**Структура и основные компоненты вычислительной системы**

Работая на машине, мы взаимодействуем не с аппаратной реализацией, а с программным обеспечением.

Вычислительная система - объединение программных и аппаратных средств, которые предоставляют услуги пользователю.

Структура организации вычислительной системы (ВС):

|  |
| --- |
| Прикладные программы |
| Системы программирования | | |
| Управление логическими устройствами | | | | |
| Управление физическими устройствами | | | | | | |
| Аппаратные средства | | | | | | | | |

**Аппаратные средства.**

Ресурсы ВС разделяются на два типа:

* не участвующие в управлении программой (объем винчестера и т.д.).
* участвующие в управлении программой (размер ячейки памяти, объем оперативной памяти, скорость выполнения команд).

Ресурсы второго типа называются физическими ресурсами аппаратуры.

**Управление физическими устройствами.**

Управление физическими устройствами осуществляют программы, ориентированные на аппаратуру, взаимодействующие с аппаратными структурами, знающие "язык" аппаратуры.

**Управление логическими устройствами.**

Этот уровень ориентирован на пользователя. Команды данного уровня не зависят от физических устройств, они обращены к предыдущему уровню. На базе этого уровня могут создаваться новые логические ресурсы.

**Системы программирования**.

Система программирования - это комплекс программ для поддержки всего технологического цикла разработки программного обеспечения.

**Прикладное программное обеспечение.**

Прикладное программное обеспечение необходимо для решения задач из конкретных областей.

Операционная система (ОС) - программа, обеспечивающая взаимодействие пользователя с ВС, а также управляющая ресурсами ВС (логическими и физическими). К ОС мы будем относить второй и третий уровень нашей пирамиды.

Структура ЭВМ:



Основной функцией центрального процессора (ЦП) является обработка информации и взаимодействие с устройствами. Обмениваться данными ЦП может только с ОЗУ (Оперативно Запоминающее Устройство).

В ОЗУ размещается выполняемая в данный момент программа. ОЗУ состоит из ячеек памяти. Каждая ячейка имеет свой уникальный адрес, и каждая разбита на два поля: поле внутрисистемной информации (которое, например, может содержать бит четности) и машинное слово, содержащее команду или данные. Машинное слово состоит из некоторого количества двоичных разрядов, которое определяет разрядность системы.

ЦП выбирает из ОЗУ последовательность команд для выполнения. ЦП состоит из двух компонентов:

Устройство Управления (УУ) принимает очередное слово из ОЗУ и разбирается - команда это или данные. Если это команда - то УУ выполняет ее, иначе передает АУ.

Арифметическое Устройство (АУ) занимается исключительно вычислениями.

УУ работает с регистровой памятью, время доступа к которой значительно быстрее, чем к ОЗУ, и которая используется специально для сглаживания дисбаланса в скорости обработки информации процессором и скорости доступа к ОЗУ.

Лекция №2

Мы определили, что вычислительная система (ВС) это некоторое объединение аппаратных средств, средств управления аппаратурой (физическими ресурсами), средств управления логическими ресурсами, системы программирования и прикладное программное обеспечение.

|  |
| --- |
| Прикладные программы |
| Системы программирования | | |
| Управление логическими устройствами | | | | |
| Управление физическими устройствами | | | | | | |
| Аппаратные средства | | | | | | | | |

Мы определили, что нижний уровень - это чисто аппаратура, это то, что делается из металла, пластика и прочих материалов, используемых для производства железа, или hardware компьютера.

Следующий уровень это программы, но программы, ориентированные на качество и свойства аппаратуры. Эти программы и разработчики этих программ досконально знают особенности управления каждого типа из аппаратных компонентов. Нижний уровень между физическим уровнем и аппаратурой - это интерфейс этого управления, это есть некоторые наборы команд управления физическими ресурсами, т.е. каждое устройство имеет свой язык или свой набор команд управления.

Следующий уровень - это уровень, который ориентирован на сглаживание аппаратных особенностей. Он целиком и полностью предназначен для создания более комфортных условий в работе пользователя. Если предположим мы работаем с устройством внешней памяти “жесткий магнитный диск”, то параметрами, которые характерны для конкретного диска, могут быть, предположим, сколько считывающих и записывающих головок имеет это устройство, сколько поверхностей, на которых находится хранящий информацию слой. И, соответственно, набор команд управления этого устройства ориентирован на эти параметры. И конечно, вам, как программистам, не интересно работать в терминах: считать бит со второй поверхности десятого цилиндра такой-то дорожки. Это тяжело и неинтересно. Этот уровень логических ресурсов создает некоторое обобщенное устройство одно на всю систему, и пользователь работает в терминах этого обобщенного устройства. А уже программы логического уровня разбираются к какой из программ управления физическими устройствами надо обратиться чтобы запрос пользователя к логическому устройству правильно оттранслировать к конкретному физическому устройству.

Мы говорили о том, что в разных текстах либо два уровня управления - логический и физический, либо три - логический, физический и система программирования, относят к операционной системе. Мы далее будем считать операционной системой два уровня - логический и физический. Мы начали рассматривать основные свойства этой иерархии, которую объявили и нарисовали достаточно простую и традициоонную схему или структуру вычислительной машины:



Это процессорный элемент, т.е. устройство, которое перерабатывает информацию, это оперативная память (Оперативное Запоминающее Устройство, ОЗУ), и устройства управления внешними устройствами (УУВУ). Мы определили основное качество оперативное памяти: именно в оперативной памяти лежит исполняемая в данный момент программа, и процессор все последующие команды исполняемой программы берет из оперативной памяти. Если чего-то не хватает, идет запрос к внешним устройству, информация подкачивается в оперативноую память, и опять-таки из оперативной памяти команды поступают в процессор на обработку.

