операционной системой, являются следующие:

- Параллельное функционирование модулей ввода, вывода информации и процессора.
- Организация кэширования данных и выполнение согласования скоростей информационного обмена.
- Разбиение модулей и информационных данных среди процессов.
- Организация удобной работы логического интерфейса между модулями и оставшейся частью системы.
- Организация поддержки различных устройств с обеспечением возможности просто их добавить.
- Режим динамической загрузки и выгрузки драйверов.
- Обеспечение поддержки набора файловых систем.

Модули, предназначенные для ввода и вывода информации, подразделяются на следующие типы:

- Модули, ориентированные на работу с блоками.
- Модули, ориентированные на работу с байтами.

Ориентированное на работу с блоками оборудование сохраняет данные в блоках, имеющих фиксированный размер и свой уникальный адрес. Примером такого устройства может служить жёсткий диск. Модули, работающие с байтами, не имеют адресации и не обеспечивают возможность поиска информации. Они только могут генерировать или потреблять байтовую очерёдность. В качестве примера таких устройств можно привести сетевые адаптеры, терминалы и так далее.

На рис. 1.2.13 SCSI-диск /dev/sda и SCSI CD-ROM /dev/sr0 отмечены как блочные устройства, универсальные устройства SCSI /dev/sg0 и /dev/sg1 и терминал /dev/tty0 — как символьные.

```
root@nnk4:~# dir -la /dev/tty0 /dev/s?? /dev/dvd
lrwxrwxrwx  1 root root      3 сен  5 17:12 /dev/dvd -> sr0
brw-rw----  1 root disk   8, 0 сен  5 18:04 /dev/sda
crw-rw----+ 1 root cdrom 21, 0 сен  5 16:55 /dev/sg0
crw-rw----  1 root disk  21, 1 сен  5 16:55 /dev/sg1
brw-rw----+ 1 root cdrom 11, 0 сен  5 17:12 /dev/sr0
crw--w----  1 root tty    4, 0 сен  5 16:55 /dev/tty0
```

Рис. 1.2.13 Первый символ в выводе команды `ls` обозначает тип устройства (**c** – character, **b** – block)

Главным принципом построения программного обеспечения ввода и вывода информации является разбиение его на отдельные уровни. При этом нижние уровни должны обеспечивать защиту особенностей своего оборудования от влияния верхних уровней, которые призваны только осуществлять удобное интерфейсное обслуживание пользователей.

Ещё одним базовым вопросом в организации программ информационного ввода и вывода считается обработка ошибок. Общеизвестно, что обработку ошибок нужно осуществлять максимально близко к оборудованию. Когда контроллер находит ошибку чтения, то ему необходимо сделать попытку её коррекции. Если эта попытка окажется неудачной, то дальше коррекцией ошибки занимается драйвер модуля. Часто ошибки пропадают при повторном выполнении операции ввода или вывода информации. Но когда ликвидацию ошибки не удалось выполнить на нижнем уровне, идёт сообщение об ошибке на верхний уровень.

Программное обеспечение ввода и вывода информации делится на следующие уровни:

- Уровень обработки прерываний.
- Уровень драйверов оборудования.
- Уровень независимого от оборудования слоя операционной системы.

- Пользовательский уровень программного обеспечения.

В программном наборе операционной системы лишь драйвер оборудования имеют информацию о фактических свойствах какого-либо модуля. Значительная часть программного обеспечения ввода и вывода информации выполнена как независимая от оборудования. Конкретная грань между драйверами и независимыми от оборудования приложениями назначается системой, поскольку отдельные функции, которые возможно исполнить независимой методикой, по факту реализуются в формате драйверов для увеличения эффективности или по иным причинам.

Также следует упомянуть технологию, которая позволяет создавать несколько сред или выделенных ресурсов из единой физической аппаратной системы называется **виртуализация** [9]. Программное обеспечение, гипервизор, напрямую подключается к этой аппаратной системе и позволяет разбить ее на отдельные, безопасные среды – виртуальные машины. По идее, гипервизор должен распределять аппаратные ресурсы между виртуальными машинами так, чтобы процессы выполнялись быстрее. Физическая машина с гипервизором называется **хостом**, а виртуальные машины, которые используют ресурсы данного хоста – **гостями**. Для них ресурсами являются процессор, память, хранилище. Для получения доступа к этим ресурсам операторы управляют виртуальными экземплярами.

