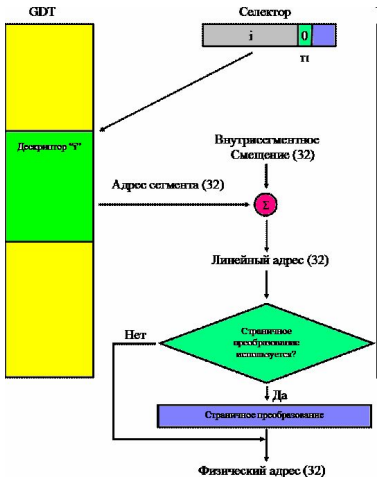
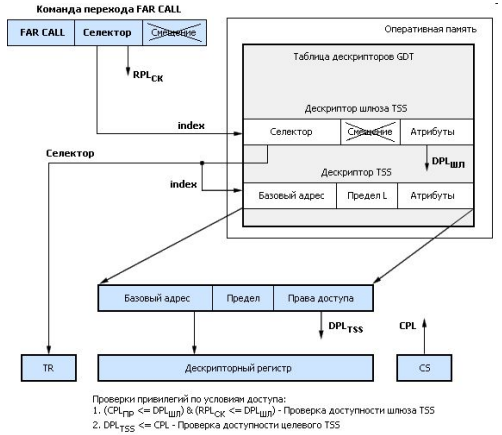
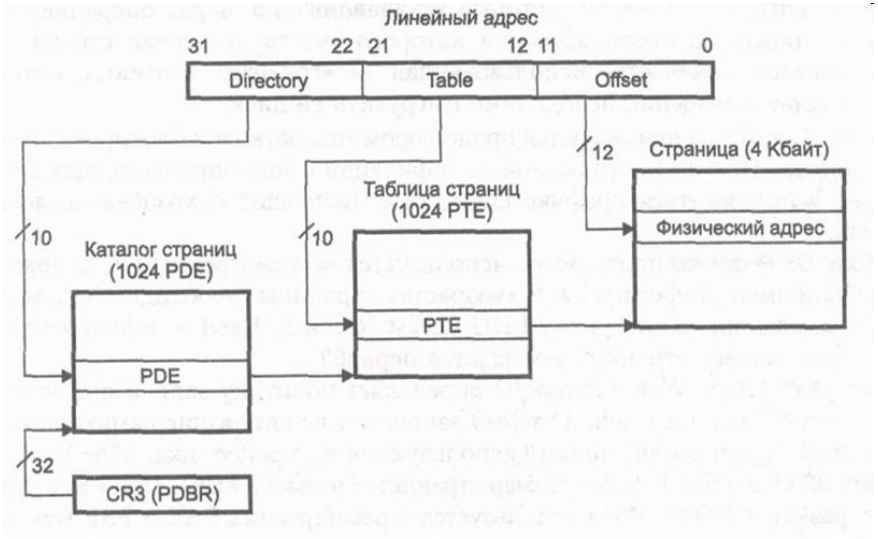
Переключение задач. Страничное упра





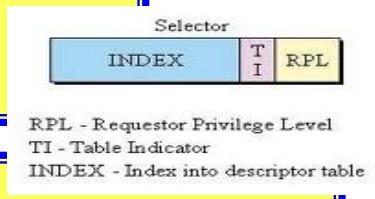


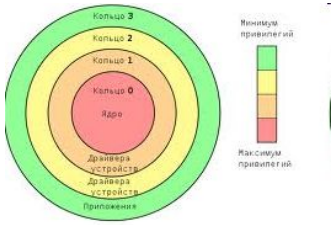
Механизмы реализации мультизадачности :

- общие понятия;
- описание сегмента состояния задачи;
- обращения процессора к памяти .

Поля Task Status Segment:

- карта разрешения ввода/вывода;
- дескриптор шлюза задачи.



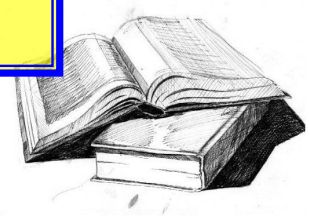


Переключения задач:

- межсегментные переключения задач;
- прямое переключение задач;
- переключение задач с использованием шлюза задач
- последовательность действий при переключении задач.

Страничное управление памятью:

- понятия свопинга;
- базовый механизм страничного управления.



Резюме к лекции и список используемой литературы

Переключение задач. Общие понятия.

Для многозадачных и многопользовательских операционных систем важна способность процессора к быстрому переключению выполняемых задач. Операция переключения задач процессора (Task Switch Operation) сохраняет состояние процессора и связь с предыдущей задачей, загружает состояние новой задачи и начинает ее выполнение. Переключение задач выполняется по инструкции межсегментного перехода (JMP) или вызова (CALL), ссылающейся на *сегмент состояния задачи* TSS (Task State Segment) или *дескриптор шлюза задачи* в GDT или LDT. Переключение задач может происходить также по аппаратным и программным прерываниям и исключениям, если соответствующий элемент в IDT является дескриптором шлюза задачи. *Дескриптор* TSS указывает на *сегмент*, содержащий полное состояние процессора, а *дескриптор шлюза задачи* содержит *селектор*, указывающий на дескриптор TSS.

Механизмы реализации мультизадачности в Intel x86.

Важнейшей особенностью защищенного режима является возможность «параллельного» выполнения нескольких программ.
Под *многозадачностью* понимается поочередное выполнение нескольких программ (или фрагментов программ - подпрограмм).

Переключение задач

осуществляется

Программно (командами CALL или JMP)

команда JMP или CALL может передать управление либо дескриптору TSS, либо шлюзу задачи.

```
JMP   dword ptr adr_sel_TSS(/adr_task_gate)

CALL dword ptr adr_sel_TSS(/adr_task_gate)
```

по прерываниям (например, от таймера)

Каждой задаче отводится определенный квант времени, после чего управление передается следующей задаче (и так циклически), в результате чего возникает иллюзия того, что все задачи выполняются враз.

Задачей может быть как целая программа, так и ее часть (например, некоторая процедура), или программа обработки прерывания.

При переключении на выполнение новой задачи процессор сохраняет состояние текущей, с тем, что бы потом возобновить ее выполнение.

http://x86.migera.ru/text/glava_5.html

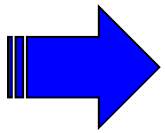
Task Status Segment

- Для хранения информации о задачи существует специальная структура – TSS (Task S Segment)
- В то время как при переходе к выполнению проце процессор запоминает (в стеке) лишь точку возвр (CS:IP)

Поддержка многозадачности обеспечивается за счет следующих аппаратно поддерживаемых структур и элементов:

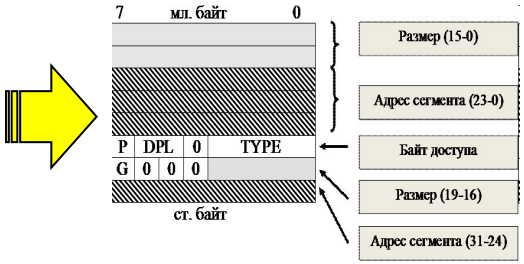
- Сегмент состояния задачи (TSS)
- Дескриптор сегмента состояния задачи
- Регистр задачи (TR)
- Дескриптор шлюза задачи.

Для сохранения состояния задачи определена такая структура, как сегмент состояния задачи (TSS - Task State Segment), может располагаться как в отдельном сегменте данных, так и внутри сегмента данных задачи



Task Status Segment. Общие понятия.