В принципе внешнее устройство можно реализовать на оперативной памяти. Если вы знаете, есть такая замечательная программа, которая называется MS-DOS. Эта операционная система (хотя она классически не является операционной системой) имеет ограничения на размер используемой памяти 640Кб. А аппаратура реальных машин на сегодняшний день может иметь физическую оперативную память существенно больших размеров. В этой системе можно создавать логический диск, который размещается на оперативной памяти. Т.е. по всем интерфейсам работа с ним будет осуществляться как с жестким диском, но размещаться он будет на оперативной памяти. И здесь разница в том, что из тех 640Кб процессор берет команды на исполнение, а из оставшихся, которые мы объявили логическим диском, не берет, потому что он будет работать с ним как с обычным жестким диском или любым другим носителем.

Мы с вами начали более подробно говорить о процессоре и зафиксировали одну из основных проблем, которая имеет место быть в области вычислительной техники. Она формулируется так - это несоответствие в скоростях доступа и обработки информации, различных компонентов вычислительной системы. Почему - у каждого компонента есть своя предыстория. Где-то это просто реально медленное устройство, где-то влияет на эту скорость вина проводников, которые находятся между процессорным элементом и конкретным устройством. Для каждого случая причина своя. Но это есть проблема. То, что реальная оперативная память на порядки более медленна, чем скорость обработки информации в процессоре. И здесь возникает в общем сумасшедшая проблема: зачем нам повышать производительность процессора, если доступ к памяти (а мы все время что-то берем из памяти, так как работа процессора это обработка информации, которую он берет из памяти) если доступ к памяти у нас настолько замедлен. Т.е. гарантировано, если ничего не будет сделано конструктивно, то скорость всей системы будет равняться скорости работы компонента, имеющего наименьшую скорость.

**Регистры.** Мы начали смотреть, какие конструктивные решения есть в аппаратуре вычислительной системы, которые предназначены для сглаживания этого дисбаланса. И первое о чем мы начали говорить - это регистры. В процессоре имеются устройства, способные хранить некоторую информацию. К этим устройствам возможен доступ прямым или косвенным способом из программы, выполняемой на машине. При этом есть группа регистров, которые называются регистрами общего назначения, которые доступны из всех команд. Эти регистры могут обладать свойством обработки и хранения и обработки определенных типов данных - это могут быть вещественные данные, короткие целые данные, которые используются, предположим, для индексирования, это могут быть длинные целые данные. При этом скорость доступа к регистрам общего назначения соизмерима со скоростью обработки информации в процессоре. При умелом программировании, можно использовать регистры общего назначения в целях сокращения числа обращений к оперативной памяти. Это означает, что торможение на участке процессор-оперативная память сокращается. Рассмотрим другие группы регистров.

**Специальные регистры.** К этой группе относятся две подгруппы регистров.

**1.** Первая подгруппа - это регистры, отвечающие за состояние исполняемой программы. К этим регистрам относится счетчик команд. Этот регистр содержит адрес исполняемой в данный момент команды. Это тот самый регистр, который можно изменять только косвенно, передав управление куда-то. Второй регистр из этой же подгруппы - регистр результата (flags), содержащий результат выполнения последней команды. По значению этого регистра можно организовывать те или иные действия. К этой подгруппе относится также регистр указателя стека. Есть команды, которые работают со стеком. Эти команды обычно используются для программирования переходов из и в функцию. Стек в системе используется для передачи параметров и организации автоматической памяти. Это память, которая занимается относительно вершины стека при входе в функцию, и, при выходе, она освобождается. Поэтому в автоматических переменных нельзя хранить данные после выхода из функции.

**2.** Вторая подгруппа регистров - это регистры управления компонентами вычислительной системы, или управляющие регистры. Практически в любой вычислительной системе имеются регистры, предназначенные для организации взаимосвязи процессора с внешним миром. Эти регистры связываются с УУВУ и через эти регистры процессор может организовывать управление внешними устройствами. Например, если возьмем регистр управления жесткого диска, то у него могут быть следующие поля:

Поле, указывающее кому предназначена информация на этом регистре в данный момент времени (процессору или диску).

Если эта команда находится в формате от процессора к устройству, в нем может находиться код операции управления устройством, могут находиться некоторые операнды и т.д. Устройство пытается выполнить эту команду, и по результату ее выполнения возвращается результат тоже в управляющий регистр (это может быть информация о том, что обмен закончен успешно, или обмен незакончен и причина этого).

**Система прерываний.** К средствам, управляющим взаимосвязью с внешними устройствами, можно отнести систему прерываний. В каждой вычислительной машине имеется предопределенный, заданный при разработке и производстве, набор некоторых событий и аппаратных реакций на возникновение каждого из этих событий. Эти события называются прерываниями. Аппарат прерываний используется для управления внешними устройствами и для получения возможности асинхронной работы с внешними устройствами. Синхронная работа, это когда система говорит: “Дай мне блок информации”, и затем стоит и ждет этого блока. Асинхронная - это когда система говорит: “Принеси мне, пожалуйста, блок информации”, и продолжает свою работу, а когда приходит блок, она прерывается (по прерыванию завершения обмена) и принимает информацию. Такова схема прерываний. Одним из прерываний, которые есть в системе, является прерывание по завершению обмена.

В момент возникновения прерывания, действия в аппаратуре ВС следующие:

В некоторые специальные регистры аппаратно заносится (сохраняется) информация о выполняемой в данный момент программе. Это минимальная информация, необходимая для начала обработки прерывания. Обычно, в этот набор данных входит счетчик команд, регистр результата, указатель стека и несколько регистров общего назначения. (Эти действия называются малым упрятыванием).

В некоторый специальный управляющий регистр, условно будем его называть регистром прерываний, помещается код возникшего прерывания.

Запускается программа обработки прерываний операционной системы.

