Под мультизадачностью подразумевают способность компьютера выполнять несколько задач одновременно. На самом деле процессор некоторое время выполняет один командный поток, затем быстро переключается на второй и выполняет его, переключается на третий и т. д. При этом при каждом переключении сохраняется контекст прерываемого потока, так что потом процессор сможет "безболезненно" продолжить выполнение прерванного потока команд. Благодаря высокому быстродействию создается иллюзия того, что все задачи выполняются одновременно (параллельно).

Для управления мультизадачностью нет специальных команд. Задачи переключаются командами **FAR CALL, FAR JMP, INT, IRET**. Однако при этом участвуют специальные дескрипторы: дескриптор сегмента состояния задачи (Task State Segment) и дескриптор шлюза задачи. Когда управление передается на один из таких дескрипторов, происходит переключение задачи. При переключении задачи процессор сохраняет (восстанавливает) свой контекст в сегменте состояния задачи (TSS). Селектор TSS выполняемой задачи хранится в регистре задачи (Task Register). При переключении задачи процессор может сменить LDT, что позволяет назначить каждой задаче свое адресное пространство, недоступное для других задач. Можно также перегрузить CR3 (PDBR/PDPTR), что позволяет применить для изолирования задач механизм страничного преобразования.

Для работы мультизадачной системы нужны следующие основные факторы:

1.Каждая задача (т.е. программа) должна быть изолирована от других, её код, данные и стек должны использоваться только ею самой и в идеале задача не должна "подразумевать" наличие других задач.

2. В мультизадачной системе должна быть одна задача - "хозяин", которая имеет высший приоритет по сравнению со всеми остальными задачами. Это нужно для того, чтобы переключать между собой задачи и управлять ресурсами, доступ к которым одновременно возможен для нескольких задач.

3. Использование разделяемых ресурсов (т.е. тех, которые разделяют между собой несколько задач, например, дисковая подсистема) должно быть корректным - либо каждая задача должна быть "в курсе" того, что кроме неё ещё кто-то может использовать этот ресурс, либо для каждого ресурса создаётся свой диспетчер - процедура или даже отдельная задача, которой принадлежит исключительное право на управление ресурсом.

4. При переключении с одной задачи на другую система должна обеспечить способ, благодаря которому состояние "старой" задачи не изменится состоянием "новой" задачи.

5. Необходимо учитывать временные характеристики - каждая задача должна подразумевать, что её выполнение может быть прервано в любой момент на неопределённое время. Этот фактор нужно учитывать при построении систем ввода/вывода, зависящих от времени, например, задача, управляющая модемом не должна терять данные по причине своего прерывания.

Сегмент состояния задачи TSS (Task State Segment) - это структура данных, которая определяет состояние (т.е. контекст) задачи. В ней хранится содержимое всех регистров общего назначения, сегментных и некоторых системных регистров а также некоторая дополнительная информация.

Для каждой задачи в системе нужно определить один сегмент TSS и далее процессор будет сам использовать эти сегменты задач, т.е. структура TSS поддерживается аппаратно.

Биты 31 … 15 … 0	Поле			Смещение
0	Link			00h
ESP0				04h
0	SS0			08h
ESP1				0Ch
0	SS1			10h
ESP2				14h
0	SS2			18h
CR3				1Ch
EIP				20h
EFLAGS				24h
EAX				28h
ECX				2Ch
EDX				30h
EBX				34h
ESP				38h
EBP				3Ch
ESI				40h
EDI				44h
0	ES			48h
0	CS			4Ch
0	SS			50h
0	DS			54h
0	FS			58h
0	GS			5Ch
0	LDTR			60h
I/O Map	0		T	64h