TSS описывается системным дескриптором, который может находиться только в GDT



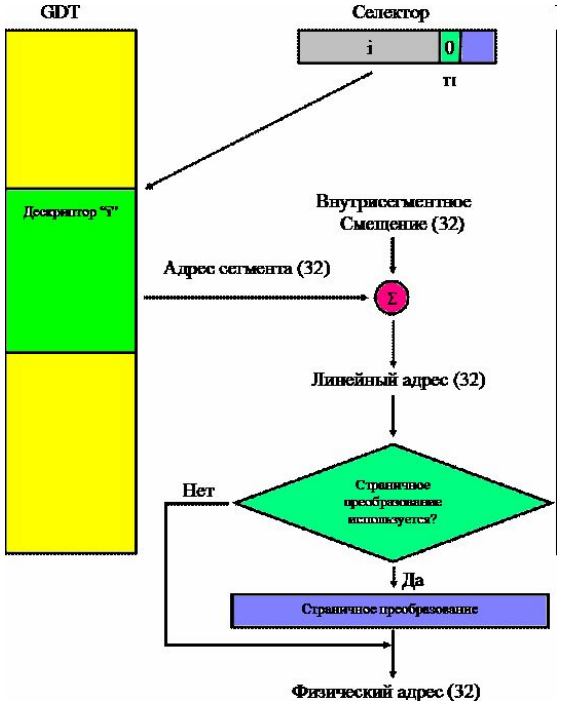
Поле TYPE для дескрипторов TSS имеет значение:

- 1 - доступный сегмент состояния задачи
- 3 - занятый сегмент состояния задачи
- 9 - доступный сегмент состояния задачи;
- B - занятый сегмент состояния задачи.

- Селектор TSS текущей задачи хранится в регистре TR.
- Регистр TR имеет "видимую" часть (т.е. часть, которую может считывать и изменять программное обеспечение) и "невидимую" часть (т.е. часть, обслуживаемую процессором и недоступную программному обеспечению).

. Селектор, находящийся в видимой части, индексирует дескриптор TSS в GDT, давайте рассмотрим как это происходит

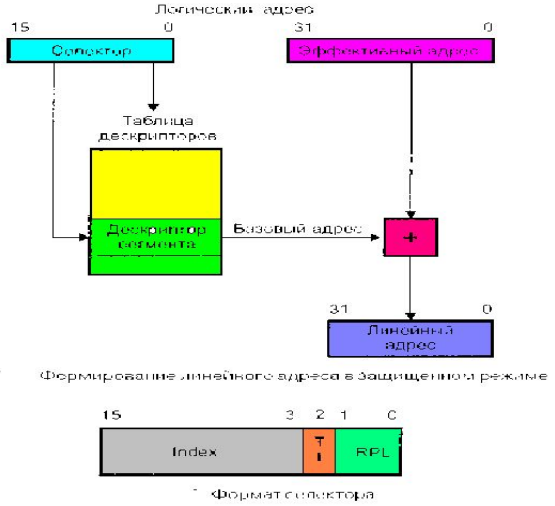
Пример обращения процессора к памяти

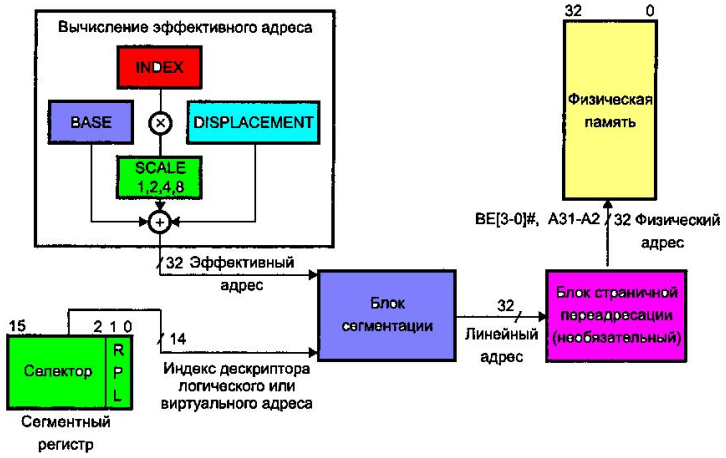


Рассмотрим пример обращения процессора к памяти в сегмент (данных), описанный в глобальной дескрипторной таблице (GDT) – т.е. бит TI(индикатором таблицы) в селекторе сегмента = 0.

RPL в данном селекторе не используется.

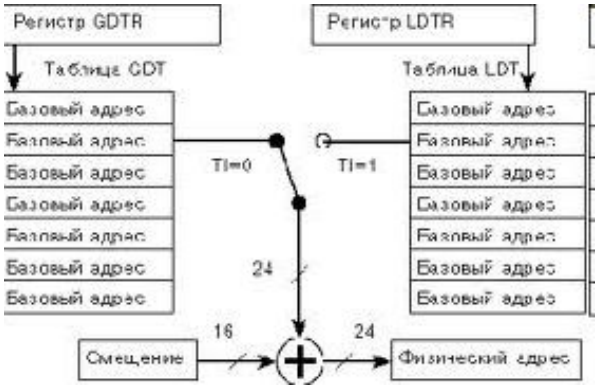
- Старшие 13 байт селектора являются смещением искомого дескриптора от начала дескрипторной таблицы.
- Младшие байты играют роль атрибутов и при адресации внутри таблицы интерпретируются как нули – за счет чего происходит адресация таблицы по полю “i” с точностью до восьмибайтного слова, или (что в данном случае одно и тоже) до дескриптора.





Пример обращения процессора к памяти

- Если бит TI в селекторе сегмента = 1, то происходит выборка сегмента из локальной дескрипторной таблицы.
- В этом случае регистр LDTR должен содержать селектор этой таблицы (индекс) – локальная дескрипторная таблица описывается дескриптором глобальной таблицы.
- Сам же селектор (в котором TI=1) определяет дескриптор сегмента памяти, описанного в локальной дескрипторной таблице.
- В регистре LDTR бит TI всегда нулевой – т. к. дескриптор-описатель локальной таблицы может находиться только в GDT.



- Адрес сегмента, прочитанный из дескриптора, складывается с внутрисегментным смещением и при отсутствии страничного преобразования адреса выставляется на адресную шину.

Поля Task Status Segment

Для каждой задачи (фрагмента кода), участвующей в переключении создается свой TSS

Поля TSS делятся на две основные категории:

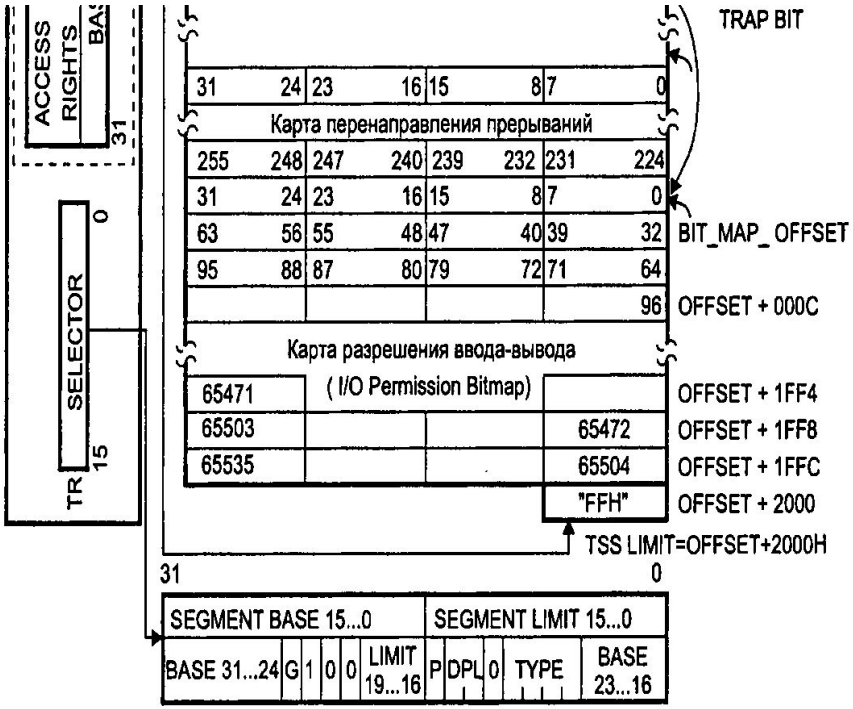
1. Динамические поля, обновляемые Процессором при каждом переключении задачи. В число этих полей входят:

- Регистры общего назначения (EAX, ECX, EDX, EBX, ESP, EBP, ESI и ED
- Сегментные регистры (ES, CS, SS, DS, FS и GS).
- Регистр флагов (EFLAGS).
- Указатель команд (EIP).
- Селектор для TSS предыдущей задачи (обновляется только когда ожидается возвр

2. Статические поля, которые процессор считывает, но не изменяет. Эти поля устанавливаются при создании задачи:

- Селектор для LDT задачи.
- Логический адрес для стеков привилегированных уровней 0,1,2.
- Бит T (бит отладочной ловушки), который, будучи установленным, заставляет процессор устанавливать при переключении задачи отладочное исключение.
- Базовый адрес битового массива разрешения ввода/вывода. При наличии, данный массив всегда хранится в TSS по старшим адресам. Базовый адрес указывает на начало массива.

Карта разрешения ввода/вывода



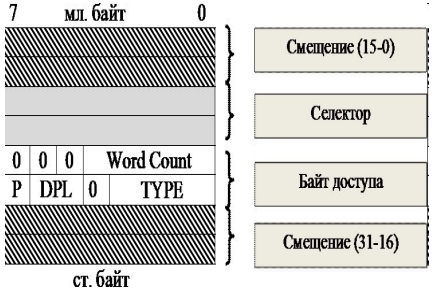
Карта разрешения ввода-вывода (I/O Permission Bit Map), расположенная в конце TSS, имеет по одному биту на каждый адрес портов .

- Разрешению обращения соответствует нулевое значение бита. Максимальный размер таблицы(2000h), соответствующий всем 64Кбайт адресов, может быть урезан лимитом TSS, но байт-терминатор $0FF_h$ должен обязательно вписываться в лимит TSS.
- Порты с адресами, не попавшими в усеченную таблицу, считаются недоступными.

Для возврата управления задаче, вызвавшей текущую задачу или ею прерванной, используется инструкция IRET. В регистре флагов имеется флаг вложенной задачи NT (Nested Task), который управляет функцией инструкции IRET. При NT = 0 IRET работает обычным образом, оставаясь в текущей задаче. При NT = 1 (текущая задача — вложенная) IRET выполняет переключение в предыдущую задачу.

Когда инструкции CALL, JMP или INT выполняют переключение задач, старый (кроме случая JMP) и новый сегмент TSS помечаются как занятые (меняется значение TYPE в их дескрипторах), и в поле обратной ссылки в новом TSS устанавливается значение селектора старого TSS. Инструкции CALL и INT, переключающие задачи, устанавливают в новой задаче бит NT. Прерывание, не вызывающее переключения задач, сбросит бит NT. Этот бит может устанавливаться и сбрасываться инструкциями POPF и IRET.

Дескриптор шлюза задачи



Дескриптор шлюза задачи обеспечивает косвенные, защищенные ссылки к задаче.

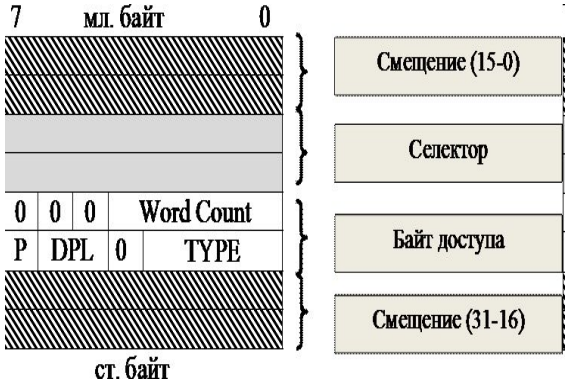
Поле «Селектор» шлюза задачи индексирует дескриптор TSS.

- RPL в данном селекторе не используется.
- DPL шлюза задачи управляет доступом к дескриптору для переключения задачи.
- Процедура не может выбрать дескриптор шлюза задачи до тех пор, пока RPL селектора и CPL процедуры не будут численно меньше или равны DPL дескриптора.

Тем самым предотвращается переключение задачи менее привилегированными, чем она сама, процедурами.

Отметим, что при использовании шлюза задачи DPL дескриптора TSS назначения не используется.

Дескриптор шлюза задачи



Linux's GDT	Segment Selectors
null	0x0
reserved	
reserved	
reserved	
not used	
not used	
TLS #1	0x33
TLS #2	0x3b
TLS #3	0x43
reserved	
reserved	
reserved	
kernel code	0x60 (__KERNEL_CS)
kernel data	0x68 (__KERNEL_DS)
user code	0x73 (__USER_CS)
user data	0x7b (__USER_DS)

Linux's GDT	Segment Selectors
TSS	0x80
LDT	0x88
PNPBIOS 32-bit code	0x90
PNPBIOS 16-bit code	0x98
PNPBIOS 16-bit data	0xa0
PNPBIOS 16-bit data	0xa8
PNPBIOS 16-bit data	0xb0
APMBIOS 32-bit code	0xb8
APMBIOS 16-bit code	0xc0
APMBIOS data	0xc8
not used	
not used	
not used	
not used	
double fault TSS	0xf8

- Доступ к дескриптору TSS не дает возможности читать или модифицировать дескриптор.
- Чтение и модификация его возможны только путем отображения в тот же адрес памяти дескриптора данных.
- Загрузка дескриптора TSS в сегментный регистр вызывает исключение.
- Дескрипторы TSS могут находиться только в таблице GDT. Попытка доступа к TSS при помощи селектора с установленным битом TI (который обозначает текущую LDT) генерирует исключение.

Межсегментные переключения задач

Межсегментные переключения задач - это переходы с использованием механизма переключения задач.

Возможны две модели переключения задач:

- прямое переключение задач;
- переключение задач с использованием шлюзов (косвенное переключение задач).

```
JMP   dword ptr adr_sel_TSS(/adr_task_gate)

CALL dword ptr adr_sel_TSS(/adr_task_gate)
```

Переключение программ может производиться командами JMP и CALL типа FAR(так называемый дальний указатель, который определяется моделью памяти), командами вызова прерываний, например, INT n, или командой IRET.
Кроме этого, переключения задач могут инициироваться прерываниями и ловушками.