Запущенная программа в начале потребляет столько ресурсов (не более), сколько освобождено при аппаратном упрятывании информации. Эта программа производит анализ причины прерывания. Если это прерывание было фатальным (деление на ноль, например), то продолжать выполнение программы бессмысленно и управление передается части операционной системы, которая эту программу выкинет. Если это прерывание не фатальное, происходит дополнительный анализ, который приводит к ответу на вопрос: можно ли оперативно обработать прерывание? Пример прерывания, которое всегда можно обработать оперативно - прерывание по таймеру. Например, прерывание, связанное с приходом информации по линии связи нельзя обработать оперативно, т.к. происходит расчищение в системе места для программы операционной системы, которая займется обработкой этого прерывания. Происходит т.н. полное упрятывание. Теперь прячется не только информация о некоторых регистрах исполнявшейся программы, теперь все регистры сохраняются в таблицах системы (а не в аппаратных регистрах, как раньше) и фиксируется то, что пространство оперативной памяти, занимаемое программой, может быть перенесено (при необходимости) на внешнее устройство. Дальше идет обработка прерывания и возврат из прерывания.

Здесь надо отметить одно важное свойство: прерывания могут быть инициированы схемами контроля процессора (например, при делении на ноль), могут быть инициированы внешним устройством (при нажатии клавиши на клавиатуре возникает прерывание, по которому процессор считывает из некоторого регистра нажатый символ).

Возвращаемся к нашей основной проблеме. ВЗУ на многие порядки более медленно, чем оперативная память, т.е. возникает проблема торможения ВС. Если бы все обмены с внешними устройствами происходили в синхронном режиме, то производительность ВС была бы очень низкой. Одной из причин появления аппарата прерываний было сглаживание скоростей доступа к внешним устройствам для процессора и оперативной памяти (оперативная память здесь выступает как более высокоскоростное устройство). То, что, используя аппарат прерываний можно было работать с внешними устройствами в асинхронном режиме, т.е. задать заказ на обмен и забыть о нем до прерывания завершения обмена, позволило в целом увеличить производительность ВС.

**Регистры буферной памяти (Cache, КЭШ).** Следующая группа регистров - регистры, относящиеся к т.н. буферной памяти. Мы возвращаемся к проблеме взаимодействия процессора и оперативной памяти и сглаживанию скоростей доступа в оперативную память.

Предположим, у нас есть некоторая программа, которая производит вычисление некоторого выражения, при этом, процесс вычисления этого выражения будет представим следующим образом. В какие-то моменты идут обращения за операндами в оперативную память, в какие-то моменты обработанные данные записываются в оперативную память. Есть один из нескольких путей, которые сглаживают несоответствие скоростей процессора и оперативной памяти, который заключается в сокращении реальных обращений к оперативной памяти. Процессоры содержат быстродействующую регистровую память, призванную буферизовать обращения к оперативной памяти.

Алгоритм чтения из оперативной памяти следующий:

проверяется наличие в специальном регистровом буфере строчки, в которой находится исполнительный адрес, совпадающий с исполнительным адресом требуемого операнда. Если такая строчка имеется, то соответствующее этому адресу значение, считается значением операнда и передается в процессор для обработки (т.е. обращение в оперативную память не происходит).

Если такой строчки нет, то происходит обмен с оперативной памятью, и копия полученного значения помещается в регистровый буфер и помечается исполнительным адресом этого значения в оперативной памяти. Содержимое операнда поступает в процессор для обработки. При этом решается проблема размещения новой строчки. Аппаратно ищется свободная строка (но она может быть только в начале работы машины), и если таковая не найдена, запускается аппаратный процесс вытеснения из этого буфера наиболее “старой” строчки. Старость определяется по некоторому предопределенному критерию. Например, признаком старения может быть количество обращений к этому буферу при котором нет обращений к этой строчки. В каждом таком случае число в третьем столбце таблицы увеличивается на единицу.

**Регистровый буфер**

|  |  |  |
| --- | --- | --- |
| Исполнительный  адрес | Содержимое | Признак  “старения” |
|  |  |  |
| **...** | **...** | **...** |
|  |  |  |

Короче говоря, аппаратура решает, какую из строк надо вытолкнуть из таблицы, чтобы на ее место записать новое содержимое. При этом учитывается информация о том, были ли обращения к данной строке с использованием команд записи в память. Если такие обращения были, то перед выталкиванием происходит запись в ОЗУ по исполнительному адресу содержимого нашей строчки.

Алгоритм записи в оперативную память симметричен. Когда в программе встречается команда записи операнда в память, аппаратура выполняет следующие действия. Проверяется наличие в буфере строки с заданным исполнительным адресом. Если такая строка есть, то в поле “Содержимое” записывается новое значение, и аппаратно корректируется признак старения строк. Если такой строчки нет, то запускается описанный выше процесс выталкивания, и затем информация размещается в освободившейся строке.

Этот буфер чтения/записи служит достаточно мощным средством для минимизации обращений к ОЗУ. Наибольший эффект достигается при небольших циклах, когда все операнды размещаются в буфере, и после этого циклический процесс работает без обращений к ОЗУ. Иногда, эти буфера называют КЭШ-буферами, а также ассоциативной памятью, потому что доступ к этой памяти осуществляется не по адресу (как в ОЗУ), а по значению поля. Реально, все механизмы могут быть устроены иначе, чем мы здесь изучаем, т.к. мы изучаем некоторую обобщенную систему.

Следующим компонентом, который мы с вами рассмотрим с точки зрения системного подхода, а системный подход подразумевает то, что вы рассматриваем вещь не саму по себе, а в контексте взаимосвязи с другими компонентами, это некоторые свойства ОЗУ.

