Помимо предоставления абстракций, таких как процессы, адресные пространства и файлов, операционная система также контролирует все операции ввода-вывода (ввода/вывода) компьютера. пороки. Это должен проблема команды к в устройства, ловить прерывает, и справиться ошибки. Он также должен обеспечивать интерфейс между устройствами и остальной частью системы. который прост и удобен в использовании. Насколько это возможно, интерфейс должен быть одинаково для всех устройств (независимость от устройства). Код ввода/вывода представляет собой важную дробная часть из в общее действующий система. Как в действующий система управляет ввод/вывод является в предмет этой главы.

Эта глава организована следующим образом. Сначала мы рассмотрим некоторые принципы схемы аппаратного обеспечения ввода-вывода, а затем и программного обеспечения ввода-вывода в целом. Программное обеспечение ввода-вывода может быть структурированы слоями, каждый из которых имеет четко определенную задачу. мы посмотрим на эти слои к видеть Какие Они делать и как Они поместиться все вместе.

Следующий, мы воля смотрю в несколько ввод/вывод устройства в деталь: диски, часы, клавиатуры, и дисплеи. Для каждого устройства мы рассмотрим его аппаратное и программное обеспечение. Наконец-то, мы воля рассмотреть власть управление.

# [ПРИНЦИПЫ ИЗ ввод/вывод АППАРАТНОЕ ОБЕСПЕЧЕНИЕ](#_bookmark6)

Разные люди смотрят на аппаратное обеспечение ввода-вывода по-разному. Инженеры-электрики посмотрите на него с точки зрения чипов, проводов, блоков питания, моторов и всего остального. кал составные части который включать в аппаратное обеспечение. Программисты смотрю в в интерфейс

**337**

представленные программному обеспечению — команды, которые принимает аппаратное обеспечение, функции, которые оно выполняет, и ошибки, о которых можно сообщить обратно. В этой книге нас интересует с программированием устройств ввода-вывода, а не с их проектированием, созданием или обслуживанием, поэтому нас интересует, как аппаратное обеспечение запрограммировано, а не как оно работает внутри. Никогда- тем не менее, программирование многих устройств ввода-вывода часто тесно связано с их внутреннюю работу. В следующих трех разделах мы дадим немного общего сведения об аппаратном обеспечении ввода-вывода в части программирования. Его можно рассматривать как обзор и расширение из вступительный материал в сек. 1.3.

## [ввод/вывод Устройства](#_bookmark6)

Устройства ввода-вывода можно условно разделить на две категории: **блочные устройства** и **характерные устройства** . Блочное устройство — это устройство, которое хранит информацию в фиксированном размере. блоков, каждый со своим адресом. Общие размеры блоков варьируются от 512 до 65 536 байт. Все переводы осуществляются в единицах одного или нескольких полных (последовательных) блоков. Существенным свойством блочного устройства является возможность чтения или записи каждого блокировать независимо из все в Другой те. Жесткий диски, Блю рей диски, и USB палочки являются обычными блочными устройствами.

Если очень внимательно присмотреться, то граница между устройствами, которые имеют блочный адрес- в состоянии и те, которые не являются четко не определены. Все согласны с тем, что диск — это блокировать адресное устройство, потому что независимо от того, где рука в данный момент находится, она всегда можно искать другой цилиндр, а затем ждать, пока нужный блок повернется под голову. Теперь рассмотрим старомодный ленточный накопитель, который до сих пор иногда используется для создание резервных копий дисков (потому что ленты дешевы). Ленты содержат последовательность блоки. Если ленточному накопителю дать команду прочитать блок *N* , он всегда может перемотать ленту и идите вперед, пока не дойдете до блока *N.* Эта операция аналогична disk выполняет поиск, за исключением того, что это занимает гораздо больше времени. Кроме того, это может быть, а может и не быть. можно перезаписать один блок в середине ленты. Даже если бы можно было использовать ленты как блочные устройства с произвольным доступом, что несколько натягивает вопрос: они находятся обычно этим не пользовался способ.

Другим типом устройства ввода-вывода является символьное устройство. Символьное устройство эры или принимает а ручей из символы, без внимание к Любые блокировать состав. Это является не адресуемый и не имеет операции поиска. принтеры, сетевые интерфейсы, мыши (для указания), крысы (для психологических лабораторных экспериментов) и большинство других устройств который не дискообразные увидимся как символьные устройства.

Эта схема классификации несовершенна. Некоторые устройства не подходят. Часы, например, не адресуются блоками. Они также не генерируют и не принимают характер потоки. Все, что они делают, это вызывают прерывания через четко определенные промежутки времени. Отображение памяти экраны тоже не подходят к модели. Как и сенсорные экраны, если уж на то пошло. Все еще, модель блочных и символьных устройств является достаточно общей, чтобы ее можно было использовать в качестве основу для того, чтобы сделать часть программного обеспечения операционной системы, работающего с устройствами ввода-вывода, в зависимый. Файловая система, например, имеет дело только с абстрактными блочными устройствами и листья в аппаратно-зависимый часть к Нижний уровень програмное обеспечение.

