БИЛЕТ № 28

1) TSS. Структура и назначение.

TSS – занимает 104 байта (содержит состав регистров процессора и регистра флагов).

TR - указатель какая текущая задача (указывает на селектор, который указывает на где находится TSS) и туда переписывается содержимое регистров, поле CR3 находится в середине TSS. При вызове из CR3 содержимое записывается в TSS.

В TSS находится вся информация, которая нужна для восстановления задачи после прерывания. Планировщик информирует селектор, и по нему находится адрес входящей задачи.

В TR хранится дескриптор текущей задачи, а в CR3 лежит адрес задачи, которую нужно загрузить.

В регистр TR записывается селектор расположения TSS для активированного процесса. При смене процессов с помощью содержимого TR в TSS записывается содержимое регистра SS, CS, DS. В этих регистрах указаны селекторы этих сегментов при восстановлении задачи. Место расположения TSS каждой задачи находится в GDT.

Допю инфа

Структура дескриптора TSS.

Задача и сегмент состояния задачи

До сих пор мы говорили о параллельной работе программ. На самом деле каждая отдельная программа может состоять из нескольких частей, работающих параллельно. Например, текстовый процессор может содержать программные модули, которые параллельно с редактированием текста выполняют нумерацию страниц, печать текста или автоматическое сохранение его на диске. Каждую такую часть программы мы будем называть задачей.

Исходя из этой терминологии в мультизадачной среде одновременно выполняется много задач, принадлежащих разным программам. Причём количество задач больше или равно количеству выполняющихся программ.

Как правило, квантование времени процессора выполняется на уровне задач, а не на уровне программ. По прерыванию таймера процессор переключается от одной задачи к другой, и таким образом осуществляется параллельное выполнение программ.

Для хранения контекста неактивной в настоящей момент задачи процессор i80286 использует специальную область памяти, называемую сегментом состояния задачи TSS (Task State Segment). Формат TSS представлен на рис. 14.

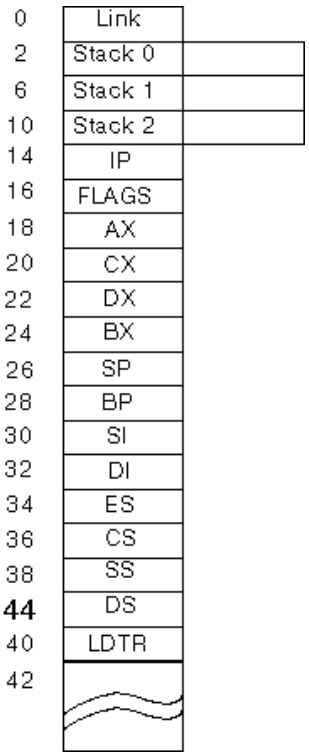

Рис. 14. Формат сегмента состояния задачи TSS.

Сегмент TSS адресуется процессором при помощи 16-битного регистра TR (Task Register), содержащего селектор дескриптора TSS, находящегося в глобальной таблице дескрипторов GDT (рис. 15).

Рис. 15. Дескриптор сегмента состояния задачи TSS.

Поле доступа содержит бит B - бит занятости. Если задача активна, этот бит устанавливается процессором в 1.

Операционная система для каждой задачи создаёт свой TSS. Перед тем как переключиться на выполнение новой задачи, процессор сохраняет контекст старой задачи в её сегменте TSS.

Что же конкретно записывается в TSS при переключении задачи?

Записывается содержимое регистров общего назначения AX, BX, CX, DX, регистров SP, BP, SI, DI, сегментных регистров ES, CS, SS, DS, содержимое указателя команд IP и регистра флажков FLAGS. Кроме того, сохраняется содержимое регистра LDTR, определяющего локальное адресное пространство задачи.

Дополнительно при переключении задачи в область TSS со смещением 44 операционная система может записать любую информацию, которая относится к данной задаче. Эта область процессором не считывается и никак не модифицируется.

Поле Link представляет собой поле обратной связи и используется для организации вложенных вызовов задач. Это поле мы рассмотрим в следующем разделе.

Поля Stack 0, Stack 1, Stack 2 хранят логические адреса (селектор:смещение) отдельных для каждого кольца защиты стеков. Эти поля используются при межсегментных вызовах через вентили вызова.

Для обеспечения защиты данных процессор назначает отдельные стеки для каждого кольца защиты. Когда задача вызывает подпрограмму из другого кольца через вентиль вызова, процессор вначале загружает указатель стека SS:SP адресом нового стека, взятого из соответствующего поля TSS.

Затем в новый стек копируется содержимое регистров SS:SP задачи (т.е. адрес вершины старого стека задачи). После этого в новый стек копируются параметры, количество которых задано в вентиле вызова и адрес возврата.

Таким образом, при вызове привилегированного модуля через вентиль вызова менее привилегированная программа не может передать в стеке больше параметров, чем это определено операционной системой для данного модуля.

Включение адресов стеков в TSS позволяет разделить стеки задач и обеспечивает их автоматическое переключение при переключении задач.

2) Взаимодействие конроллера прерываний и ОС.
3) Системы работы с памятью.
4) Согласование файлов.