**Оперативная память**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1-й блок |  | 2-й блок | **. . .** | k-й блок |
| 0 |  | 1 | **. . .** | k-1 |
| k |  | k+1 | **. . .** | 2k-1 |
| . . . |  | . . . | . . . | . . . |

**Использование расслоения памяти**. Физически ОЗУ представимо в виде объединения k устройств, способных хранить одинаковое количество информации и, при этом, способные взаимодействовать с процессором независимо друг от друга. При этом адресное пространство ВС организовано таким образом, что подряд идущие адреса, или ячейки памяти, находятся в соседних устройствах (блоках) оперативной памяти.

Программа состоит (в большей степени) из линейных участков. Если использовать этот параллелизм, то можно организовать в процессоре еще один буфер, который организован также, но в котором размещаются машинные команды. За счет того, что есть параллельно работающие устройства, то этот буфер автоматически заполняется вперед. Т.е. за одно обращение можно прочесть k машинных слов и разместить их в этом буфере. Далее, действия с буфером команд похожи на действия с буфером чтения/записи. Когда нужна очередная команда (ее адрес находится в счетчике команд) происходит ее поиск (по адресу) в буфере, и если такая команда есть, то она считывается. Если такой команды нет, то опять-таки работает внутренний алгоритм выталкивания строки, и новая строка считывается из памяти и копируется в буфер команд. Расслоение памяти в идеале увеличивает скорость доступа в k раз, плюс буфер команд позволяет сократить обращения к ОЗУ.

Лекция №3

Мы продолжаем рассмотрение нашей упрощенной схемы. В современных машинах имеется еще одно аппаратное средство, которое призвано поддерживать работу вычислительной системы. Это, так называемая, виртуальная память. На сегодняшний день все вычислительные машины (за исключением особо раритетных) работают в мультипрограммном (мультипроцессорном) режиме. Суть его заключается в том, что имеется несколько процессов, которые одновременно выполняются в вычислительной системе. Посмотрим, как этот мультипрограммный режим влияет на использование оперативной памяти.

|  |
| --- |
| Операционная  система |
| Задача №1 |
| Задача №2 |
| Задача №3 |
| **.**  **.**  **.** |

Предположим: в начальный момент времени какую-то часть оперативной памяти заняла операционная система (так оно и происходит). После этого была загружена программа №1 , затем программы №2, №3 (и т.д.). Операционная система начала выполнять эти программы в мультипрограммном режиме. Возникает вопрос: всегда ли любую программу можно поместить в произвольный диапазон адресного пространства оперативной памяти? Обладает ли программа свойством перемещения по памяти? Насколько задача связана с адресным пространством, на которое ее запрограммировали? Это первая проблема. Вторая проблема: этот процесс идет, и понятно, что в какие-то моменты времени какие-то из этих задач заканчиваются (например Задача №2). При этом в памяти образуются свободные фрагменты.

|  |  |  |
| --- | --- | --- |
| Операционная  Система |  | Операционная  Система |
| Задача №1 |  | Задача №1 |
|  |  | Задача №6 |
|  |  |  |
| Задача №3 |  | Задача №3 |
| **.**  **.**  **.** |  | **.**  **.**  **.** |

Программа операционной системы, которая загружает задачи в память, может посмотреть, какие из задач ожидают обработки, и если есть задача, которая помещается в один из свободных фрагментов, она может ее загрузить. Но возникает второй вопрос: а как быть, если нет такой задачи, которая поместится в освободившийся фрагмент. При этом проблема заключается в том, что даже если найдется задача (№6), которая поместится в указанном фрагменте, то останется еще меньший фрагмент памяти, в который уже нельзя практически ничего записать. Количество таких, никому не нужных, фрагментов постепенно увеличивается. Это процесс фрагментации памяти.

Фрагментация может происходить и в оперативной памяти, и на внешнем запоминающем устройстве. В результате через некоторое время возникает достаточно интересная и грустная ситуация: свободного адресного пространства много, но при этом мы не можем загрузить ни одной новой задачи. Это означает, что система в целом начинает деградировать. Например, если Вы пошлете заказ на выполнение какого-то действия, то Вам придется неоправданно долго ждать.

Для борьбы с фрагментацией памяти, а также для решения проблемы перемещения программы по адресному пространству, используется, так называемая, виртуальная память. Суть ее работы заключается в следующем. Пусть имеется некоторое адресное пространство программы, то есть то адресное пространство, в терминах которого оперирует программа. И имеется адресное пространство физическое, которое зависит от времени. Оно характеризует реальное состояние физической оперативной памяти.

В машинах, поддерживающих виртуальную память, существует механизм преобразования адресов из адресного пространства программы в физическое адресное пространство, то есть при загрузке задачи в память машины операционная система размещает реальную задачу в той оперативной памяти, которая является свободной, вне зависимости от того, является ли этот фрагмент непрерывным, либо он фрагментирован. Это первое действие выполняет операционная система. Она знает о состоянии своих физических ресурсов: какие свободны, какие заняты.

Второе: операционная система заполняет некоторые аппаратные таблицы, которые обеспечивают соответствие размещения программы в реальной оперативной памяти с адресным пространством, используемым программой. То есть можно определить, где в физической памяти какая часть программы размещена, и какая часть адресного пространства программы поставлена ей в соответствие.

После этого запускается программа, и начинает действовать аппарат (или механизм) виртуальной памяти. Устройство управления выбирает очередную команду. Из этой команды оно выбирает операнды, то есть адреса и те индексные регистры, которые участвуют в формировании адреса. Устройство управления (автоматически) вычисляет исполнительный адрес того значения, с которым надо работать в памяти. После этого автоматически (аппаратно) происходит преобразование адреса исполнительного программного (или виртуального) в адрес исполнительный физический с помощью тех самых таблиц, которые были сформированы операционной системой при загрузке данной программы в память. И продолжается выполнение команды. Аналогично выполняется и, например, команда безусловного перехода на какой-то адрес. Точно так же устройство управления вычисляет сначала адрес исполнительный, после чего он преобразуется в адрес физический, а потом значение этого физического адреса помещается в счетчик команд. Это и есть механизм виртуальной памяти.