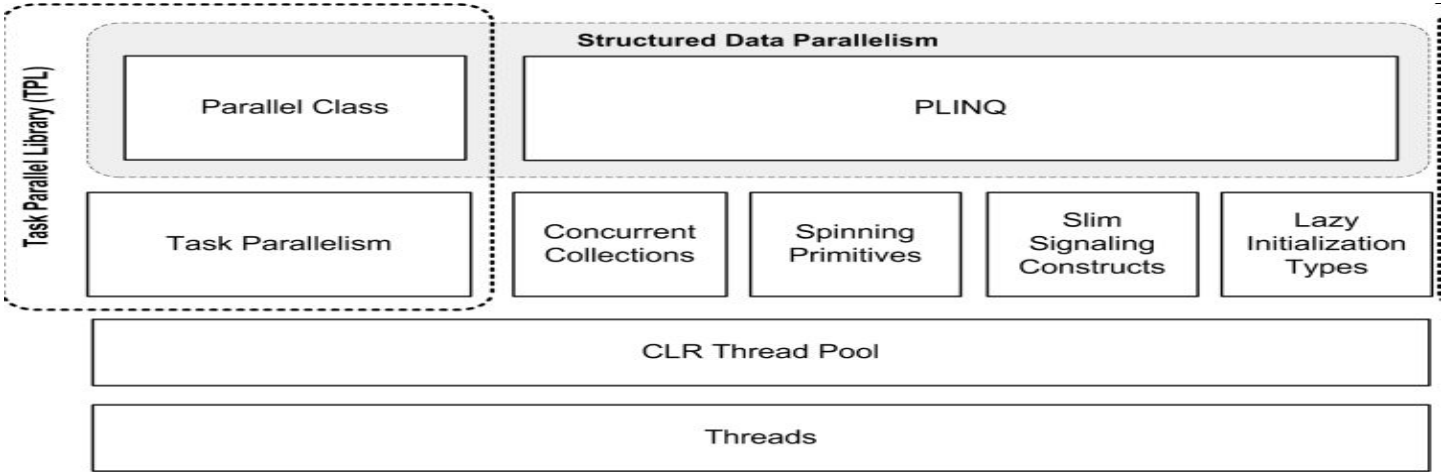
Модели переключения задач могут использоваться:

- для переключения независимых программ при параллельном их выполнении в режиме разделения времени;
- для вызова процедур;
- для вызова программ обработки прерываний и ловушек;
- для возврата из программ обработки прерываний и ловушек.

Межсегментные переключения задач

Переключение задач при параллельном выполнении программ в режиме разделения времени

- Переключение задач производится с использованием специальных сегментов состояния задач.
- Это системные сегменты, предназначенные для копирования основных регистров процессора, хранящих "контекст" программы. Иногда переключение задач называют сменой контекста.

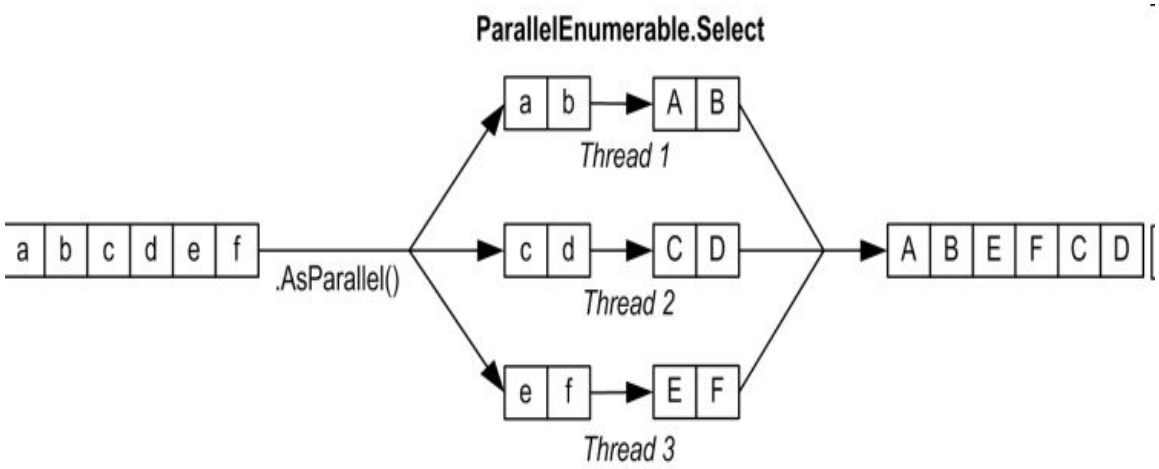


Термин "задача" здесь означает "выполняемая программа", вернее - "программа, находящаяся на стадии выполнения". В многопрограммном (многозадачном) режиме работы в стадии выполнения могут находиться несколько программ. Для каждой из них создается сегмент состояния задачи - TSS. Выполнение этих программ может производиться одним процессором в режиме разделения времени. Основным назначением механизма переключения задач является организация очередных переходов между выполняемыми программами.

http://www.internals.com/articles/protmode/introduction.htm

Межсегментные переключения задач

При переключении программ содержимое процессора (контекст программы) копируется в TSS текущей программы, а содержимое TSS целевой программы переписывается в регистры процессора. Такой переход исключает прямое взаимодействие переключаемых программ.




```
"abcdef".AsParallel().Select (c => char.ToUpper(c)).ToArray()
```

Отсутствие прямого взаимодействия программ при переходах с использованием механизма переключения задач позволяет значительно смягчить требования к доступности программ по условиям корректности переходов с изменением PL.
Но все проверки, непосредственно не связанные с PL, выполняются в том же объеме.

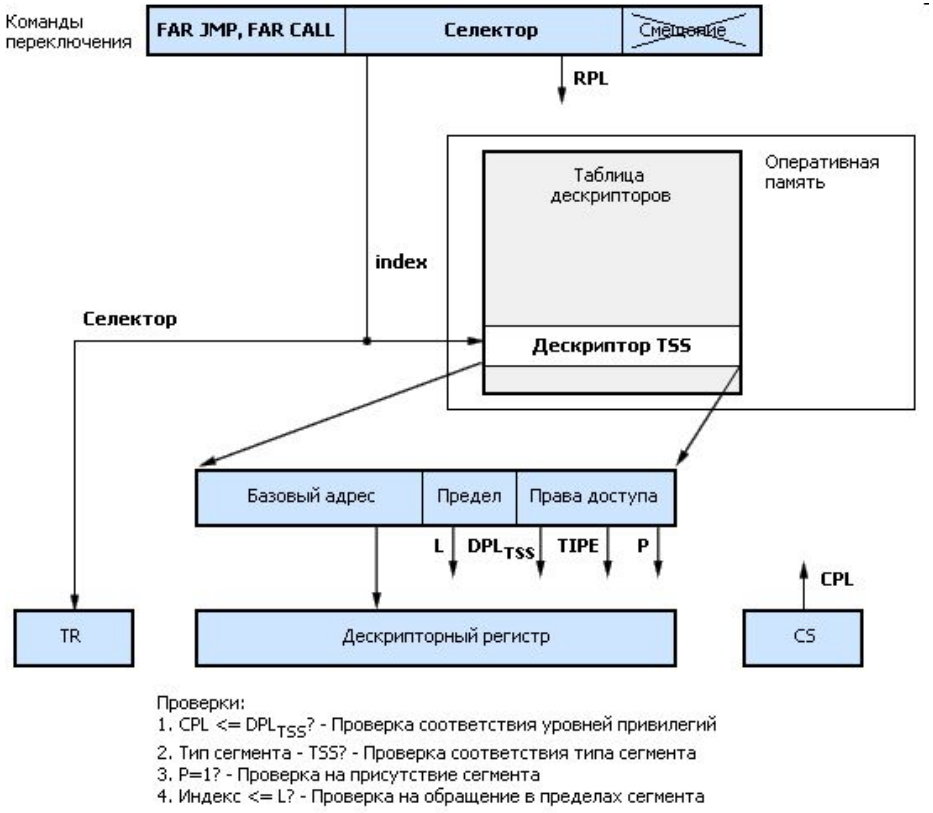
Прямое переключение задач

Прямое переключение задач - это переключение задач без использования шлюза задач. Прямое переключение задач производится командами JMP и CALL типа FAR .