Устройства ввода/вывода охватывают огромный диапазон скоростей, что оказывает значительное давление на программное обеспечение хорошо работает на многих порядках скорости передачи данных. Фигура 5-1 показаны скорости передачи данных некоторых распространенных устройств. Большинство этих устройств, как правило, получать Быстрее время идет.

|  |  |
| --- | --- |
| **Устройство** | **Данные показатель** |
| Клавиатура | 10 байт/сек |
| мышь | 100 байт/сек |
| 56К модем | 7 КБ/сек |
| Сканер при разрешении 300 dpi | 1 МБ/сек |
| Цифровой видеокамера | 3,5 МБ/сек |
| 4x Блю рей диск | 18 МБ/сек |
| 802.11n Беспроводная связь | 37,5 МБ/сек |
| USB 2.0 | 60 МБ/сек |
| FireWire 800 | 100 МБ/сек |
| Гигабит Ethernet | 125 МБ/сек |
| САТА 3 диск привод | 600 МБ/сек |
| USB 3.0 | 625 МБ/сек |
| SCSI Ультра 5 автобус | 640 МБ/сек |
| Одиночный ряд PCIe 3.0 автобус | 985 МБ/сек |
| Удар молнии 2 автобус | 2,5 ГБ/сек |
| СОНЕТ Сеть ОС-768 | 5 ГБ/сек |

**Фигура 5-1.** Некоторые типичный устройство, сеть, и автобус данные тарифы.

## [Устройство Контроллеры](#_bookmark6)

Блоки ввода-вывода часто состоят из механического компонента и электронного компонента. нет. Можно разделить две части, чтобы обеспечить более модульную и генеральный дизайн. электронный составная часть является называется в **устройство контроллер** или **адаптер** . На персональных компьютерах он часто имеет форму чипа на основе процессора. плату или печатную плату, которую можно вставить в слот расширения (PCIe). механический составная часть является в устройство сам. Этот расположение является показано в Инжир. 1-6.

На плате контроллера обычно имеется разъем, в который вводится кабель само устройство может быть подключено. Многие контроллеры могут обрабатывать два, четыре или даже восемь одинаковых устройств. Если интерфейс между контроллером и устройством является постоянным dard, либо официальный стандарт ANSI, IEEE или ISO, либо стандарт де-факто, затем компании могут производить контроллеры или устройства, соответствующие этому интерфейсу. Многие ком- Компании, например, производят дисководы, соответствующие SATA, SCSI, USB, Thunder- болт, или FireWire (IEEE 1394) интерфейсы.

Интерфейс между контроллером и устройством часто очень низкоуровневый. один. Например, диск может быть отформатирован с 2 000 000 секторов по 512 байт. на трек. Однако на самом деле с накопителя выходит последовательный битовый поток, начиная с ing с **преамбулой** , затем 4096 бит в секторе и, наконец, контрольная сумма, или **ECC** ( **код исправления ошибок** ). Преамбула пишется, когда диск for- матовый и содержит номер цилиндра и сектора, размер сектора и т.п. данные, также как синхронизация Информация.

Задача контроллера состоит в том, чтобы преобразовать последовательный поток битов в блок байтов и выполнить любую необходимую коррекцию ошибок. Блок байтов обычно является первой сборкой. по крупицам помещается в буфер внутри контроллера. После проверки контрольной суммы Fied и блок был объявлен свободным от ошибок, его можно затем скопировать в main объем памяти.

Контроллер для ЖК-монитора также работает как последовательное устройство на одинаково низкий уровень. Он считывает байты, содержащие символы, которые должны отображаться из объем памяти и генерирует в сигналы к изменить в поляризация из в подсветка за соответствующие пиксели, чтобы записать их на экране. Если бы не контроллер дисплея, программист операционной системы должен был бы явно про- вычислить электрические поля всех пикселей. С помощью контроллера запускается операционная система. задает контроллер несколькими параметрами, такими как количество символов или пикселей на строку и количество строк на экран, и позволяет контроллеру позаботиться об фактически управлять электрическими полями.

За очень короткое время ЖК-экраны полностью заменили старые **ЭЛТ -экраны.** ( **электронно-лучевая трубка** ) мониторы. ЭЛТ-мониторы направляют пучок электронов на флуоресцентную лампу. оресцирующий экран. Используя магнитные поля, система способна изгибать луч и рисовать пиксели на экране. По сравнению с ЖК-экранами ЭЛТ-мониторы были громоздкими, жадный до власти и хрупкий. Более того, разрешение современных (Retina) ЖК-дисплеев экраны является так хороший который в человек глаз является не в состоянии к выделить физическое лицо пикселей. Это Сегодня трудно представить, что в прошлом ноутбуки поставлялись с небольшим ЭЛТ-экраном, сделанный их больше, чем 20 см глубиной с хороший рабочий вес из около 12 килограммы.

## [Отображение памяти ввод/вывод](#_bookmark6)

Каждый контроллер имеет несколько регистров, которые используются для связи с ЦПУ. Записывая в эти регистры, операционная система может управлять де- способ доставки данных, приема данных, включения или выключения себя или иного выполнения некоторых действие. Читая эти регистры, операционная система может узнать, что устройства государство является, будь то Это является готовый к принимать а новый команда, и так на.