Link (смещение 0) - это поле обратной связи. Когда процессор переключается с одной задачи на другую, он записывает в это поле информацию о предыдущей задаче, чтобы "знать", куда ему возвращать управление после того, как эта задача отработает.
SSi:ESPi (i = 0,1,2) - это пары значений указателя на стек для разных уровней привилегий - 0, 1 и 2. Задачу можно вызывать с разных уровней привилегий и для корректной работы нужно разделение стека. Работа с этими полями - отдельная тема и пока мы использовать их не будем.
CR3 (смещение 1Ch) - это содержимое регистра CR3 для данной задачи. Как вы помните, CR3, он же PDBR, определяет параметры каталога страниц. То, что каждая задача имеет в своём TSS поле CR3 означает, что каждая задача может иметь свой "личный" набор страниц и работать по своей собственной схеме страничного преобразования. Однако, на практике, удобнее использовать одно значение для всех CR3 - это повышает надёжность и защищённость системы.
LDTR (смещение 60h) - это селектор дескриптора LDT, который используется данной задачей. Если задаче не нужна LDT, то в этом поле хранится 0 - вот здесь проявляется смысл условия, по которому нулевой дескриптор в GDT не используется - нулевой селектор означает отсутствие этого селектора. Если LDT используется, то задача может иметь свой собственный набор дескрипторов.

T (0-й бит по смещению 64h) - флаг трассировки, применяется для отладки задач, процессор его только считывает. Если этот флаг установлен, то при переключении на данную задачу процессор генерирует исключение отладки (прерывание 1). Этот флаг также может быть полезен в системах, использующих FPU (сопроцессор), MMX и XMM. Как вы заметили, в TSS нет полей для значений регистров этих модулей, поэтому для сохранения и загрузки контекста этих устройств требуется дополнительные усилия со стороны операционной системы.
I/O Map (смещение 66h) - это адрес карты ввода вывода, точнее - смещение её начала от начала TSS. Каждая задача может иметь такую карту, в которой каждый бит определяет возможность доступа к конкретному порту - установлен - нет доступа, сброшен - есть. Таким образом реализуется защита на уровне доступа к портам ввода/вывода.
Дополнительная часть TSS (начиная со смещения 68h) - это та часть TSS, которая предусматривается программистом при создании данной задачи. Здесь должна храниться карта ввода/вывода (если она есть), здесь также может быть контекст FPU, MMX и XMM и другая информация, в зависимости от конкретной реализации ОС. Процессор непосредственно обращается только к карте ввода/вывода, к остальной информации он явно не обращается и на размер этой части не накладывается особых ограничений (т.е. не более 1Мб).

TSS определяется специальным дескриптором. Такой дескриптор является системным объектом и может находиться только в GDT (в LDT - нельзя), а это значит, что задача самостоятельно не может определять для себя другие задачи в LDT и для операционной системы есть условия для контроля задач.

Формат дескриптора TSS следующий:

```
dw limit_low ; Младшая часть предела
dw address_low ; Младшая часть адреса
 db address_mid ; 3-й (из 4-х) байт адреса
 db access_rights ; Права доступа
db limit_hi ; Старшая часть предела
db address_hi ; Старший байт адреса
```

Примечание 1:

Байт прав доступа access_rights имеет следующий формат:

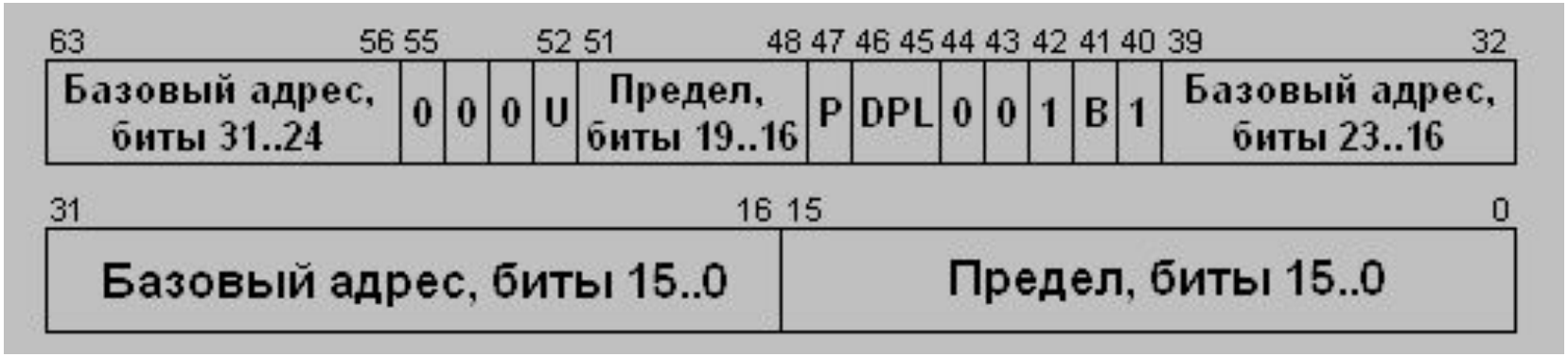
```
бит: описание
 0: = 1
 1: Бит B (Busy) - занятость задачи
 2: = 1
 3: = 0
 4: = 0
```

5,6: = DPL - Уровень привилегий сегмента TSS

7: P - бит присутствия сегмента, обычно установлен в 1.