1.2.4. Система прерываний

Для получения услуг от операционной системы пользовательская программа должна осуществить системный вызов, который перехватывается внутри ядра и вызывает операционную систему. Инструкция перехвата TRAP 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. Системный вызов следует считать специальной разновидностью инструкции вызова процедуры, у которой есть дополнительное свойство переключения из пользовательского режима в режим ядра.

Конечно, компьютеры имеют и другие системные прерывания, не предназначенные для перехвата инструкции выполнения системного вызова. Но большинство других системных прерываний вызываются аппаратно для предупреждения о возникновении исключительных ситуаций, например, попыток деления на нуль или исчезновении порядка при операции с плавающей точкой. Во всех случаях управление переходит к операционной системе, которая и должна решать, что делать дальше. Иногда работа программы должна быть прервана сообщением об ошибке. В других случаях ошибка может быть проигнорирована (например, при исчезновении порядка числа оно может быть принято равным нулю). Наконец, когда программа заранее объявила, что с некоторыми из возможных ситуаций она собирается справляться самостоятельно, управление должно быть возвращено программе, чтобы она сама разрешила возникшую проблему.

Все прерывания и особые ситуации имеют уникальные идентификационные номера. Эти номера называются векторами прерываний и лежат в пределах от 0 до 255. Векторы от 0 до 31 отведены для особых ситуаций и немаскируемого прерывания, причем некоторые из них зарезервированы и не должны использоваться программами. Векторы от 32 до 255 свободны для любого использования пользовательскими программами и внешними устройствами [10].

Классификация прерываний

Существует два источника поступления прерываний (interrupt) и три типа особых ситуаций (исключений, exception). Кроме того, различают внутренние (программные, software) и внешние (аппаратные, hardware) источники генерации прерываний и особых ситуаций.

В зависимости от источника, прерывания делятся на [11]:

- **аппаратные** — возникают как реакция микропроцессора на физический сигнал от некоторого устройства (клавиатура, системные часы, клавиатура, жесткий диск и т.д.), по времени возникновения эти прерывания асинхронны, т.е. происходят в случайные моменты времени;
- **программные** — вызываются искусственно с помощью соответствующей команды из программы (int), предназначены для выполнения некоторых действий операционной системы, являются синхронными;
- **исключения** — являются реакцией микропроцессора на нестандартную ситуацию, возникшую внутри микропроцессора во время выполнения некоторой команды программы (деление на ноль, прерывание по флагу TF (трассировка)).

Общая классификация прерываний:

- **внешние** — вызываются внешними по отношению к микропроцессору событиями (это группа аппаратных прерываний) Вложенных прерываний нет!
- **внутренние** — возникают внутри микропроцессора во время вычислительного процесса (по существу это исключительные ситуации и программные прерывания).

Внешние прерывания возникают по сигналу какого-нибудь внешнего устройства.

Внешние прерывания подразделяются на не-маскируемые и маскируемые.

В связи с тем, что существуют два специальных внешних сигнала среди входных сигналов процессора, при помощи которых можно прервать выполнение текущей программы и тем самым переключить работу центрального процессора. Это сигналы NMI (no mask interrupt, немаскируемое прерывание) и INTR (interrupt request, запрос на прерывание).

Маскируемые прерывания генерируются контроллером прерываний по заявке определенных периферийных устройств. Контроллер прерываний (в первых PC был выполнен в виде специальной микросхемы i8259A) поддерживает восемь уровней (линий) приоритета; к каждому уровню «привязано» одно периферийное устройство. Именно маскируемые прерывания часто называют аппаратными прерываниями.

В ПК, начиная с IBM PC AT, построенных на базе микропроцессора i80286, используются два контроллера прерываний i8259A; они соединяются последовательно каскадным образом, что увеличивает количество внешних источников прерываний до 15 (каждая по 8).