В добавление к в контроль регистры, многие устройства имеют а данные буфер который в оп- erating система может читать и писать. Например, обычный способ для компьютеров отображать пиксели на экране, чтобы иметь видеопамять, которая в основном просто данные буфер, доступный за программы или в действующий система к написать в.

Таким образом, возникает вопрос о том, как ЦП взаимодействует с управляющими регистрами. и также с в устройство данные буферы. Два альтернативы существует. В в первый подход,

каждому управляющему регистру назначается номер **порта ввода-вывода** , 8- или 16-битное целое число. набор всех портов ввода/вывода образует **пространство портов ввода/вывода** , которое защищено таким образом, что обычные пользователь программы не могу доступ Это (Только в действующий система может). С использованием а особенный ввод/вывод инструкция такая в качестве

В РЕГ, ПОРТ,

в ЦПУ может читать в контроль регистр ПОРТ и хранить в результат в ЦПУ регистр

РЕГ . Сходным образом, с использованием

ВНЕ ПОРТ, РЕГ.

CPU может записать содержимое REG в управляющий регистр. Самые ранние компьютеры, включая почти все мейнфреймы, такие как IBM 360 и все его преемники, работал Сюда.

В этой схеме адресные пространства для памяти и ввода-вывода различны, как показано на рисунке. в Инжир. 5-2 (а). Инструкции

В R0,4

и

МОВ R0,4

в этом дизайне совсем другие. Первый считывает содержимое порта ввода-вывода 4. и помещает его в R0 , тогда как последний считывает содержимое слова памяти 4 и помещает его в R0 . 4 с в эти примеры ссылаться к разные и несвязанный адрес пространства.

Два адрес Одно адресное пространствоДва адрес пространства

0xFFFF…



объем памяти

ввод/вывод порты

0



(а)

**Фигура 5-2.** (а) Отдельный ввод/вывод и объем памяти Космос. (б) Отображение памяти ввод/вывод.

Гибридный.

Второй подход, представленный в PDP-11, состоит в отображении всех элементов управления. регистрируется в пространстве памяти, как показано на рис. 5-2(b). Каждый управляющий регистр назначается уникальный адрес памяти, которому не назначена память. Эта система называется **вводом-выводом с отображением памяти** . В большинстве систем присвоенные адреса находятся в или возле в Топ из в адрес Космос. А гибридный схема, с отображаемый в памяти ввод/вывод буферы данных и отдельный ввод/вывод порты для контроль регистры, есть показано на рис. 5-2(с).

x86 использует это архитектура, с адреса 640К к 1М  1 существование зарезервированный за устройство данные буферы в IBM ПК совместимые, в добавление к порты ввода/вывода 0 к 64К  1.

Как эти схемы работают на практике? Во всех случаях, когда процессор хочет прочитать слово либо из памяти, либо из порта ввода/вывода, он помещает его адрес потребности на адресных линиях шины, а затем выставляет сигнал READ на управление шиной. линия. Вторая сигнальная линия используется, чтобы сообщить, занято ли пространство ввода/вывода или пространство памяти. нужный. Если это пространство памяти, память отвечает на запрос. Если это ввод-вывод пробел, устройство ввода-вывода отвечает на запрос. Если есть только место в памяти [как в Рис. 5-2(b)], каждый модуль памяти и каждое устройство ввода/вывода сравнивают адрес строки в диапазон адресов, которые он обслуживает. Если адрес попадает в его диапазон, он повторно отвечает к в запрос. С нет адрес является Когда-либо назначенный к оба объем памяти и ан ввод/вывод устройство, нет никакой двусмысленности и никакого конфликта.

Эти две схемы адресации контроллеров имеют разную силу и недостатки. Начнем с преимуществ ввода-вывода с отображением памяти. Прежде всего, если специальные инструкции ввода-вывода необходимы для чтения и записи регистров управления устройством, доступ к ним требует использования ассемблерного кода, так как нет возможности выполнить инструкция IN или OUT в C или C++. Вызов такой процедуры увеличивает нагрузку на управление вводом/выводом. Напротив, при вводе-выводе с отображением памяти регистры управления устройством просто переменные в памяти и могут быть адресованы в C так же, как и любые другие переменные. iables. Таким образом, при вводе-выводе с отображением в память драйвер устройства ввода-вывода может быть написан целиком. в С. Без отображаемого в память ввода-вывода, некоторые ассемблерный код нужно.

Во-вторых, при вводе-выводе с отображением памяти не требуется никакого специального механизма защиты. чтобы пользовательские процессы не выполняли ввод-вывод. Все, что должна сделать операционная система, это воздержаться от размещения той части адресного пространства, которая содержит управляющую регистра- в любом виртуальном адресном пространстве пользователя. А еще лучше, если у каждого устройства будет свое управление регистрируется на другой странице адресного пространства, операционная система может дать пользовательский контроль над определенными устройствами, но не над другими, путем простого включения желаемого страницы в своей таблице страниц. Такая схема позволяет использовать различные драйверы устройств. размещены в разных адресных пространствах, не только уменьшая размер ядра, но и сохраняя один водитель не мешает другие.