Старшая часть предела limit_hi имеет такой формат:

```
 бит: описание
0..3: Старшие 4 бита предела
   4: Бит U (User) - этот бит ОС может использовать в своих целях
5..7: = 0
```



Дескриптор TSS определяет сегмент состояния задачи. Этот дескриптор описывает системный объект, т.к. бит S в нём равен 0 (это 44-й бит дескриптора или 4-й бит прав доступа). То, что дескриптор описывает системный объект означает, что непосредственно пользоваться этим объектом может только процессор, но не программа - любая попытка загрузки селектора этого дескриптора в сегментный регистр вызовет генерацию исключения общей защиты (#GP - прерывание 0Dh).

В локальной дескрипторной таблице LDT нельзя использовать некоторые системные объекты. Т.к. дескрипторы LDT и TSS являются такими объектами, то их нельзя применять в LDT, иначе при обращении к ним будет генерироваться исключения. Смысл такого ограничения состоит в том, что задаче запрещено определять задачи внутри себя и устанавливать дополнительные дескрипторные таблицы и таким образом, эти действия может выполнять только ядро ОС.

Переключение на задачу процессор производит автоматически и он сам использует дескриптор TSS и все поля из которых состоит сегмент состояния задачи.

Переключение производится командами далёкой передачи управления:
far jmp
far call
iret

Каждая из этих команд имеет адрес перехода, состоящий из двух частей - селектора сегмента и смещения. Для того, чтобы переключиться на задачу, нужно в качестве селектора указать селектор TSS, смещение при этом будет проигнорировано процессором, потому что в самом TSS описаны все регистры задачи, в частности, CS и EIP - вот они и определят, откуда именно будет выполняться задача.

Для переключения на задачу в процессоре не предусмотрено специальных команд, просто когда процессор обнаруживает, что селектор в адресе далёкого перехода указывает на дескриптор TSS, то запускается механизм переключения задач.
Процессор попросту не допустит любой задаче выйти за поставленные ей рамки, которые и определяются содержимым TSS - его дескрипторами, LDT, CR3 и картой разрешения ввода/вывода. Любая попытка нарушить условия защиты приведёт к генерации исключения и операционная система будет "знать" о нарушителе всё. Что дальше с ним делать уже определяется идеологией самой ОС

Для того, чтобы переключиться с одной задачи на другую, нужно чтобы процессор уже выполнял задачу, т.е. переключиться на задачу можно только из задачи. Однажды запустив мультизадачность, программа остаётся в ней до перехода в режим реальных адресов.
Переключение задач происходит следующим образом. Процессор сначала сохраняет параметры текущей задачи в её TSS, заполняя все поля, начиная с CR3. Потом он ищет в GDT дескриптор указанной новой задачи и после проверки всех параметров дескриптора, начинает загружать регистры значениями из полей нового TSS. При этом процессор контролирует значение каждого системного и сегментного регистра и если обнаруживает ошибку, то генерирует исключение. После загрузки всех значений, процессор записывает в поле Link (смещение 0 в TSS) селектор дескриптора предыдущей задачи. После всего этого процессор передаёт управление по адресу CS:EIP и задача начинает свою работу (или продолжает, если она была прервана).