В случае использования команд JMP и CALL типа FAR селекторы этих команд должны выбирать дескриптор TSS и должно выполняться условие доступа прямого переключения: уровень привилегий TSS программы цели должен быть не выше уровня привилегий текущей программы и запроса, т.е.:

$(CPL <= DPL_{TSS})$ & $(RPL <= DPL_{TSS})$

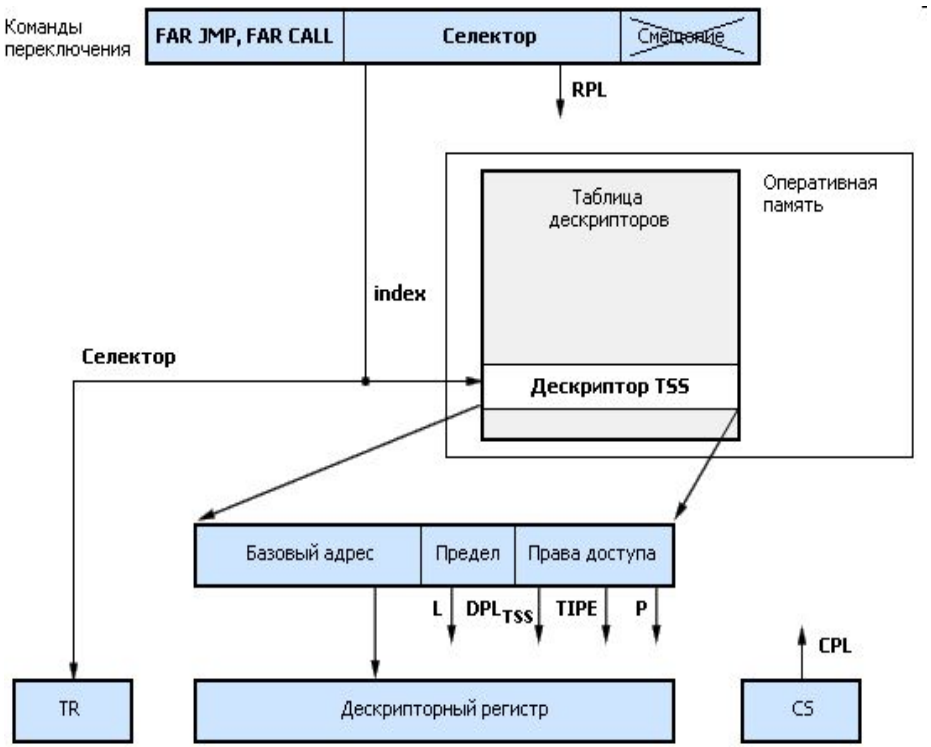
где DPL_{TSS} - уровень привилегий сегмента TSS (берется из поля уровня привилегий дескриптора TSS).



http://zzak.ru/protection/r_4_28.htm

Прямое переключение задач

- Перед обращением к таблице дескрипторов проверяется действительность селектора и соответствие индекса границам таблицы дескрипторов.
- После выборки дескриптора TSS проверяется его тип, присутствие TSS в оперативной памяти и его доступность по уровням привилегий.
- При положительных результатах проверок производится процедура переключения задач (сохранение содержимого регистров процессора в старом TSS и загрузка селектора целевого TSS в регистр TR).



Загрузка селектора целевого TSS приводит к загрузке регистров процессора контекстом целевой задачи,
т.е. к **переключению задач.**

http://zzak.ru/protection/r_4_28.htm

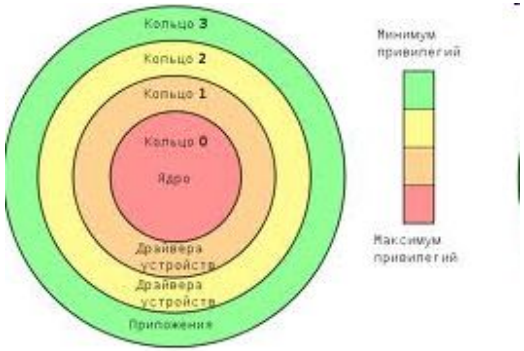
Переключение задач с использованием шлюза задач

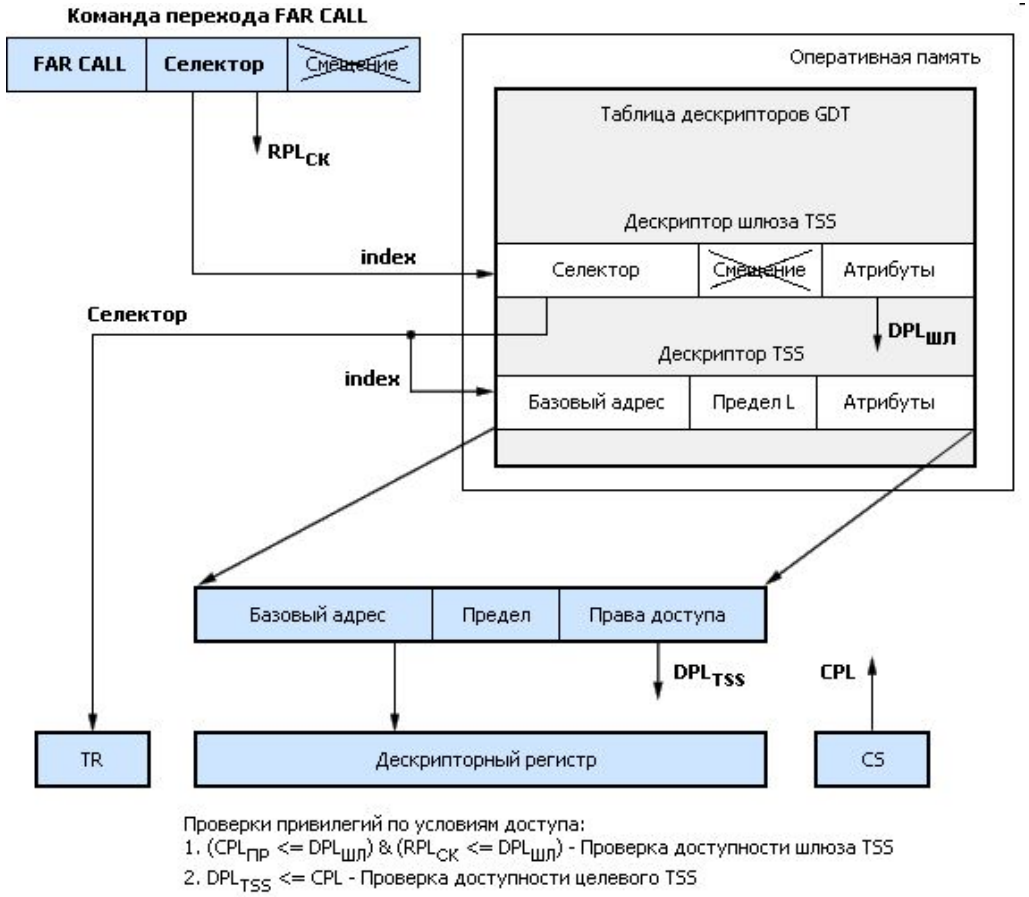
Любое переключение задач на программы более высокого уровня привилегий допускается только при помощи команд CALL типа FAR, IRET, INT n, а также в результате прерываний и ловушек. При этом использование шлюза задачи (шлюза TSS) - обязательно.