В-третьих, при вводе-выводе с отображением памяти каждая инструкция, которая может обращаться к памяти, также может ссылаться на управляющие регистры. Например, если есть инструкция TEST , который проверяет слово памяти на 0, его также можно использовать для проверки управляющего регистра на 0, что может быть сигналом того, что устройство находится в режиме ожидания и может принять новую команду. код на ассемблере может выглядеть так:

ПЕТЛЯ: ТЕСТ ПОРТ 4 // проверить, если порт 4 равно 0

БЫТЬ Q ГОТОВ // если 0, переходим в состояние готовности

ВЕТКА ЦИКЛ // в противном случае, Продолжить тестирование

ГОТОВЫ:

Если отображаемый в памяти ввод/вывод является нет настоящее время, в контроль регистр должен первый быть читать в в ЦПУ, затем проверено, требующий два инструкции вместо из точный один. В в дело из

цикла, приведенного выше, необходимо добавить четвертую инструкцию, немного замедляющую в быстрота обнаружения бездействующее устройство.

В компьютерном дизайне практически все связано с компромиссами, и это тут тоже дело. Ввод-вывод с отображением памяти также имеет свои недостатки. Во-первых, наиболее ком- путеры в настоящее время имеют некоторую форму кэширования слов памяти. Кэширование устройства контрольный регистр был бы катастрофическим. Рассмотрим цикл ассемблерного кода, приведенный выше. при наличии кеширования. Первое упоминание о ПОРТ 4 приведет к тому, что кэшировано. Последующие ссылки будут просто брать значение из кеша, а не даже спросить устройство. Затем, когда устройство, наконец, было готово, программное обеспечение было бы имеют нет способ из найти вне. Вместо, в петля было бы идти на навсегда.

К предотвращать это ситуация с отображаемый в памяти ввод/вывод, в аппаратное обеспечение имеет к быть способный для выборочного отключения кэширования, например, для каждой страницы. Эта функция добавляет дополнительную сложность как для аппаратного обеспечения, так и для операционной системы, которая должна управлять возраст выборочное кэширование.

Второй, если там является Только один адрес Космос, затем все объем памяти модули и все ввод/вывод устройства должен исследовать все объем памяти использованная литература к видеть который те к отвечать к. Если компьютер имеет одну шину, как на рис. 5.3(а), то каждый смотрит на каждую адрес прост.

ЦП читает и записывает память идти над это высокая пропускная способность автобус



I/O

Memory

CPU

CPU

Memory

I/O

Все адреса (память и ввод/вывод) иди сюда



Автобус

Этот порт памяти чтобы разрешить устройства ввода/вывода доступ к объем памяти

(б)

**Фигура 5-3.** (а) А одноместный автобус архитектура. (б) А двойная шина объем памяти архитектура.

Тем не менее, в современных персональных компьютерах наблюдается тенденция иметь выделенный высокопроизводительный компьютер. скорость шины памяти, как показано на рис. 5-3(b). Шина адаптирована для оптимизации памяти. высокая производительность без компромиссов в пользу медленных устройств ввода-вывода. x86 систем- темы может имеют несколько автобусов (объем памяти, PCIe, SCSI, и USB), в качестве показано в Инжир. 1-12.

Проблема с отдельной шиной памяти на машинах с отображением памяти заключается в том, что устройства ввода-вывода не имеют возможности видеть адреса памяти, когда они шина памяти, поэтому они никак не могут на них реагировать. Опять же, специальные меры уре имеют к быть взятый к сделать отображаемый в памяти ввод/вывод работай на а система с несколько

автобусов. Одна из возможностей состоит в том, чтобы сначала отправить все ссылки на память в память. Если память не отвечает, тогда ЦП пробует другие шины. Этот дизайн может быть сделанный к работать, но требует дополнительное оборудование сложность.

А второй возможное дизайн является к помещать а вынюхивание устройство на в объем памяти автобус к передать все адреса, представленные потенциально заинтересованным устройствам ввода/вывода. Проблема здесь является который ввод/вывод устройства май нет быть способный к процесс Запросы в в скорость в объем памяти может.

Третий возможный дизайн, который вполне соответствовал бы дизайну, набросанному в Рис. 1-12 предназначен для фильтрации адресов в контроллере памяти. В таком случае память Микросхема контроллера содержит регистры диапазона, которые предварительно загружаются во время загрузки. Для экс- обильный, 640К к 1М  1 мог быть отмечен в качестве а не память диапазон. Адреса который попадают в один из диапазонов, помеченных как непамятные, перенаправляются на устройства в вместо памяти. Недостатком этой схемы является необходимость выяснения время загрузки, адреса памяти которых на самом деле не являются адресами памяти. Таким образом, каждый схема имеет аргументы за и против Это, так компромиссы и компромиссы находятся неизбежный.