Как видите, переключение задачи довольно-таки трудоёмкий процесс и требует много времени на различные проверки и обращения к памяти. Кроме того, переключение на задачу не кэшируется буфером ветвлений, представленных в процессорах, начиная с Pentium и при переключении на задачу происходит сброс конвейера команд, что потребует перед выполнением первой команды задачи потерю времени на ожидание, пока заполнится линия кэша из памяти и пока команды пройдут по конвейеру. Дополнительные задержки происходят из-за того, что при чтении новых значений из TSS, происходит загрузка CR3, а это значит, что производится сброс буферов TLB и процессору понадобится обращаться к каталогу и таблицам страниц, даже если он только что с ними работал (загрузка значения в CR3, даже того же самого, что в нём и было, приводит к сбросу буферов TLB). Наблюдая столько сложностей и проблем с этими задачами, возникает вполне логичный вопрос о целесообразности их применения, однако задачи процессором **поддерживаются аппаратно** и только из-за этого уже стоит их применять. К тому же, с каждым новым процессором Intel увеличивает производительность в первую очередь в таких проблемных местах, как переключение задач и в современных процессорах оно происходит очень быстро.

Передача управления от задачи к задаче может происходить тремя способами:

1. Одна задача вызывает другую командой FAR CALL, а вторая задача возвращает управление первой командой IRET.

2. Задача переключается на другую задачу командой FAR JMP.

3. Происходит прерывание - либо аппаратное (IRQ), либо исключение и обработчик такого прерывания является отдельной задачей. Обработчик возвращает управление прерванной задаче командой IRET.

Итак, команда IRET может производить переключение задач. Как вам известно, обычное назначение этой команды - возврат из прерывания, т.е. она отличается от RET тем, что извлекает из стека ещё и регистр флагов (FLAGS, если стек - 16-разрядный, EFLAGS - если 32-разрядный).

Как определить, что же именно выполнит данная команда - переключение на задачу или возврат из прерывания? Для этого в регистре флагов EFLAGS введён специальный флаг NT (Nested Task - вложенная задача). Каждый раз, когда процессор выполняет команду FAR CALL, вызывая задачу, этот флаг устанавливается в 1. При выполнении команды IRET процессор проверяет этот флаг - если он равен 1, то он переключается на предыдущую задачу (селектор которой хранится в поле Link по смещению 0 в TSS текущей задачи). Если NT = 0, происходит возврат из прерывания.

Если переключение на задачу было произведено командой FAR CALL либо оно произошло в результате прерывания, то процессор после загрузки контекста новой задачи установит флаг NT в регистре EFLAGS. Это обеспечит возврат в старую задачу командой IRET.

Обратите внимание на бит B (Busy) в дескрипторе TSS. Этот бит определяет занятость задачи. Каждый раз, когда процессор переключается на задачу, он устанавливает этот бит. Когда задача передаёт управление другой задаче командой FAR JMP или IRET, процессор сбрасывает флаг B в дескрипторе текущей задачи (той, из которой происходит передача управления).

Передача управления командой FAR CALL не сбрасывает флаг B. Если происходит прерывание, то этот флаг в прерванной задаче также не сбрасывается. Таким образом, установленный бит B показывает, что данная задача занята, т.е. она либо работает в данный момент, либо она прервана (находится в состоянии паузы).

Необходимость бита занятости B заключается в том, что процессор запрещает передачу управления занятой задаче (он генерирует исключение общей защиты). Это сделано для того, чтобы задачи не могли себя вызывать рекурсивно. Вот здесь проявляется отличие задачи от процедуры - задача, как программа, рекурсивной не бывает.

Однако, при помощи манипуляций непосредственно над содержимым дескрипторов TSS (сброс и установка флага B "вручную") можно добиться переключения на занятую задачу. Это иногда бывает необходимым, в частности, при запуске мультизадачности.

Дескриптор TSS содержит в байте прав доступа поле DPL (Descriptor Privilege Level). В данном случае, для дескриптора TSS, это поле содержит уровень привилегий, **с которого** процессор разрешает переключение на эту задачу. Переключения с меньшего уровня привилегий также разрешены. Другими словами, при передаче управления должно выполняться следующее условие:
CPL <= DPL, где CPL - это Current Privilege Level - текущий уровень привилегий (т.е. тот, на котором выполняется код в данный момент).

Это ещё одно средство контроля целостности системы. Для системных задач поле DPL нужно устанавливать в 00b, для прикладных задач - в 11b, при этом системная задача, с DPL = 00b может переключаться на задачу с любым DPL, а, например, задача с DPL = 10b - на задачи с DPL = 10b и 11b.