В случае сбоя файловой системы для согласования составляется два вектора (битовых массива). Для занятых и для свободных секторов. Если бит занят и не свободен (после сбоя) то система ставит что он свободен. Если для сектор и бит занятости и бит свободности равен 1 , то в векторе для свободных ставится 0. Если в векторе6 занятых стоит 2, значит из двух файлов идёт обращение к этому сектору, тогда система копирует этот сектор и раскидывает эти два указателя (расшивка).

Если есть информация о свободных и занятых местах, то при сбое системы выполняется согласование файловой системы: составляется 2 битовых вектора : занятые и свободные и проверяется на разсогласованность

Программа согласования файловой системы ведет борьбу с рассогласованием файловой системы. Система пытается проверить соответствуют записи занятых файлов реальным записям. Рассогласования может появится например при непредвиденном выключении системы, когда информация о записях файлов (например таблицы ФАТА), не была скопирована на жесткий диск, хотя физически информация поменялась.

В программах согласования составляют 2 таблицы: занятых и свободных кластеров.

Далее проверяются записи, если есть указатель на кластер тогда в таблицу занятых кластеров вносится 1. Если он свободен – тогда 1 в таблицу свободных кластеров.

Тогда сочетание 1(таблица занятых) и 0 (таблица свободных) – это нормальное сочетание, а вот 1 (таблица занятых) и 1 (таблица свободных) – несогласования. Если на 1 и тот же кластер ссылается 2 файла, то фиксируется ошибка и кластер дублируется еще раз и ссылка расшивается. Часто для повышения эффективной работы с файлом рекомендуется отображение файла в ОП.

Программа согласования файловой системы. В каких случаях она применяется.

Это scandisk, fsck. Применяется, если не вышли из системы, не сохранив КЭШи файловой системы.

Для FAT: Составляем таблицу занятых кластеров (по дескрипторам файлов) и таблицу свободных кластеров (по FAT). Если кластер одновременно и «занят» и «свободен», то помечаем его как занятый. Если на один кластер ссылается два файла, то делается дубль кластера.

Такие системы, как JFS, XFS, Ext3 не нуждаются в таком восстановлении, поскольку записывают все выполняемые ими действия в «журнал».

5) Назначение работы файловой системы, ее оптимизация.

Конспект: ФС исторически развивались из библиотек.

ФС обладают оптимизацией своей работы.

ФС решает след проблемы:

- записать файл на диск так чтобы его можно было быстро найти и прочитать

Поэтому испол спец структ действия, как:

1)быстрый поиск свободного места

2)кеширование данных, читаемых с диска

3)упреждающее чтение

4)операции блокирования и разблокирования записей

5)объединение секторов в кластеры

Таненбаум:

Всем компьютерным приложениям нужно хранить и получать информацию. Во время работы процесс может хранить ограниченное количество данных в собственном адресном пространстве. Однако емкость такого хранилища ограничена размерами виртуального адресного пространства. Для некоторых приложений такого размера вполне достаточно, но для других, например систем резервирования авиабилетов, систем банковского или корпоративного учета, одного только виртуального адресного пространства будет недостаточно.

Кроме того, после завершения работы процесса информация, хранящаяся в его адресном пространстве, теряется. Для большинства приложений (например, баз данных) эта информация должна храниться неделями, месяцами или даже вечно. Исчезновение данных после завершения работы процесса для таких приложений неприемлемо. Информация должна сохраняться даже при аварийном завершении процесса в случае сбоя компьютера.

Третья проблема состоит в том, что часто возникает необходимость нескольким процессам одновременно получить доступ к одним и тем же данным (или части данных). Если интерактивный телефонный справочник будет храниться в адресном пространстве одного процесса, то доступ к нему будет только у этого процесса. Для решения этой проблемы необходимо отделить информацию от процесса.

Таким образом, к долговременным устройствам хранения информации предъявляются три следующих важных требования:

1. Устройства должны позволять хранить очень большие объемы данных.
2. Информация должна сохраняться после прекращения работы процесса, использующего ее.
3. Несколько процессов должны иметь возможность получения одновременного доступа к информации.

Обычное решение всех этих проблем состоит в хранении информации на дисках и других внешних хранителях в модулях, называемых **файлами**. Процессы могут по мере надобности читать их и создавать новые файлы. Информация, хранящаяся в файлах, должна обладать **устойчивостью** (в данном контексте иногда применяется термин **персистентность**), то есть на нее не должны оказывать влияния создание или прекращение работы какого-либо процесса. Файл должен исчезать только тогда, когда его владелец дает команду удаления файла.

Файлами управляет операционная система. Их структура, именование, использование, защита, реализация и доступ к ним являются важными пунктами устройства операционной системы. Часть операционной системы, работающая с файлами, называется **файловой системой**. Ей и посвящена данная глава.

С точки зрения пользователя наиболее важным аспектом файловой системы является ее внешнее представление, то есть именование и защита файлов, операции с файлами и т. д. Такие детали внутреннего устройства, как использование связанных списков или бит-карт для слежения за свободными и занятыми блоками диска, число физических секторов в логическом блоке, представляют для пользователя меньший интерес, хотя и крайне важны для разработчиков файловой системы. По этой причине мы разбили главу на несколько разделов. Первые два раздела посвящены пользовательскому интерфейсу файлов и каталогов. В следующих разделах мы рассмотрим способы реализации файловой системы. Наконец, будут приведены несколько примеров существующих файловых систем.