Рассмотрим простейший пример организации виртуальной памяти: вычислительную систему с, так называемой, страничной организацией памяти. Суть страничности памяти заключается в том, что аппаратно все адресное пространство оперативной памяти разделено на блоки фиксированного объема. Обычно размер таких блоков равен степени двойки. При этом сохраняется сквозная нумерация ячеек памяти. Структура адреса в данной вычислительной машине такова: адрес любой ячейки памяти представлен в виде двух полей; старшие его разряды являются номером страницы, а младшие разряды являются смещением относительно страницы (здесь используется тот факт, что размер страницы равен степени двойки).

**Исполнительный адрес**

|  |  |
| --- | --- |
| **№ Страницы** | **Смещение относительно страницы** |

Итак, мы имеем машину со страничной организацией памяти. Для управления этой страничной памятью, процессор содержит, так называемую, таблицу приписки (ТП). Это аппаратное средство машины, реализованное с помощью регистровой памяти. Структура этой таблицы:

**Таблица приписки**

|  |  |
| --- | --- |
| **№ Виртуальной**  **страницы** | **№ Физической**  **страницы** |
| **0** | **25** |
| **1** | **1** |
| **2** | **30** |
| **.** | **.** |
| **.** | **.** |
| **.** | **.** |
| **L-1** | **-1** |

ТП имеет L строк, где L - максимальное число страниц, адресуемых в данной машине (то есть адресное пространство программы может состоять не более чем из L страниц). Каждая строка этой таблицы соответствует виртуальной странице программы с номером, совпадающим с номером строки. Строка содержит номер физической страницы, соответствующей данной виртуальной странице.

При загрузке и запуске программы операционная система размещает виртуальные страницы в некоторых физических страницах оперативной памяти, а их соответствие устанавливается в ТП. Например, операционная система взяла 0-ую виртуальную страницу памяти и поместила ее в 25-ую страницу физической памяти, 1-ую страницу разместила в 1-ой странице, а 2-ую - в 30-ой (и так далее распределила все страницы). После этого передается управления на начало программы, а за тем работает тот механизм, который был описан выше: по номеру виртуальной страницы из таблицы аппаратно выбирается соответствующая строка, и содержащийся в ней номер физической страницы аппаратно подставляется в исполнительный адрес вместо номера виртуальной страницы, то есть происходит просто замена старших битов исполнительного адреса.

На самом деле, когда операционная система обращается к какой-то строчке таблицы, то сначала она производит контроль. В действительности, строка ТП содержит некоторый код, который является либо адресом физической страницы, либо некоторым специальным кодом, на который схемы управления процессора реагируют определенным образом. Например, если этот код больше либо равен нулю, это значит, что можно продолжать работу. Если же в этой строке содержится код меньше нуля, например -1, то обращение к ней устройств преобразования виртуального адреса в физический, вызовет прерывание. Это прерывание обычно называется прерыванием по защите памяти. Дело в том, что операционная система, заполняя ТП, указывает, какие из виртуальных страниц адресного пространства не принадлежат данной программе, и если происходит выход за пределы дозволенной памяти, генерируется прерывание по защите памяти.

Для каждой программы нужна своя таблица приписки. При переходе от одной программы к другой содержимое ТП сохраняется операционной системой в некоторой своей программной таблице (массиве) и затем изменяются значения в ТП.

**Продолжение в теме “Операционные системы”**

**(Подкачка, или SWAPPING)**

***Внешние устройства***

Внешние устройства можно определить как все те устройства, которые отличаются от процессора и памяти. Управление внешними устройствами осуществляется через систему прерываний. Внешние устройства можно подразделить на Внешние Запоминающие Устройства (ВЗУ) и Устройства Ввода/Вывода (УВВ) информации. ВЗУ - это устройства, способные хранить информацию некоторое время, связанное с физическими свойствами конкретного устройства, и обеспечивать чтение и/или запись этой информации в оперативную память. Если рассматривать ВЗУ с точки зрения использования различными компонентами программного обеспечения, то можно выделить следующие типы устройств:

**1. Магнитный барабан.** Магнитный барабан - это устройство, которое характерно для больших вычислительных комплексов. Обычно оно используется операционной системой для хранения системной информации. Суть работы этого устройства состоит в следующем.

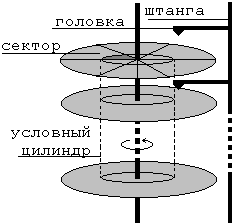


Имеется металлический цилиндр большого веса (вес здесь имеет значение для поддержания стабильной скорости вращения), который вращается вокруг своей оси. Поверхность этого цилиндра покрыта слоем материала, способного хранить информацию (с него можно читать и на него можно записывать информацию). Над поверхностью барабана размещается *p* считывающих головок. Их положение зафиксировано над поверхностями, которые называются треками (track). Каждый трек разделен на равные части, которые называются секторами. В каждый момент времени в устройстве может работать только одна головка. Запись информации происходит по трекам магнитного барабана, начиная с определенных секторов. Координатами информации служат следующие параметры (№Трека, №Сектора и Объем информации).

Для чтения информации с магнитного барабана производятся следующие действия: включается головка, соответствующая номеру трека, и прокручивается барабан до появления под головкой начала сектора с заданным номером. После этого начинается обмен. Практически во всех ВЗУ, основанных на вращении носителя, существует понятие сектора, и в каждый момент времени устройство знает, над каким сектором оно находится. Магнитные барабаны - это устройства, имеющие одну из самых больших скоростей доступа, так как электронные и механические действия в его работе минимальны (вращение барабана).

**Магнитные диски.**

Имеется несколько дисков, размещенных на одной оси, которые вращаются с некоторой постоянной скоростью. Каждый такой диск может иметь две информационно-несущие поверхности (верхнюю и нижнюю), покрытые слоем, способным фиксировать информацию. Диски имеют номера; поверхности каждого диска также пронумерованы (0,1). Концентрическим окружностям одного радиуса на каждом диске соответствует условный цилиндр. Диск также разбит на сектора. Координаты информации на диске (№Диска, №Поверхности, №Цилиндра, №Сектора).