Благодаря применению мультизадачности, операционная система может контролировать операции ввода/вывода любой задачи на уровне отдельных портов. Этот контроль производится аппаратно и в случае попытки доступа к запрещённым портам ввода/вывода генерируется исключение общей защиты.

Каждая задача может иметь карту разрешения ввода/вывода. Эта карта должна располагаться в TSS задачи и её параметры задаётся следующим образом:
в TSS в dw-поле "I/O Map" (по смещению 66h) находится базовый адрес карты ввода/вывода, этот адрес - смещение начала карты от начала TSS; сверху карта ограничена пределом TSS, установленным в дескрипторе TSS. Каждый бит карты соответствует одному однобайтовому порту. Например: при выводе **байта** в порт по адресу 21, вывод будет в этот порт и будет проверяться 21-й бит карты - это 5-й (от нуля) бит в 2-м (от нуля) байте этой карты.

Если бит в карте разрешения ввода/вывода сброшен, то вывод разрешён, если установлен - то запрещён. Таким образом, если при операции ввода/вывода двойного слова процессор обнаружит, что хотя бы один из 4-х бит в карте установлен, он сгенерирует исключение общей защиты - обращение к порту запрещено.

Карта разрешения ввода/вывода используется, когда процессор работает в режиме виртуального 8086, либо когда в задаче CPL > IOPL. Если процессор находится в защищённом режиме и выполняет задачу, CPL которой не больше её IOPL, то операции ввода/вывода разрешены по всем адресам.

Таким образом, обычно для всех системных задач ввод/вывод разрешается по всем адресам, а для задач на ненулевых уровнях привилегий вывод либо запрещается вообще, либо разрешается по определённым адресам. Например, драйвера устройств в ОС расположены на 1-м уровне привилегий и имеют карту ввода/вывода с разрешёнными портами для конкретного устройства или класса устройств, обслуживаемых этим драйвером, а прикладным программам, расположенным на 3-м уровне привилегий, ввод/вывод вообще запрещён. Благодаря этому, все операции с портами производятся через драйвера, надёжность которых всегда подразумевается выше, чем прикладные программы и такая система получается устойчивой к несанкционированному доступу к портам и, следовательно, к устройствам.

Карта разрешения ввода/вывода может располагаться по любому смещению в TSS и иметь любой размер. Если адрес карты больше, чем предел TSS, либо равен ему, то считается, что карты нет и все команды ввода/вывода (при проверке карты) будут генерировать исключение. Адрес карты должен быть не более DFFFh.

При конструировании дескриптора TSS необходимо всегда помнить, что минимально допустимый предел TSS для любой задачи равен 67h. TSS с таким пределом имеет размер в 68h, т.е. в 104 байт и содержит все поля. Если при переключении на задачу процессор обнаружит, что в дескрипторе TSS указан меньший предел, то он сгенерирует исключение недопустимого TSS (#TS).

Регистр задачи TR.

TR (Task Register) - это 16-разрядный системный регистр, который хранит селектор дескриптора TSS текущей задачи. Этот регистр также имеет теневую 64-разрядную компоненту, используемую только самим процессором, в которой хранится содержимое дескриптора TSS текущей задачи - это повышает производительность процессора.

Когда происходит переключение со старой задачи на новую, процессор помещает в TSS новой задачи в поле Link содержимое регистра TR и таким образом обеспечивает связь в со старой задачей. После загрузки значений из TSS новой задачи, в регистр TR процессор записывает селектор дескриптора TSS этой новой задачи.

Для загрузки значения в регистр TR используется команда LTR, единственным операндом которой служит 16-разрядный регистр общего назначения или переменная в памяти. Для чтения значения из этого регистра используется команда STR, которая также имеет один операнд - 16-разрядный регистр общего назначения или переменную в памяти.

Команда LTR относится к привелегированным командам - она может выполняться только на нулевом уровне привилегий. Команду STR можно выполнить на любом уровне привилегий. Это может показаться странным - процессор позволяет прикладным задачам считывать "конфиденциальную" информацию - селектор дескриптора TSS, однако, на самом деле, информации значение TR программе не даёт, потому что в этом регистре находится селектор дескриптора TSS текущей задачи и задача ничего с этим сделать не сможет - переключения на саму себя запрещены.