Переключение задачи по команде JMP или CALL типа FAR допускается, если уровень привилегий шлюза TSS не выше текущего уровня привилегий и уровня привилегий запроса, т.е.:

$(CPL <= DPL_{ШлTSS})$ & $(RPL <= DPL_{ШлTSS})$

где $DPL_{ШлTSS}$ - уровень привилегий шлюза задачи.



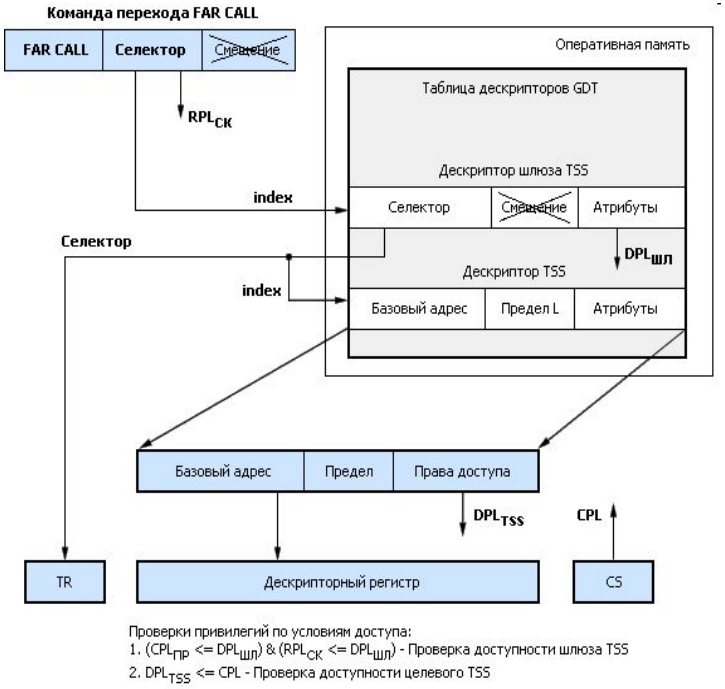


Переключение задач с использованием шлюза задач

Проверки соответствия уровней CPL, RPL и DPL_{TSS} не требуется.

Операционная система, создавая шлюзы задач в глобальной или локальных таблицах дескрипторов с низкими PL, может допускать программы нижнего уровня привилегий к разрешенным сервисным процедурам.

При переключении задач с использованием шлюза дескриптор TSS целевой задачи определяется селектором шлюза задачи.
При этом, PL шлюза TSS и дескриптора TSS могут не совпадать.
=> использование шлюзов TSS допускает любые межуровневые переключения задач.

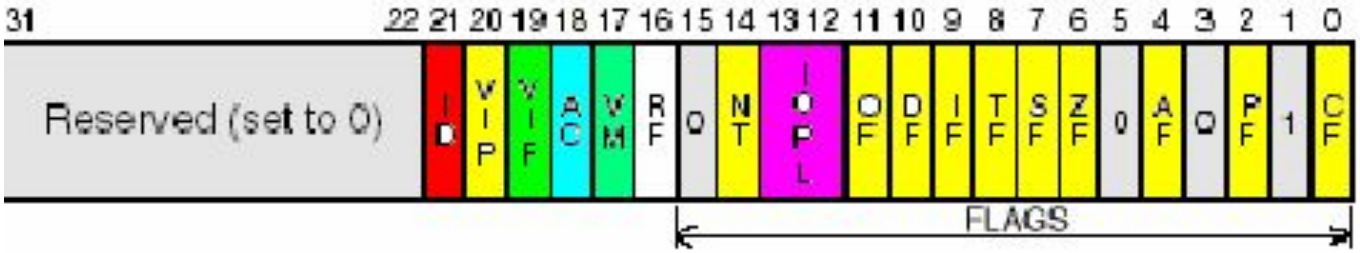


При использовании команд IRET, команды вызова прерываний, например, INT n или в случаях прерываний и ловушек, как прямое переключение задач, так и переключение задач с использованием шлюзов задач производится вне зависимости от значений, соответственно, DPL_{TSS} или $DPL_{ШлTSS}$, т.е. без проверок условий доступа по уровням привилегий.

Программное переключение

Операция переключения задач сохраняет состояние процессора (в TSS текущей задачи), и связь с предыдущей задачей (в TSS новой задачи), загружает состояние новой задачи и начинает ее выполнение. Задача, вызываемая по команде JMP, должна заканчиваться аналогичной командой обратного перехода.

- В случае использования команды CALL - возврат должен происходить по команде IRET, которая сохраняет контекст завершаемой задачи и загружает контекст прерванной.
- Алгоритм работы команды IRET в случае возврата из прерывания и в случае обратного переключения задач различен. И определяется значением флага NT (Nested Task)



Если флаг сброшен, то выполняется обычный возврат из прерывания (через стек). Если флаг установлен, то команда IRET инициирует обратное переключение задач.

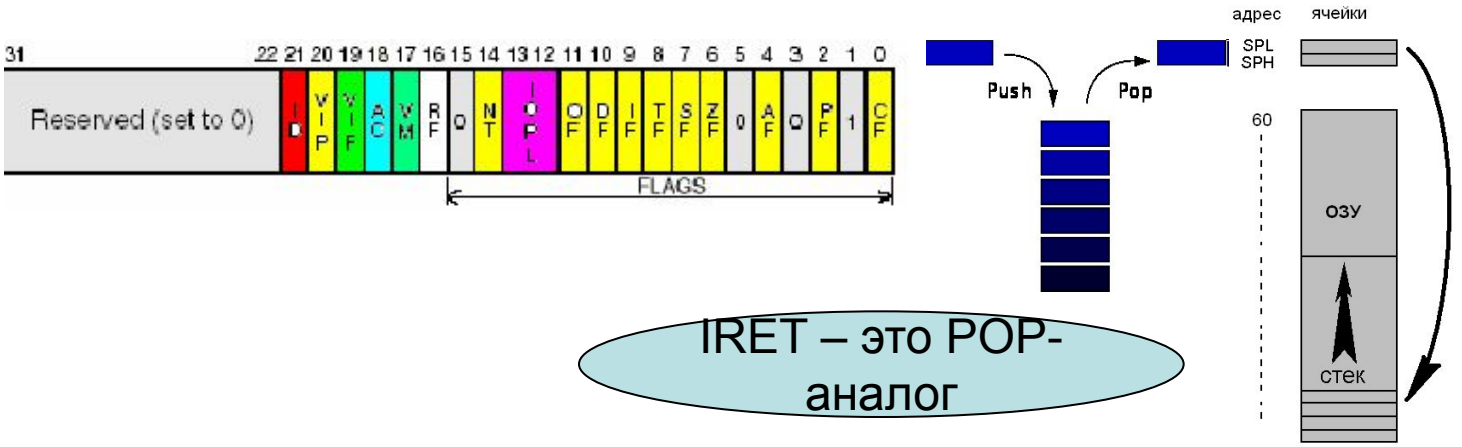
Программное переключение

- После загрузки компьютера флаг NT находится в установленном состоянии.