Механически управляемая штанга имеет щупы, на концах которых находятся считывающие и записывающие головки. Количество этих щупов может быть равно количеству дисков (считывается либо верхняя, либо нижняя поверхность).

Обмен информацией осуществляется следующим образом: на блок управления диском подается набор координат с требуемым объемом информации. Блок головок вводится внутрь диска между поверхностями до заданного номера цилиндра. Затем, включается головка, читающая заданную поверхность заданного диска. После этого ожидается подход заданного сектора и начинается обмен. Здесь, в отличие от магнитного барабана, уже два механических действия, что ухудшает скоростные свойства магнитных дисков. Примерами магнитных дисков являются винчестер и гибкие диски (floppy).

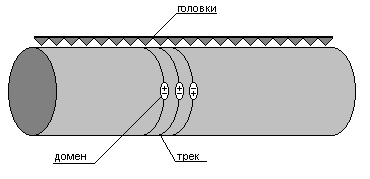
Лекция №4

Мы с вами продолжаем обзор некоторых свойств архитектуры вычислительной системы, и я обращаю ваше внимание на то, что мы рассматриваем не просто ЭВМ как набор плат, размещенных на них микросхем, каких-то проводников, механических устройств и всего прочего. Мы рассматриваем вычислительную систему, то есть систему, объединяющую аппаратуру и программное обеспечение, а также нюансы их взаимодействия. Мы с вами уже посмотрели и поговорили о некоторых таких взаимосвязях. Виртуальная память и вся та буферизация, о которой мы с вами говорили, - это аппаратные средства, которые поддерживает программное обеспечение с целью повышения эффективности работы системы в целом. Система прерываний - это также аппаратное средство, которое ориентировано на поддержку программного обеспечения, то есть средство, обеспечивающее взаимодействие программы с внешними устройствами.

Я обращаю ваше внимание на то, что эти лекции не есть лекции по UNIX-у, который мы с вами будем рассматривать, это не есть лекции по страничной организации памяти - это лекции, направленные на то, чтобы вы поняли значение слова “система” в термине “вычислительная система”. Уже сейчас мы затрагиваем механизмы работы с КЭШами, с буферами и прерываниями. И это достаточно сложно. А если мы возьмем машину, у которой не один, а несколько процессорных элементов, работающих с одной памятью, то у нас возникают достаточно сложные проблемы с буферизацией работы с памятью. То, что мы с вами рассматриваем на лекции - это очень простой срез (можно сказать, лекция для колхозников), чтобы у вас сложилось хотя бы концептуальное понимание, ибо мы не охватим всего.

Мы с вами закончили прошлую лекцию рассмотрением некоторых типов ВЗУ: магнитного барабана, магнитных дисков. Мы выявили на концептуальном уровне отличия этих устройств, потому что цель наших лекций не изучить их работу, а научиться сравнивать те или иные компоненты вычислительной системы. Мы с вами выяснили, что ВЗУ, такие как магнитный барабан и магнитные диски, могут характеризоваться степенью участия механических действий в обработке заказа на обмен. Чем больше механических действий, тем медленнее происходит обмен. Существует экзотический вид внешней памяти - память на магнитных доменах.

**Память на магнитных доменах.** Так же, как элементарной единицей электричества является электрон, элементарная единица в магнетизме - магнитный домен. Он подобен простому магниту, одна сторона которого заряжена положительно, а другая - отрицательно.

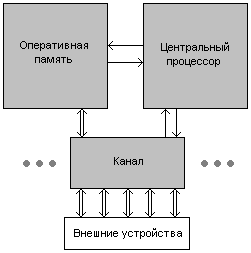


Есть барабан, у которого так же есть треки, и так же над каждым треком расположена головка для обмена, но сам барабан не вращается, а за счет некоторых магнитно-электрических эффектов осуществляется перемещение по треку цепочки доменов. При этом каждый домен однозначно ориентирован, то есть либо он бежит стороной, заряженной “+”, либо стороной, заряженной “-”. Так кодируются ноль и единица. Эта память очень быстродейственна, так как в ней нет никаких механических действий. Эти устройства обычно используются во встроенных вычислительных системах. (Например, она используется в американских ШАТЛах).

Теперь рассмотрим ВЗУ иначе - по методам доступа. Суть почти любого запоминающего устройства заключается в том, что информация в нем записывается некоторыми блоками, или записями. В некоторых устройствах размер блока фиксирован и чем-то напоминает страничную организацию памяти. В некоторых устройствах размер блока может быть переменным, и определяется некоторым начальным и конечным маркером, который можно программно записать на носитель этого устройства. В контексте работы с блоками, ВЗУ можно подразделить на два типа: устройства прямого доступа и устройства последовательного доступа. В бытовом плане можно рассмотреть компакт диск (CD) и аудиокассету. Воспроизведение десятой записи на компакт диске начинается почти сразу после соответствующей команды, за счет своей системы координат, подобной системе координат магнитного диска. Это устройство прямого доступа. Устройство последовательного доступа - это аудиокассета. Если вам надо воспроизвести пятую песню, то приходится, перематывая кассету, искать начало, так или иначе прослушивая предыдущие четыре. Устройства последовательного доступа, это те устройства, которые для чтения i-той записи, должны просмотреть предыдущие (i-1)-ну запись. Прямой доступ лишен этого недостатка. Примеры устройств прямого доступа - это магнитный барабан, всевозможные вариации магнитных дисков, память на магнитных доменах и прочее. Устройства последовательного доступа - это либо магнитная лента на больших машинах, либо стримерные устройства, которые используют для организации долговременного хранения данных на маленьких машинах.