Регистр TR имеет, в основном, два применения:

1. Считывая значение TR, программа может определить текущую выполняемую задачу. Это может пригодится при отладке, например, для вывода на экран селектора дескриптора TSS текущей задачи, либо, для тех обработчиков прерываний и исключений, которые не являются сами задачами и выполняются в контексте текущей задачи, значение из TR позволяет определить текущую задачу.
2. Для перевода процессора в режим мультизадачности необходима загрузка селектор в регистр TR. Как именно это делается, вы увидите на примерах. При загрузке регистра TR в дескрипторе TSS задачи устанавливается флаг занятости B. Загрузка этого регистра связывает текущее состояние процессора с контекстом данной задачи.

При переключении задачи с помощью прерывания или особого случая происходит автоматический возврат к прерванной задаче. Однако, организуя вложение задач, необходимо помнить, что, в отличие от процедур, задачи не являются реенетрантными, т.к. при переключении задачи в стек ничего не включается. Дескриптор TSS задачи, выполняемой в данный момент, помечается как "занятый". При переключении на другую задачу с вложением (по **INT** или **FAR CALL**) Дескриптор TSS остается помеченным. Переключиться на занятую задачу нельзя (возникает нарушение общей защиты - исключение #13).

Для переключения задач также действуют правила привилегий. По команде **JMP** или **CALL** можно переключиться на задачу, TSS которой менее привилегирован:
DPLTSS>=max(CPL,RPL).

Для особых случаев и прерываний это правило не действует. Если обработчик прерывания выполнен в виде отдельной задачи, то он может быть вызван независимо от значения CPL.

Не совсем удобно адресовать именно TSS для переключения задачи, т. к., во-первых, TSS могут быть размещены только в GDT (а в IDT или LDT - нет), а, во-вторых, если пользоваться только TSS, то каждую задачу мы "намертво" привязываем к определенному уровню привилегий (DPLTSS), с которого она доступна для переключения. Этих недостатков лишены шлюзы задачи. Шлюз задачи содержит селектор TSS. Шлюзы задач можно размещать и в IDT, что позволяет выполнять обработчики прерываний в виде отдельных задач, и в LDT, что позволяет более гибко управлять переключением задач: для второй задачи первая может быть видна с одного уровня привилегий, а для третьей - с другого. Последняя возможность обеспечивается особым правилом привилегий: при переключении задачи через шлюз учитывается только DPL шлюза, а DPLTSS не играет роли, поэтому одной задаче может соответствовать множество шлюзов с различными DPL.

Следует отметить, что при переключении задачи не сохраняется контекст сопроцессора, т.к., многие задачи могут не использовать сопроцессор, а на сохранение/восстановление его контекста уходит много времени. В процессоре предусмотрена возможность простой программной реализации переключения задачи с учетом контекста сопроцессора. Дело в том, что при переключении задачи процессор выставляет бит 3 (Task Switched) в регистре CR0. Если новая задача далее не использует сопроцессор, то ничего не происходит. В противном случае, встретив команду сопроцессора, когда флаг TS=1, процессор генерирует особый случай сопроцессора (исключение #7). Обработчик этого исключения может сохранить/восстановить контекст сопроцессора, сбросить флажок TS (командой **CLTS**) и возобновить прерванную задачу.

Лишь значение первых 68h байт сегмента состояния задачи строго определены. Именно это число является минимальным размером TSS. Операционная система может по своему усмотрению устанавливать размер TSS и заполнять сегмент данными. Она, например, может отвести в нем место под контекст сопроцессора. Кроме того, в TSS может располагаться необязательная для задачи структура - двоичная карта разрешения ввода вывода (I/O permission bit map). Ее адрес задается в последнем обязательном поле TSS. Процессор обращается к этой карте, когда IOPL не позволяет выполнять инструкцию ввода-вывода. Каждый бит этой карты соответствует одному порту ввода-вывода. Если бит сброшен в 0, то операция ввода-вывода выполняется без нарушения общей защиты, а если бит выставлен в 1, то генерируется исключение #13.