Немаскируемые прерывания (говорят, что оно одно, т.к. подается на вывод микропроцессора NMI) инициируют источники, требующие безотлагательного вмешательства со стороны микропроцессора.

Особые ситуации, генерируемые процессором, подразделяются на три типа — ошибки, ловушки и сбои. В зависимости от типа особой ситуации различается реакция процессора на ее возникновение.

- **Ошибка** (Fault) — это особая ситуация, которая может быть исправлена обработчиком особой ситуации. При встрече ошибки состояние процессора сохраняется в том виде, каким оно было до начала выполнения команды, инициировавшей генерацию ошибки, а значения CS:EIP, указывающие на эту команду сохраняются в стеке обработчика. Прерванная программа после

исправления ошибки может быть продолжена непосредственно с команды, вызвавшей эту ошибку.

- **Ловушка** (Trap) — особая ситуация, которая генерируется после выполнения соответствующей команды. В этом случае сохраняемые в стеке значения CS:EIP, указывают на команду, которая будет выполняться вслед за командой, вызвавшей ловушку; например, если ловушка произошла во время команды JMP, то сохраненные значения CS:EIP указывают на команду, являвшуюся целью команды JMP.
- **Сбой** (Abort) — это особая ситуация, которая не допускает точную локализацию вызвавшей ее команды и не допускает перезапуска. Сбои используются для сообщений о некоторых ошибках, таких как: технические неисправности и наличие некорректных значений в системных таблицах.

Обработка прерываний в процессорах x86

Обработка прерывания в реальном режиме производится в три этапа [11]:

1. Прекращение выполнения текущей программы.

Для этого необходимо сохранить содержимое регистров, так как они являются ресурсами, разделяемыми между программами. Обязательными для сохранения являются регистры **cs**, **ip**, **flags** (пара CS:IP содержит адрес команды, с которой необходимо начать выполнение после возврата, flags — состояние флагов после выполнения последней команды прерванной программы). Эти регистры сохраняются микропроцессором автоматически. Сохранение остальных регистров - должно обеспечиваться программистом. Наиболее удобным местом хранения регистров является стек.

После сохранения регистров в стеке микропроцессор сбрасывает бит флага IF (т.е.=0) (при этом в стеке записан регистр flags с еще установленным IF). Этим предотвращается возможность возникновения вложенных внешних прерываний и порча регистров исходной программы вследствие неконтролируемых действий со стороны программы - обработчика вложенного прерывания. После того как необходимые действия по сохранению контекста завершены, обработчик аппаратного прерывания может разрешить вложенные прерывания командой sti.

2. Переход к выполнению и выполнение программы обработки прерывания.

Здесь определяется источник прерывания и вызывается соответствующий обработчик прерывания. В реальном режиме микропроцессора допускается 256 источников - по количеству элементов таблицы векторов прерываний.

Структура одного элемента в таблице векторов прерываний:

WORD (2 байта) — значение смещения начала программы-обработчика прерывания от начала кодового сегмента

WORD (2 байта) — значение базового адреса сегмента, в котором находится программа-обработчик.

Фактически, на втором этапе микропроцессор:

- по номеру источника прерывания определяет смещение в таблице векторов прерываний;
- помещает первые два байта в регистр IP;
- помещает вторые два байта в регистр CS;
- передает управление по адресу CS:IP.

Далее выполняется сама программа обработки прерывания.

(Она тоже может быть прервана поступлением запроса от более приоритетного источника. Все источники прерывания имеют приоритеты.)

3. Возврат управления прерванной программе

Необходимо привести стек в состояние, в котором он был сразу после передачи управления данной процедуре. Для этого программист должен указать необходимые действия по восстановлению регистров и очистке стека. Этот участок необходимо защитить от возможного искажения содержимого регистров (в результате появления аппаратного прерывания) с помощью команды **cli**.

Последние команды в обработчике прерывания — **sti**, **iret** (**sti** — разрешить аппаратные прерывания, устанавливает флаг IF=1, не имеет операндов), **iret** — извлечь последовательно три слова из стека и поместить их соответственно в регистры ip, cs, flags.