Вот два взгляда на внешние устройства. Теперь рассмотрим внешние устройства с точки зрения управления. На ранних стадиях устройство управления внешними устройствами в большей мере являлось некоторым интерфейсом в получении всех управляющих команд от процессора и передачи их конкретному внешнему устройству. Это означает, что центральный процессор (ЦП) должен был отрабатывать практически все действия, предусмотренные системой команд управления конкретного устройства. Несмотря на то, что при этом уже был реализован и использовался аппарат прерываний, позволяющий проводить обмен асинхронно с работой центрального процессора, у нас были достаточно большие потери за счет того, что процессор должен был часто прерываться на выполнение последовательности небольших по размеру указаний для управления внешним устройством. Поэтому появились специализированные устройства, которые называются каналами.

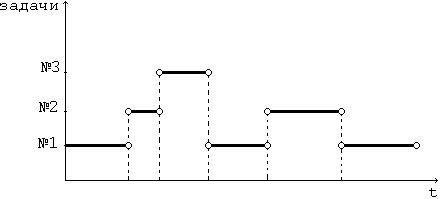
Канал - это специализированная вычислительная машина, имеющая специализированный процессорный элемент и необходимую память. Канал обычно имеет следующую структуру: у него есть высокоскоростной канал связи с оперативной памятью основной машины, управляющие каналы для взаимодействия с ЦП и имеется некоторое количество каналов для подключения внешних устройств.



Функцией канала является выполнение макрокоманд, обеспечивающих ввод/вывод. То есть ЦП подает не последовательность команд (к примеру, включить двигатель магнитных головок, переместить магнитную головку на заданный цилиндр, дождаться нужного сектора, произвести чтение порции данных, произвести проверку правильности и т.п.), а макрокоманду (произвести обмен заданного объема по заданным координатам), и необходимую последовательность команд выполняет канал. Таким образом, канал разгружает ЦП. При этом каналы могут быть достаточно интеллектуальны. В канале может быть организована буферизация за счет своей внутренней памяти (аналогично буферизации при работе с оперативной памятью). Современная вычислительная машина может иметь несколько таких каналов, которые позволяют организовывать управление внешними устройствами, а с другой стороны освобождают процессор от лишней работы.

***Мультипрограммирование***

Давайте рассмотрим еще один внутрисистемный аспект, который может продемонстрировать взаимное влияние программного обеспечения и аппаратуры. Этот аспект связан с мультипрограммированием.



Нарисуем простую диаграмму:

Ось абсцисс связана со временем. На оси ординат отмечены выполняемые задачи. Пусть процессор работает над Задачей №1. В какой-то момент ему потребовались данные, которые находятся на ВЗУ. Формируется заказ на обмен, но данные с устройства еще не поступили (мы знаем, что скорость обмена низка). И какое-то время процессор простаивает. Затем обмен завершился, и программа начала выполняться далее. Затем эта ситуация повторяется (опять простой и т.д.). В зависимости от типа или класса решаемой на этой машине задачи, таких простоев может быть от 99% всего времени до единиц процентов в том случае, если программа долго что-то вычисляет, не обращаясь к внешним устройствам. Но в любом случае эта ситуация приводит к неэффективной работе вычислительной системы, поэтому было бы неплохо при наличии асинхронно работающих устройств (то есть тех устройств, которые работают под управлением аппарата прерываний) в промежутки времени, когда одна программа не может выполняться (так как она ожидает данные), запускать другую программу (Задачу №2). Задача №2 будет жить по тем же правилам (то есть в периоды обмена Задачи №2 и Задачи №1 с внешними устройствами будет запускаться третья задача, и так далее).

*Режим работ программного обеспечения и аппаратуры, обеспечивающий одновременное выполнение нескольких программ, называется мультипрограммным режимом.* Изначально мультипрограммирование появилось в целях максимальной загрузки процессора, поскольку когда-то это устройство было самым дорогостоящим. На сегодняшний день это устройство одно из самых дешевых в компьютере. Посмотрим, что нужно от аппаратуры вычислительной системы для поддержания режима мультипрограммирования. Давайте перечислим те проблемы, которые могут возникнуть, когда, кроме некоторой управляющей программы (операционной системы) и одной программы пользователя, появилась еще одна программа пользователя. То есть, если ранее все ошибки и некорректности в отношении системы приводили к наказанию самого себя, то теперь, при появлении чужих программ в памяти возникают новые проблемы.

*Первая проблема*: кто-то взял и записал в пространство моей программы свою информацию или считал из пространства моей программы мою информацию (может быть, конфиденциальную). Если программа испорчена, то будет сбой в системе. Следовательно, возникает проблема защиты памяти, то есть в вычислительной системе должен быть реализован на аппаратном уровне механизм, обеспечивающий защиту адресного пространства каждой из программ от несанкционированного доступа других программ. Это означает, что этот механизм при выполнении доступа по исполнительным адресам будет контролировать корректность доступа.

На прошлой лекции мы с вами рассматривали механизм виртуальной памяти на примере страничной организации памяти. Если в таблице приписки в какой-то строке находился код меньше нуля, это означало, что эта виртуальная страница недоступна. И при попытке обратиться к этой странице, срабатывает аппарат защиты памяти. А срабатывает он таким образом: в системе возникает прерывание по защите памяти, при обработке этого прерывания операционная система (ОС) смотрит, а действительно ли этой страницы памяти у данной программы нет (то есть это чужая страница). Если эта страница действительно чужая, то прерывание прекращает выполнение данного процесса с диагностикой обращения в чужую память. Возможна и другая ситуация: если какие то страницы еще не загружены в память и ОС в своих внутренних программных таблицах отметила, что на самом деле у меня есть эта страница и что, предположим, она находится на внешнем устройстве. В этом случае прерывание игнорируется, так как ошибки нет (просто нет еще достаточно информации для продолжения программы). Так устроена защита памяти.