При переключении задачи процессор может продолжить выполнение новой задачи в особом состоянии - в состоянии эмуляции 8086 (VM86). Переход в это состояние инициирует установка в 1 флажка 17 (Virtual Mode) в регистре EFLAGS при восстановлении контекста процессора из TSS. Особенностью этого состояния является то, что линейные адреса в такой задаче формируются по безселекторной схеме, поэтому защита на уровне сегментов фактически не используется. Однако если включено страничное преобразование, то все связанные с ним нюансы верны и защита на уровне страниц работает. Для задачи в состоянии VM86 считается, что CPL=3, код и данные по умолчанию имеют размер 16 бит, пределы всех сегментов - 64Кбайт, адресуется только нижний мегабайт линейного пространства.

Процессор может перейти в состояние VM86 не только при переключении задач, но и если при возврате из обработчика прерываний по **IRET** из стека восстанавливается образ EFLAGS с битом VM=1. Возврат в обычное состояние происходит по прерыванию или особому случаю с переключением задачи или с переходом на нулевой уровень привилегий. В системе может быть несколько задач в состоянии VM86. Для их изолирования удобно использовать страничное преобразование.

Задачи и флаги.

Для управления задачами процессор использует несколько флагов:

Флаг занятости B (Busy) - находится в дескрипторе TSS, в байте прав доступа. Устанавливается всякий раз, когда происходит переключение на задачу и когда установлен, означает, что задача занята. Переключение на занятую задачу запрещено и этот флаг предназначен для предотвращения рекурсивного вызова задачи. Флаг занятости сбрасывается при переключении на другую задачу командами FAR JMP либо IRET; при переключении командой FAR CALL либо при прерывании (даже если обработчик прерывания - тоже задача) флаг не сбрасывается.

Флаг трассировки T - находится в сегменте состояния задачи TSS, это 0-й бит по смещению 64h в TSS. Если флаг установлен, то при переключении на задачу процессор сначала загрузит значения из всех полей TSS, проверит их правильность и, если не обнаружит нарушений, сгенерирует исключение отладки (прерывание 1). Если флаг сброшен, то при переключении на задачу исключение отладки не генерируется. Этот флаг предназначен для отладки задач и также может применяться для явного системного дополнения контекста задачи, например, для загрузки регистров FPU.

Флаг NT (Nested Task) - находится в регистре EFLAGS. Если переключение на новую задачу было вызвано командой FAR CALL либо старая задача была прервана исключением или прерыванием и его обработчик также является задачей, то флаг NT устанавливается в регистре EFLAGS новой задачи. Благодаря этому новая задача может вернуть управление старой задаче командой IRET. Команда IRET выполняет одно из двух действий:
Если NT = 0, то производит обычный возврат из прерывания;
Если NT = 1, то производит переключение на предыдущую задачу, селектор дескриптора TSS которой находится в поле Link в TSS текущей задачи.

Флаг TS (Task Switched) - находится в регистре управления CR0. Этот флаг устанавливается каждый раз, когда процессор переключается на задачу и служит индикатором переключения задач. При попытке выполнить команды FPU, MMX или XMM, процессор может генерировать исключение неприсутствующего устройства (#NM - прерывание 7), что позволяет системе выполнить смену контекста FPU, MMX и XMM. Процессор позволяет пользоваться режимом виртуального 8086 только в контексте задачи и этот режим добавляет некоторые другие флаги, с которыми взаимодействует процессор в задаче

Воздействие при переключении задачи на флаги Busy, NT, TS и поле Link.

Флаг или поле	Эффект от команды JMP	Эффект от команды CALL или прерывания	Эффект от команды IRET
Флаг B (Busy) новой задачи	Установлен. Перед переходом должен быть сброшен.	Установлен. Перед вызовом должен быть сброшен.	Не меняется. Перед возвратом должен быть установлен.
Флаг B старой задачи.	Сбрасывается.	Не меняется, он уже установлен	Сбрасывается.
Флаг NT новой задачи	Не меняется.	Устанавливается.	Восстанавливается из TSS новой задачи.
Флаг NT старой задачи	Не меняется.	Не меняется.	Сбрасывается.
Поле Link новой задачи.	Не меняется.	Загружается селектором TSS старой задачи.	Не меняется.
Поле Link старой задачи.	Не меняется.	Не меняется.	Не меняется.
Флаг TS в регистре CR0.	Устанавливается.	Устанавливается.	Устанавливается.