## [Прямой объем памяти Доступ](#_bookmark6)

Независимо от того, имеет ЦП ввод-вывод с отображением памяти или нет, ему необходимо обращаться к контроллерам устройств для обмена данными с ними. ЦП может запросить данные от контроллера ввода-вывода по одному байту за раз, но при этом тратится время процессора, поэтому часто используется другая схема, называемая **DMA** ( **прямой доступ к памяти ).** К упростим объяснение, будем считать, что ЦП обращается ко всем устройствам и памяти через единую системную шину, которая соединяет ЦП, память и устройства ввода-вывода, как показано в Инжир. 5-4. Мы уже знать который в настоящий организация в современное системы является более сложный, но все в принципы находятся в одно и тоже. действующий система может используйте только DMA, если аппаратное обеспечение имеет контроллер DMA, что есть в большинстве систем. Иногда этот контроллер интегрируется в дисковые контроллеры и другие контроллеры, но такая конструкция требует отдельного контроллера прямого доступа к памяти для каждого устройства. Больше ком- Только один контроллер прямого доступа к памяти доступен (например, на материнской плате) для регулирования. инг переводы в несколько устройств, часто одновременно.

Независимо от того, где он физически расположен, контроллер прямого доступа к памяти имеет доступ к системная шина, независимая от ЦП, как показано на рис. 5-4. Он содержит несколько рег- isters, которые могут быть записаны и прочитаны процессором. К ним относятся адрес памяти регистр, регистр счетчика байтов и один или несколько управляющих регистров. Контрольный регистр- указывают используемый порт ввода-вывода, направление передачи (чтение из порта ввода-вывода). устройство или запись на устройство ввода/вывода), единица передачи (байт за раз или слово за время), и количество байтов для передачи в одном пакете.

Чтобы объяснить, как работает DMA, давайте сначала посмотрим, как происходит чтение с диска, когда ДМА не используется. Сначала контроллер диска считывает блок (один или несколько секторов) с накопителя последовательно, бит за битом, пока весь блок не окажется во внутренней памяти контроллера. буфер. Следующий, Это вычисляет в контрольная сумма к проверять который нет читать ошибки имеют произошел.



Drive

CPU

1. CPU programs the DMA controller

DMA

controller

Disk controller

Main memory

Buffer

4. Ack

Interrupt when done

2. DMA requests transfer to memory

3. Data transferred

Bus

Control

Count

Address

**Фигура 5-4.** Операция из а прямой доступ к памяти перевод.

Затем в контроллер причины ан прерывать. Когда в действующий система начинается Бег, он может считывать блок диска из буфера контроллера по байту или по слову за раз, выполнение цикла, при котором на каждой итерации считывается один байт или слово из детерминированного контроллера. порок зарегистрироваться и хранение это внутри основная память.

При использовании DMA процедура отличается. Сначала ЦП программирует прямой доступ к памяти контроллер от параметр это регистры так Это знает Какие к перевод куда (шаг 1 на рис. 5-4). Он также выдает команду контроллеру диска, говорящую ему прочитать данные. с диска в его внутренний буфер и проверить контрольную сумму. Когда достоверные данные в в дисковый контроллер буфер, прямой доступ к памяти может начинать.

Контроллер DMA инициирует передачу, отправляя запрос на чтение по шине. к контроллеру диска (шаг 2). Этот запрос на чтение выглядит как любой другой запрос на чтение, и контроллер диска не знает (или не заботится), исходит ли он от ЦП или от DMA-контроллера. Как правило, адрес памяти для записи находится на шине. адресные строки, поэтому, когда контроллер диска извлекает следующее слово из своего внутреннего буфер, он знает, куда его записать. Запись в память — еще одна стандартная шина. цикл (шаг 3). Когда запись завершена, контроллер диска отправляет подтверждение. сигнал фронта контроллеру прямого доступа к памяти, также по шине (шаг 4). DMA con- троллер затем приращения в объем памяти адрес к использовать и декременты в байт считать. Если в байт считать является все еще больший чем 0, шаги 2 через 4 находятся повторный до того как в считать достигает 0. В который время, в прямой доступ к памяти контроллер прерывает в ЦПУ к позволять Это знать, что передача теперь завершена. При запуске операционной системы она делает не обязан скопировать диск блокировать в память; это уже есть.

Контроллеры прямого доступа к памяти значительно различаются по своей сложности. Самые простые обрабатывать по одной передаче за раз, как описано выше. Более сложные могут быть про- настроен для одновременной обработки нескольких переводов. Такие контроллеры имеют несколько Типовые наборы внутренних регистров, по одному на каждый канал. ЦП начинает с загрузки каждый поставил из регистры с в соответствующий параметры за это перевод. Каждый перевод должен

используйте другой контроллер устройства. После переноса каждого слова (шаги 2–4) на рис. 5-4 контроллер прямого доступа к памяти решает, какое устройство обслуживать следующим. Он может быть установлен до использования циклического алгоритма, или он может иметь схему приоритетов в пользу одни устройства над другими. Несколько запросов к разным контроллерам устройств могут быть в то же время, при условии, что есть недвусмысленный способ сказать знания врозь. Часто используется другая линия подтверждения на шине. за каждый прямой доступ к памяти канал по этой причине.