Некоторые наиболее важные прерывания x86

0	ошибка деления
1	прерывание пошагового режима (трассировка)
2	аппаратное немаскируемое прерывание
3	прерывание для трассировки (отладки)
8	прерывание интервального таймера, возникает 18,2 раза в секунду
9	прерывание от клавиатуры
13	прерывание от IDE-контроллера
16-31	обращение к BIOS и т.п.
32-95	обращение к MS-DOS

Обработка прерываний в RISC-V

Общая архитектура системы RISC-V для сигнализации прерываний зависит от того, построена ли она в основном для прерываний, сигнализируемых сообщениями (MSI, message-signaled interrupts), или для более традиционных проводных прерываний. В системах с полной поддержкой MSI каждый Hart имеет контроллер входящего MSI (IMSIC, Incoming MSI Controller), который служит для харта (hart, Hardware Thread) собственным частным контроллером прерываний для внешних прерываний. И наоборот, в системах, основанных в основном на традиционных проводных прерываниях, харт не имеет IMSIC. Ожидается, что более крупные системы, особенно с устройствами PCI, будут полностью поддерживать MSI, предоставляя HART-интерфейсы IMSIC, тогда как многие более мелкие системы, возможно, по-прежнему будут лучше обслуживаться с помощью проводных прерываний и более простых HART-интерфейсов без IMSIC.

Если харты RISC-V не имеют входящих контроллеров MSI, сигналы о внешних прерываниях передаются на харты через выделенные провода. В этом случае усовершенствованный контроллер прерываний уровня платформы (Advanced PLIC, Platform-Level Interrupt Controller) действует как традиционный центральный концентратор для прерываний, маршрутизации и определения приоритетов внешних прерываний для каждого порта, как показано на рисунке 1.2.14. Прерывания могут быть выборочно маршрутизированы либо на уровень машины, либо на уровень супервизора на каждом порту.

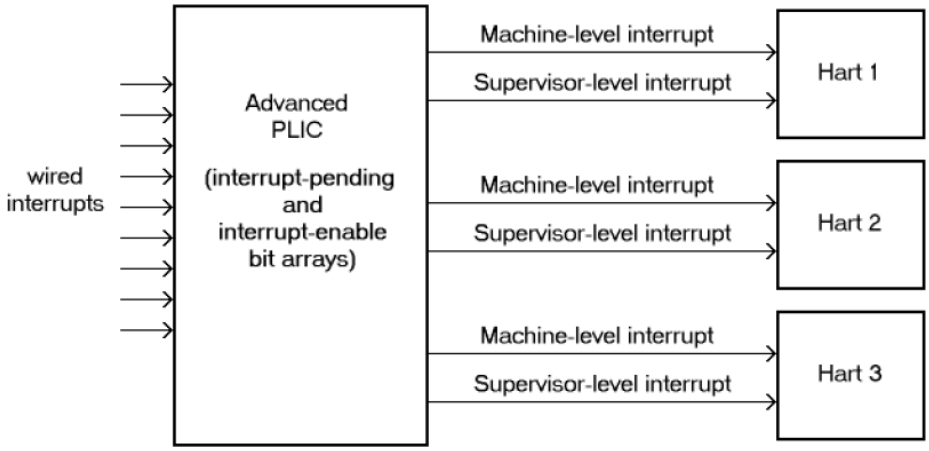

Рис. 1.2.14. Традиционная доставка проводных прерываний на харты без поддержки MSI

Чтобы иметь возможность получать прерывания, сигнализируемые сообщениями (MSI), каждый порт RISC-V должен иметь контроллер входящего MSI (IMSIC), как показано на рисунке 1.2.15. По сути, прерывание, сигнализируемое сообщением, — это просто запись в память по определенному адресу, который аппаратное обеспечение воспринимает как указание на прерывание. С этой целью каждому IMSIC назначается один или несколько отдельных адресов в адресном пространстве машины, и когда производится запись на один из этих адресов в ожидаемом формате, принимающая IMSIC интерпретирует запись как внешнее прерывание для соответствующего порта.