*Вторая проблема*: пусть в вычислительной системе есть принтер, и есть два процесса, которым он нужен. И пусть один из процессов обратился к внешнему устройству с командой напечатать строку (а он печатает, скажем, ведомость на экзамен, а другой процесс печатает, предположим, докладную декану). И первая строчка напечаталась: “Ведомость”. После этого другой процесс пишет: “Декану факультета ВМК чл. корр. РАН Д.П. Костомарову”, и тоже обращается к устройству печати. Устройство печати печатает вторую строчку, и так далее. В итоге получается документ, который остается только сжечь, так как там напечатан бред. Это первый пример. Второй пример: мы с вами говорили о виртуальной памяти и о том, что таблицы приписки заполняются оперативной системой программно (существуют команды, которые позволяют записать некоторую информацию в таблицу приписки). И пусть, появились два лихих программиста, которые решили отвести себе побольше памяти, и каждый из них исправил соответственно таблицу приписки, выделив себе всю память. Но память-то одна, и получилось нехорошо. И в первом и во втором примере программам пользователя были доступны команды управления компонентами вычислительной системы (в первом случае - команды управления внешним устройством, во втором случае - оперативной памятью), что и привело к некорректности в работе. Это означает, что, пока эти команды доступны, то корректный режим мультипрограммирования организовать невозможно, потому что всегда найдутся два молодца, которые чего-нибудь да испортят (даже не специально, а просто по незнанию).

Итак, вторым условием для организации мультипрограммного режима является наличие привилегированного режима в системе*. Привилегированный режим - это режим работы центрального процессора, в котором программе доступны все возможные команды. Непривилегированный режим (еще его называют пользовательским) - режим, в котором доступно лишь некоторое подмножество команд*. В пользовательском режиме программы пользователя не смогут обратиться напрямую к управлению компонентами вычислительной системы.

Может возникнуть вопрос: а если все-таки надо печатать, и работают несколько программ, как быть? Обычно во всех системах имеются средства обращения к ОС (ОС работает в привилегированном режиме и ей доступно все). Программа пользователя может передать заказ на некоторые действия ОС (в разных системах это реализовано по-разному), например, с помощью прерываний. Вернемся к примеру с принтером. Теперь процесс не вызывает конкретную команду, а обращается с заказом напечатать строку к ОС. Она принимает заказ и строку, и буферизует эту информацию в некоторых своих программных буферах. Реальной печати при заказе не происходит, очередная строчка просто попадает в буфер, который относится к моей программе. Печать этого буфера будет происходить только при завершении программы, когда уже ясно, что заказов на печать больше не будет.

*Третья проблема*: в одной из программ появилась ошибка, программа зациклилась и система зависла. Что в этом случае может помочь? Может помочь некоторое средство, которое будет периодически прерывать выполнение программы. Для нашего примера это должно быть прерывание по таймеру, чтобы один раз в какой-то промежуток времени процесс прерывался, и управление передавалось ОС. Она же сама должна решать, как ей поступить в этом случае.

Вот три условия на аппаратуру вычислительной системы, которые должны быть выполнены, если необходимо организовать мультипрограммный режим работы.

***Аппарат подкачки***

Следующая проблема: в системе, работающей в мультипрограммном режиме, находится много программ. Пусть есть реальная оперативная память со страничной организацией. Какую-то часть памяти занимает ОС, какую-то часть занимает Задача №1, какую-то часть - Задача №2 и т.д. В простейшем случае, в системе может находится столько программ, сколько может разместить в себе оперативная память. Это достаточно неэффективно, потому что выполняемый в данный промежуток времени код программы локализован в нескольких виртуальных страницах (например, какой-то цикл или функция, которые выполняются в данный момент). Если мы будем размещать в оперативной памяти весь код и все данные нашей программы, то большая часть оперативной памяти будет простаивать, потому что мы до нее еще должны добраться. Появляется естественное желание держать в оперативной памяти только те фрагменты адресного пространства, которые в данный момент используются. Для этого используется аппарат подкачки (или swapping). Это программно-аппаратное средство.

Суть его заключается в следующем. Мы говорили о виртуальной памяти и о том, что в строке таблицы приписки содержится некий код. В этой строке может быть еще одно поле, характеризующее частоту обращения к странице. Это поле формируется аппаратно. Аппарат подкачки работает примерно так же, как в буфере чтения записи в оперативную память (также собирается информация о старении страниц). По этим данным ОС может откачивать редко используемые страницы, принадлежащие конкретному процессу во внешнюю память (в больших машинах для этих целей обычно используются магнитные барабаны). При этом, если страница откачивается, то в таблицу приписки записывается отрицательный код, который, в случае обращения к ней, вызовет прерывание (оно определит, что это не чужая страница), и страница подкачивается обратно (а в это время опять же может начать выполняться другая задача). Этот механизм подкачки достаточно эффективен. Он позволяет держать в оперативной памяти от каждой из выполняемых программ некоторую (самую важную) часть, а когда в памяти находится большое количество программ, то это означает, что всегда будет готова очередь программ на случай обмена.

С одной стороны, подкачка - это функция, которая поддерживается аппаратурой (старение страниц памяти без аппаратной поддержки сложно организовать), а с другой стороны все реальное управление программное.

Теперь вернемся к нашей пирамиде:

|  |
| --- |
| Прикладные программы |
| Системы программирования | | |
| Управление логическими устройствами | | | | |
| Управление физическими устройствами | | | | | | |
| Аппаратные средства | | | | | | | | |

Мы видим, что все то, о чем говорилось в предыдущих лекциях тесно взаимосвязано. То есть нельзя организовать в системе (корректный) мультипрограммный режим без аппаратной поддержки, и наоборот, если есть аппаратная поддержка, но нет программы, которая использует эти аппаратные средства, то мультипрограммный режим также невозможен.