Многие шины могут работать в двух режимах: пословном режиме и блочном режиме. Некоторые контроллеры прямого доступа к памяти также могут работать в любом из этих режимов. В прежнем режиме операция аналогична описанной выше: контроллер DMA запрашивает передачу одного слово и получает его. Если процессору также нужна шина, он должен ждать. Механизм называется **кражей циклов** , потому что контроллер устройства проникает и крадет событие. Национальная шина время от времени переключается с центрального процессора, слегка задерживая его. В блочном режиме, контроллер прямого доступа к памяти сообщает устройству, что нужно захватить шину, выполнить серию передач, затем отпустите автобус. Эта форма работы называется **пакетным режимом** . Это эффективнее- эффективнее, чем кража циклов, потому что получение автобуса требует времени и нескольких слов может быть переведен за в цена из один автобус приобретение. вниз сторона к лопаться заключается в том, что он может блокировать ЦП и другие устройства на значительный период времени, если длинный пакет передается.

В модели, которую мы обсуждали, иногда называемой **режимом пролета** , Контроллер прямого доступа к памяти указывает контроллеру устройства передавать данные непосредственно в основной объем памяти. Альтернативный режим, который используют некоторые контроллеры прямого доступа к памяти, состоит в том, чтобы устройство контроллер отправляет слово контроллеру прямого доступа к памяти, который затем выдает второй повторный запрос на шину. стремление написать слово туда, куда оно должно идти. Эта схема требует дополнительный цикл шины на каждое переданное слово, но является более гибким в том смысле, что он также может выполнять устройство к устройству копии и четное память в память копии (от первый выпуск а читать на память а затем выдача написать к память в другой адрес).

Большинство прямой доступ к памяти контроллеры использовать физический объем памяти адреса за их переводы. Использование физических адресов требует от операционной системы преобразования виртуального адреса. одеть предполагаемый буфер памяти в физический адрес и записать этот физический адрес в адресный регистр контроллера прямого доступа к памяти. Альтернативная схема, используемая в несколько контроллеров DMA должны записывать виртуальные адреса в контроллер DMA в вместо. Затем контроллер DMA должен использовать MMU, чтобы иметь связь между виртуальными и физическими данными. перевод сделан. Только в том случае, если MMU является частью памяти (возможно, но редко), скорее чем часть из ЦП, может виртуальные адреса быть помещенным на автобус.

Ранее мы упоминали, что диск сначала считывает данные в свой внутренний буфер, прежде чем ДМА может начаться. Вы можете удивиться, почему контроллер не просто хранит байт в оперативной памяти, как только он получает их с диска. Другими словами, почему нужен внутренний буфер? Есть две причины. Во-первых, выполняя внутренние буферизации, контроллер диска может проверить контрольную сумму перед началом передачи. Если в контрольная сумма неверна, ошибка сигнализируется и нет передача сделана.

Вторая причина заключается в том, что после начала передачи данных биты продолжают поступать. из в диск в а постоянный показатель, будь то в контроллер является готовы за их или нет. Если

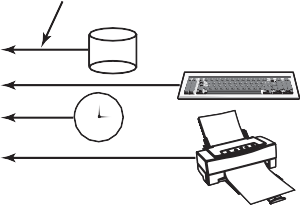
контроллер пытался записать данные напрямую в память, ему пришлось бы пройти через системной шине для каждого передаваемого слова. Если автобус был занят по какой-то другой причине, В случае его использования (например, в пакетном режиме) контроллеру придется ждать. Если следующий слово диска прибыло до того, как предыдущее было сохранено, контроллер надо где-то хранить. Если бы шина была очень загружена, контроллер мог бы хранение довольно а немного слова и имея а много из администрация к делать в качестве хорошо. Когда блок буферизуется внутри, шина не нужна до тех пор, пока не начнется DMA, поэтому конструкция контроллера намного проще, потому что передача DMA в память не критично по времени. (Некоторые старые контроллеры действительно обращались непосредственно к памяти с только небольшое количество внутренней буферизации, но когда шина была очень загружена, транс- Фер может быть имели к быть прекращенным с перерасход ошибка.)

Не все компьютеры используют DMA. Аргумент против этого заключается в том, что основной процессор часто намного быстрее, чем контроллер прямого доступа к памяти, и может выполнять работу намного быстрее (когда ограничивающим фактором является не скорость устройства ввода-вывода). Если нет другой работы для него делать, заставлять (быстрый) ЦП ждать завершения (медленного) контроллера DMA бессмысленно. Кроме того, избавившись от контроллера DMA и передав всю работу процессору, програмное обеспечение экономит Деньги, важный на бюджетный (встроенный) компьютеры.

## [прерывания Пересмотрено](#_bookmark6)

Мы кратко представили прерывания в разд. 1.3.4, но есть еще что сказать. В типичный личный компьютер система, в прерывать состав является в качестве показано в Инжир. 5-5. На аппаратном уровне прерывания работают следующим образом. Когда устройство ввода-вывода закончило возложенную на него работу, она вызывает прерывание (при условии, что прерывания были разрешено операционной системой). Он делает это путем подачи сигнала на линию шины. что оно назначено. Этот сигнал обнаруживается микросхемой контроллера прерываний на в родительская плата, которая затем решает, что делать.