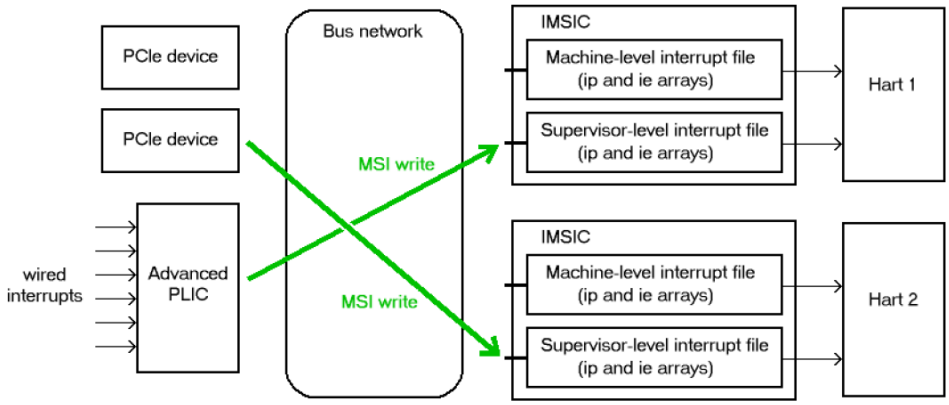

Рис. 1.2.15. Доставка прерываний с помощью MSI, когда у хартов есть IMSIC для их приема

Поскольку все IMSIC имеют уникальные адреса в физическом адресном пространстве машины, каждый IMSIC может получать записи MSI от любого агента (харта или устройства) с разрешением на запись в него. IMSIC имеют отдельные адреса для MSI, направленные на уровни машины и супервизора, отчасти для того, чтобы возможность сигнализировать о прерываниях на каждом уровне привилегий можно отдельно предоставлять или запрещать путем управления разрешениями на запись по разным адресам, а отчасти для лучшей поддержки виртуализации (притворяясь, что один уровень привилегий является более высоким уровнем). MSI, предназначенные для харта на определенном уровне привилегий, записываются внутри IMSIC в виде файла прерываний, который состоит в основном из массива битов ожидания прерывания и соответствующего массива битов разрешения прерывания, причем последний указывает, какие индивидуальные прерывания готов в данный момент получать харт.

Когда харты в системе RISC-V имеют IMSIC, система обычно по-прежнему содержит Advanced PLIC, но ее роль меняется. Вместо передачи сигналов о прерываниях напрямую по проводам, как показано на рис. 1.2.14, Advanced PLIC преобразует входящие проводные прерывания в записи MSI, которые отправляются на порты через их модули

IMSIC. Каждый MSI отправляется на один целевой порт в соответствии с конфигурацией PLIC, установленной программным обеспечением.

Помимо внешних прерываний от устройств ввода-вывода, привилегированная архитектура RISC-V определяет несколько других основных классов прерываний для харта. Прерывания по таймеру Привилегированной архитектуры остаются полностью поддерживаемыми, а программные прерывания остаются, по крайней мере, частично поддерживаемыми, хотя ни одно из них не показано на рисунках 1.2 14 и 1.2.15.

Усовершенствованная архитектура прерываний (Advanced Interrupt Architecture) обеспечивает значительную поддержку локальных прерываний в харте, при этом харт по сути прерывает себя в ответ на асинхронные события, обычно ошибки. Локальные прерывания остаются внутри харта (или близко к нему), поэтому, как и стандартные прерывания таймера RISC-V и программные прерывания, они не проходят через Advanced PLIC или IMSIC.

Привилегированная архитектура RISC-V присваивает каждой причине прерывания на харте отдельный основной идентификационный номер (major identity number), который представляет собой код исключения. Причины прерываний, стандартизированные Привилегированной архитектурой, имеют основные идентификаторы в диапазоне от 0 до 15, а номера 16 и выше официально доступны для стандартов платформы или для индивидуального использования. Расширенная архитектура прерываний претендует на дополнительные полномочия над четными идентификационными номерами в диапазоне 16–62, оставляя нечетные числа в этом диапазоне и все основные идентификаторы 63 и выше по-прежнему свободными для индивидуального использования. Таблица на рис.1.2.16 характеризует все основные идентификаторы прерываний с этим расширением.