Однако любое аппаратное прерывание или исключение сбрасывает этот флаг, в результате чего команда IRET, завершающая обработчик, выполняется в «облегченном» варианте (возврат через стек).

- То же происходит при выполнении процессором команды программного прерывания INT.

Поскольку команда IRET восстанавливает исходное состояние регистра флагов, после завершения обработчика флаг NT снова оказывается установленным (если, конечно, он не был явно сброшен выполняемой программой).

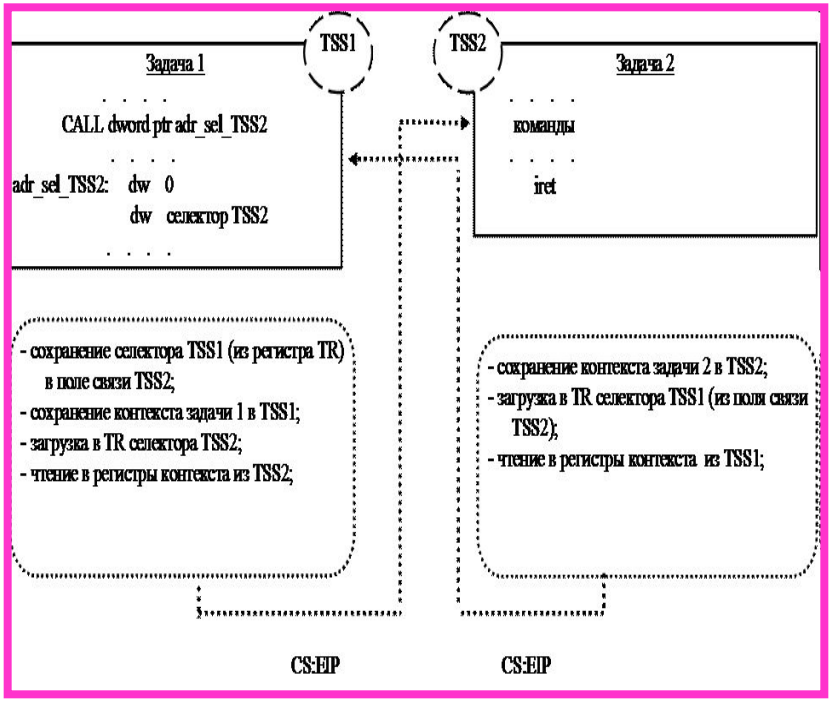


При выполнении процедуры переключения на новую задачу через шлюз или непосредственно через TSS, процессор сохраняет в TSS текущей задачи слово флагов и устанавливает в регистре флагов бит NT. Команда IRET, завершающая задачу, обнаруживает NT=1 и, вместо осуществления возврата через стек, инициирует механизм обратного переключения задач.

Последовательность действий при переключении задач

При переключении задачи процессор выполняет следующую последовательность действий:

1. Проверяет право на переключение по уровню привилегий CPL £ DPL;
2. Сохраняет в TSS исходной задачи ее контекст;
3. Загружает в регистр TR селектор TSS новой задачи;
4. В поле связи TSS новой задачи сохраняется селектор TSS исходной задачи, что обеспечивает возможность будущего обратного переключения. Считывает контекст из TSS новой задачи (в CS:IP появляется точка входа в новую задачу).
Флаг NT ¬ 1. Переключение произошло.



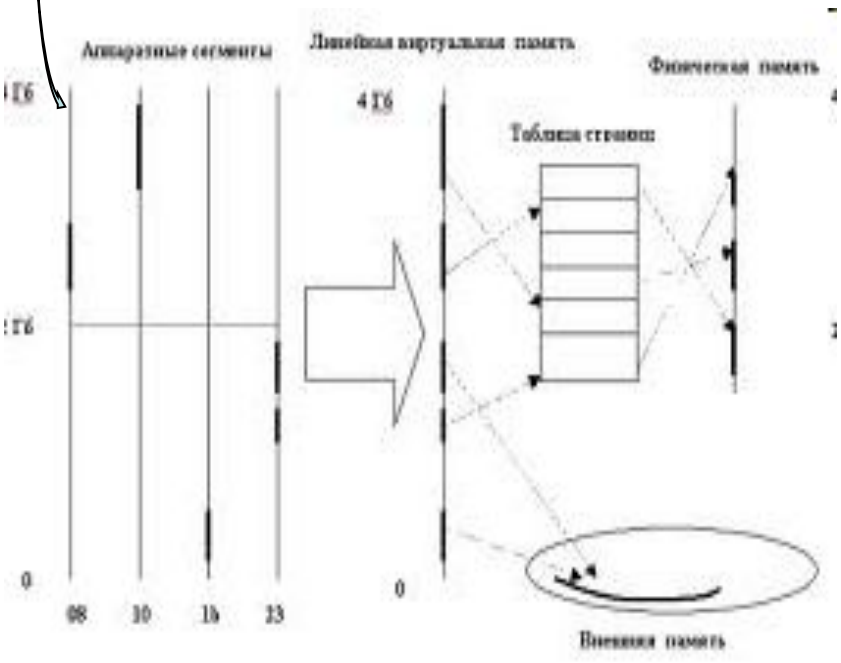
5. Когда в новой исполняемой задаче встретится команда IRET, она будет выполняться как обратное переключение задач (NT=1);
6. Контекст текущей задачи сохранится в ее TSS ;
7. В регистр TR загрузится селектор TSS исходной задачи (из поля связи TSS текущей задачи);
8. Регистры восстановится контекст исходной задачи.

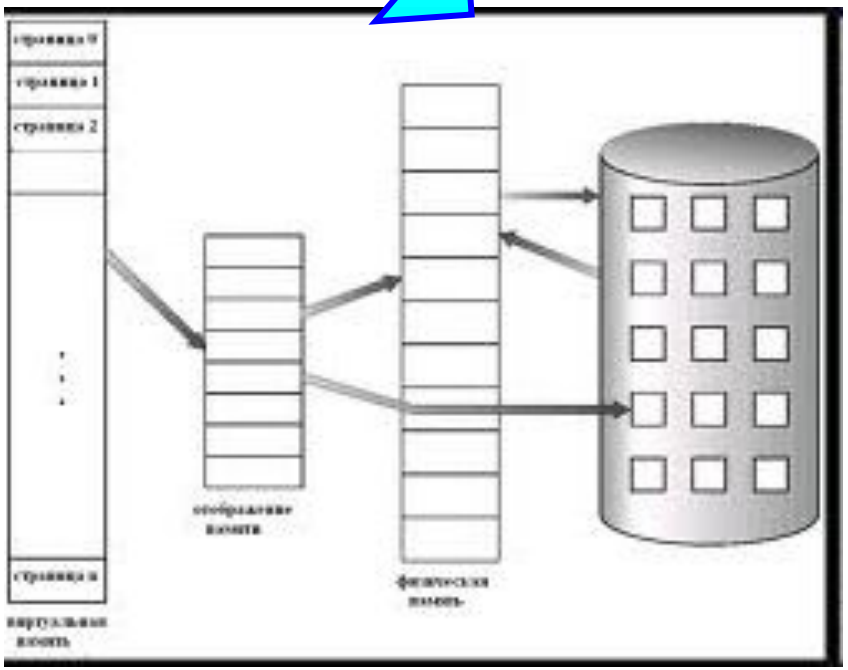
Страничное управление памятью

Страничное управление (Paging) является средством организации виртуальной памяти с подкачкой страниц по запросу (Demand-Paged Virtual Memory).