весло



CPU

3. CPU acks interrupt

Interrupt controller

1. Device is finished

Disk

Keyb

11 12 1

10 2

9 3

8 4

Clock

2. Controller issues interrupt

7 5

6

Printer

Автобус

**Фигура 5-5.** Как ан прерывать бывает. связи между в устройства и в контроллер на самом деле использовать прерывать линии на автобус скорее чем посвященный провода.

Если никаких других прерываний не ожидается, контроллер прерываний обрабатывает прерывание. немедленно. Однако, если выполняется другое прерывание или другое устройство сделанный а одновременный запрос на а более приоритетный прерывать запрос линия на в автобус,

устройство просто игнорируется на данный момент. В этом случае он продолжает утверждать не- прерывать сигнал на автобус пока это обслуживается от ЦП.

Для обработки прерывания контроллер помещает номер в адресные строки, указанные летать который устройство хочет внимание и утверждает а сигнал к прерывать в ЦПУ.

Сигнал прерывания заставляет ЦП прекратить то, что он делает, и начать делать что-то другое. Число в адресных строках используется в качестве индекса в таблице. вызывается **вектором прерывания** для получения нового счетчика команд. Этот программный счетчик указывает на начало соответствующей процедуры обслуживания прерывания. Обычно ловушки и с этого момента прерывания используют один и тот же механизм, часто используя одни и те же вектор прерывания. Местоположение вектора прерывания может быть жестко запрограммировано в системе. китайском или может быть где угодно в памяти, с регистром ЦП (загружаемым операционной системой). инг система) указывая на его происхождение.

Вскоре после Это начинается Бег, в служба прерывания процедура признает прерывание путем записи определенного значения в один из портов ввода-вывода контроллера прерываний. Это подтверждение сообщает контроллеру, что он может выдать другое прерывание. Заставляя ЦП задерживать это подтверждение до тех пор, пока он не будет готов к обработке следующего прерывание, условия гонки, включающие несколько (почти одновременных) прерываний, могут избегать. Кроме того, некоторые (старые) компьютеры не имеют централизованного межсетевого интерфейса. разрыв контроллер, так каждое устройство контроллер просит его своя прерывает.

Аппаратное обеспечение всегда сохраняет определенную информацию перед запуском сервиса. процедура. Какая информация сохраняется и где она сохраняется, сильно зависит от процессора. к ЦП. Как минимум, счетчик программ должен быть сохранен, т.е. прерванный процесс можно перезапустить. С другой стороны, все видимые регистры и большой количество внутренние регистры могут быть сохранен как хорошо.

Одна проблема заключается в том, где сохранить эту информацию. Один из вариантов - поместить его во внутренний регистры, которые операционная система может считывать по мере необходимости. Проблема с этим приложением Проблема в том, что тогда контроллер прерываний не может быть подтвержден до тех пор, пока не будут соблюдены все потенциальные возможности. была считана важная информация, чтобы второе прерывание не перезаписало внутренние регистры, сохраняющие состояние. Эта стратегия приводит к длительному простою, когда прерывания отключены и, возможно, к потерянным прерываниям и потерянным данным.

Следовательно, большинство процессоров сохраняют информацию в стеке. Однако это приложение у proach тоже есть проблемы. Для начала: чей стек? Если используется текущий стек, это вполне может быть стек пользовательского процесса. Указатель стека может быть даже недопустимым, что вызовет фатальную ошибку, когда аппаратное обеспечение попытается написать несколько слов в объявлении. платье указал на. Кроме того, он может указывать на конец страницы. Через несколько воспоминаний пишет, граница страницы может быть превышена и сгенерирована ошибка страницы. Иметь ошибка страницы, возникающая во время обработки аппаратного прерывания, создает большую проблему: куда спасти государство обрабатывать ошибка страницы?

Если используется стек ядра, гораздо больше шансов, что указатель стека быть законным и указывать на закрепленную страницу. Однако переход в режим ядра май требовать изменение ММУ контексты и воля вероятно аннулировать наиболее или все из кэш и TLB. Перезагрузка всего этого, статического или динамического, увеличит в время обработки прерывания и, таким образом, потеря процессорное время.

#### Точный и Неточный прерывания

Другая проблема связана с тем, что большинство современных процессоров сильно загружены. конвейерные и часто суперскалярные (внутренне параллельные). В старых системах после каждого выполнение инструкции было завершено, микропрограмма или аппаратное обеспечение проверили, ожидалось прерывание. Если это так, счетчик программ и PSW были сброшены. в стек, и начинается последовательность прерываний. После запуска обработчика прерывания задний ход процесс взял место, с в старый PSW и программа прилавок выскочил из в куча и предыдущий процесс продолжился.

Эта модель делает неявное предположение, что если прерывание происходит сразу после какая-то инструкция, все инструкции до этой инструкции включительно были выполняется полностью, и никакие инструкции после него не выполняются вообще. На старых ма- китайцы, это предположение было всегда в силе. На современные Это может нет быть.

Для начала рассмотрим модель конвейера на рис. 1-7(а). Что произойдет, если во- прерывание происходит, когда конвейер заполнен (обычный случай)? Многие инструкции находятся в различные этапы исполнения. При возникновении прерывания значение программы счетчик может не отражать правильную границу между выполняемыми инструкциями и невыполненные инструкции. На самом деле, многие инструкции могли быть частично выполняются, причем разные инструкции более или менее полны. В этой ситуации, программный счетчик, скорее всего, отражает адрес следующей инструкции. извлекается и помещается в конвейер, а не адрес инструкции, которая точный был обрабатывается исполнительным блоком.