Major identity	Minor identity	
0	–	*Reserved by Privileged Architecture*
1	–	Supervisor software interrupt
2	–	Virtual supervisor software interrupt
3	–	Machine software interrupt
4	–	*Reserved by Privileged Architecture*
5	–	Supervisor timer interrupt
6	–	Virtual supervisor timer interrupt
7	–	Machine timer interrupt
8	–	*Reserved by Privileged Architecture*
9	Determined by	Supervisor external interrupt
10	external interrupt	Virtual supervisor external interrupt
11	controller	Machine external interrupt
12	–	Supervisor guest external interrupt
13–15	–	*Reserved by Privileged Architecture*
even numbers, 16–62	–	Local interrupts, *or reserved*
odd numbers, 17–63	–	*Designated for custom use*
≥ 64	–	*Designated for custom use*

Рис.1.2.16. Основные и младшие идентификаторы для всех причин прерываний. Основные идентификаторы 0–15 относятся к сфере привилегированной архитектуры RISC-V.

Прерывания от большинства устройств ввода-вывода передаются в харт внешним контроллером прерываний для, которым является либо IMSIC HART (рис. 1.2.15), либо расширенная PLIC (рис. 1.2.14). Как показано на рис.1.2.15, все внешние прерывания на данном уровне привилегий имеют один основной идентификационный номер: 11 для уровня машины, 9 для уровня супервизора и 10 для уровня VS. Внешние прерывания по

разным причинам отличаются друг от друга по младшим идентификационным номерам, предоставляемым внешним контроллером прерываний.

Другие причины прерываний, помимо внешних прерываний, также могут иметь свои собственные младшие идентификационные номера.

Каждое сигнальное прерывание доставляется только к одному харту с одним уровнем привилегий, обычно тем или иным образом определяемым программным обеспечением. В отличие от некоторых других архитектур, расширенная архитектура прерываний RISC-V не обеспечивает стандартного аппаратного механизма для широковещательной или многоадресной рассылки прерываний на несколько хартов.

Для локальных прерываний и для любых «виртуальных» прерываний, которые программное обеспечение вводит в более низкие уровни привилегий на харте, прерывания являются полностью локальными на харте и никогда не видны другим хартам. Прерывания таймера привилегированной архитектуры RISC-V также уникально привязаны к отдельным хартам. Для других прерываний, получаемых хартом от источников за пределами харта, каждый сигнал прерывания (доставленный по проводу или через MSI) настраивается программным обеспечением для передачи только одному харту.

Чтобы отправить межпроцессорное прерывание (IPI) нескольким хартам назначения, исходному устройству достаточно выполнить цикл, отправляя отдельный IPI каждому харту назначения.

[1]. **Лекция. Архитектура ЭВМ**

https://studfile.net/preview/3186677/

2. Таненбаум, Э. Современные операционные системы. / Э. Таненбаум, Х. Бос. — 4-е изд. — СПб.: Питер, 2015. — 1120 с.

3. Что означает RISC и CISC?

https://habr.com/ru/companies/selectel/articles/542074/

4. Computer Organization | RISC and CISC

https://www.geeksforgeeks.org/computer-organization-risc-and-cisc/

5. RISC vs. CISC Architecture: Which is Better?

https://www.per-international.com/news-and-insights/risc-vs-cisc-architecture-which-is-better

6 Таненбаум, Э Архитектура компьютера

7. Геометрия и адресация

https://wm-help.net/lib/b/book/1767203810/25

8. Что такое черепичная магнитная запись SMR и стоит ли ее избегать?

https://interface31.ru/tech_it/2022/12/chto-takoe-cherepichnaya-magnitnaya-zapis-smr-i-stoit-li-ee-izbegat.html

9. Разбираемся, как работают операционные системы
https://proglib.io/p/how-os-work

10. Прерывания и особые ситуации: Типы прерываний
http://www.club155.ru/x86exceptions-types

11. Система прерываний 32-разрядных микропроцессоров i80x86. Работа системы прерываний в реальном режиме
http://mf.grsu.by/UchProc/livak/po/lections/lec12.htm