На суперскалярной машине, такой как на рис. 1.7(b), дела обстоят еще хуже. Инструкции можно разбить на микрооперации и микрооперации. может выполняться не по порядку, в зависимости от наличия внутренних ресурсов, таких как как функциональные блоки и регистры. Во время прерывания некоторые инструкции начатые давно, возможно, не начались, а другие, начавшиеся совсем недавно, могут быть сделано больше всего. В момент, когда сигнализируется прерывание, может быть много инструкций. в разной степени завершенности, с меньшей связью между ними и программа прилавок.

Прерывание, которое оставляет машину в четко определенном состоянии, называется **точным прерывать** (Уокер и Крагон, 1995). Такой ан прерывать имеет четыре характеристики:

* + - 1. ПК (Программа Прилавок) является сохранен в а известный место.
      2. Все инструкции до в один указал к от в ПК имеют завершенный.
      3. Нет инструкция вне в один указал к от в ПК имеет законченный.
      4. исполнение государство из в инструкция указал к от в ПК является известный.

Обратите внимание, что нет запрета на инструкции, кроме тех, на которые указывает ПК от запуска. это точный который любые изменения, которые они вносят регистрировать или память должно быть отменено до того, как произойдет прерывание. Допускается, чтобы инструкция указал к имеет был казнен. Это является также разрешенный который Это имеет нет был казнен.

Однако, Это должен быть Чисто который дело применяется. Часто, если в прерывать является ан ввод/вывод меж- rupt, выполнение инструкции еще не началось. Однако, если прерывание действительно trap или page fault, то ПК обычно указывает на инструкцию, вызвавшую ошибка, чтобы его можно было перезапустить позже. Ситуация на рис. 5-6(а) иллюстрирует точную ин- прерывать. Все инструкции до счетчика программ (316) выполнены, и ни одна из те вне Это имеют началось (или имеют был свернутый назад к отменить их последствия).

ПК

(а)

332

328

Not executed 10% executed

40% executed

35% executed

20% executed

60% executed

80% executed

Fully executed

|  |
| --- |
| Not executed |
| Not executed |
| Not executed |
| Not executed |
| Fully executed |
| Fully executed |
| Fully executed |
| Fully executed |

324

320

316 ПК 312

308

304

300

332

328

324

320

316

312

308

304

300

**Фигура 5-6.** а) А точное прерывание. (б) Ан неточный прерывать.

Прерывание, не отвечающее этим требованиям, называется **неточным прерыванием . сбой** и делает жизнь разработчика операционной системы крайне неприятной, должен выяснить, что произошло и что еще должно произойти. Рис. 5-6(b) иллюстр- обрабатывает неточное прерывание, когда рядом с программным счетчиком находятся на разных стадиях завершения, причем более старые не обязательно более сложные. полнее, чем младшие. Машины с неточными прерываниями обычно выбрасывают большое количество внутреннего состояния в стеке, чтобы дать операционной системе возможность выяснить, что происходит. Код, необходимый для перезагрузки машины, как правило, чрезвычайно сложным. Кроме того, сохранение большого количества информации в память на каждое прерывание делает прерывания медленными, а восстановление еще хуже. Этот приводит к иронической ситуации, когда очень быстрые суперскалярные процессоры иногда неподходящий для работы в режиме реального времени за счет медленный прерывает.

Некоторые компьютеры сконструированы таким образом, что некоторые виды прерываний и ловушек точны, а другие нет. Например, прерывания ввода-вывода должны быть точными, но ловушками. в связи к фатальный программирование ошибки быть неточный является нет так плохой поскольку нет пытаться необходимость быть сделан для перезапуска запущенного процесса после его деления на ноль. Некоторые машины есть бит, который можно установить, чтобы все прерывания были точными. Минус набора- Этот бит заключается в том, что он заставляет ЦП тщательно регистрировать все, что он делает, и поддерживать теневые копии регистров, чтобы он мог генерировать точное прерывание в любой момент. стант. Все это накладные расходы имеет майор влияние на производительность.

Некоторые суперскалярные машины, такие как семейство x86, имеют точные прерывания для позволяют старому программному обеспечению работать корректно. Цена, заплаченная за обратную совместимость с точный прерывает является очень сильно сложный прерывать логика в пределах в ЦПУ к убедитесь, что когда контроллер прерываний сигнализирует о том, что он хочет разрыв, все инструкции вверх к некоторые точка находятся допустимый к финиш и никто вне который

точка может иметь какое-либо заметное влияние на состояние машины. Здесь цена оплачивается не временем, а площадью фишки и сложностью дизайна. Если точно во- прерывания не требовались в целях обратной совместимости, эта область чипа будет доступен для больших встроенных кэшей, что сделает ЦП быстрее. С другой С другой стороны, неточные прерывания значительно усложняют операционную систему и помедленнее, так Это является жесткий к скажи, какой подход действительно лучше.