В отличие от сегментации, которая организует программы и данные в модули различного размера, страничная организация оперирует с памятью, как с набором страниц **одинакового размера**.

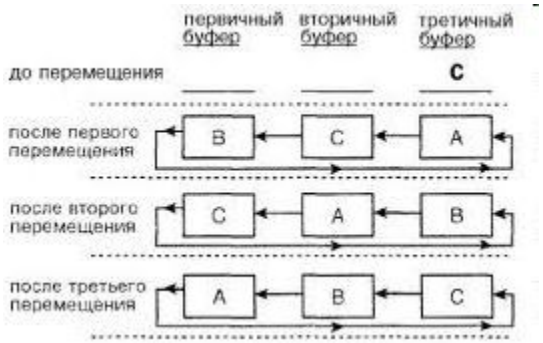
В момент обращения страница может присутствовать в физической оперативной памяти, а может быть выгруженной на внешнюю (дисковую) память.

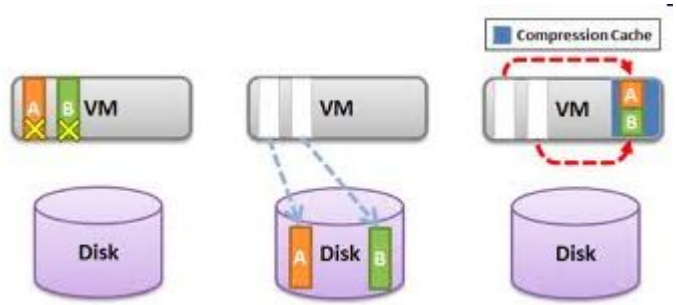




Страничное управление памятью

При обращении к выгруженной странице памяти процессор вырабатывает исключение #PF — отказ страницы, а программный обработчик исключения (часть ОС) получит необходимую информацию для **свопинга** — «подкачки» отсутствующей страницы с диска.





Страницы не имеют прямой связи с логической структурой данных или программ.

В то время как селекторы можно рассматривать как логические имена модулей кодов и данных, страницы представляют части этих модулей.

- Учитывая обычное свойство локальности (близкого расположения требуемых ячеек памяти) кода и ссылок на данные, в оперативной памяти в каждый момент времени следует хранить только небольшие области сегментов, необходимые активным задачам.
- Эту возможность (а следовательно, и увеличение допустимого числа одновременно выполняемых задач при ограниченном объеме оперативной памяти) как раз и обеспечивает

страничное управление памятью.

Страничное управление памятью

!

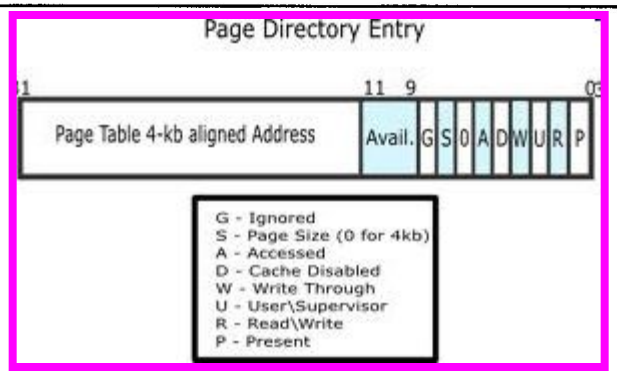
Страничная переадресация может использоваться с сегментацией без каких-либо требований по согласованию границ сегментов и страниц. Однако от разумности распределения сегментов по страницам будет зависеть производительность компьютера, поскольку подкачка страниц занимает значительное время.

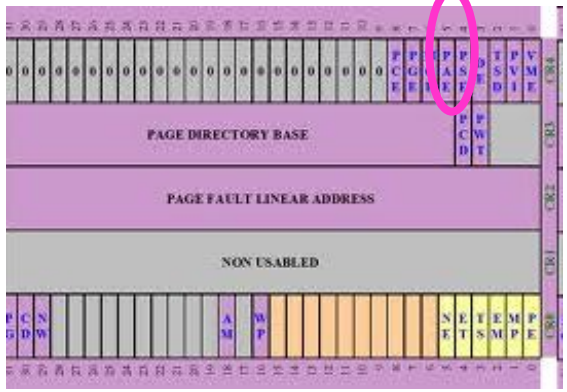
Поскольку имеется возможность расширения физического адреса до 36 бит (64 Гбайт), при котором могут использоваться страницы размером 4 Кбайт и 2 Мбайт.

Режимы страничной переадресации

CR0.PG	CR4.PAE	CR4.PSE	PDE.PS	Размер страницы	Разрядность физического адреса, бит
0	x	x	x	Трансляция запрещена	32
1	0	0	x	4 Кбайт	32
1	0	1	0	4 Кбайт	32
1	0	1	1	4 Мбайт	32 (PSE) 36 (PSE-36)
1	1	x	0	4 Кбайт	36 (PAE)
1	1	x	1	2 Мбайт	36 (PAE)

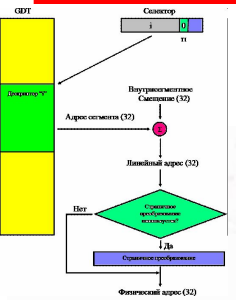
Режимом страничной переадресации управляют биты PG в регистре CR0, флаги PAE и PSE в CR4 и бит размера страницы PS в PDE – элементе каталога страниц





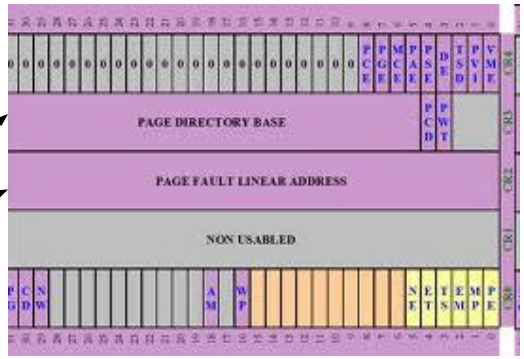
Базовый механизм страничного управления

Базовый механизм страничного управления использует двухуровневую табличную трансляцию линейного адреса в физический.

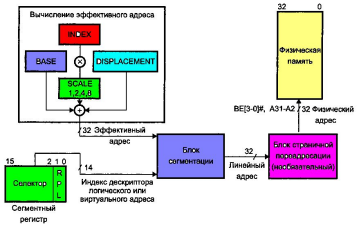


Механизм имеет три части:

1. Каталог страниц (Page Directory),
2. Таблицы страниц (Page Table)
3. Собственно страницы (Page Frame).

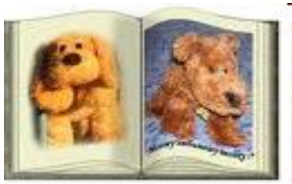


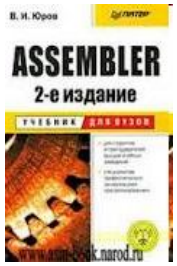
- Механизм включается установкой бита PG=1 в регистре CR0.
- Регистр CR2 хранит линейный адрес отказа (Page Fault Linear Address) — адрес памяти, по которому был обнаружен последний отказ страницы.
- Регистр CR3 хранит физический адрес каталога страниц (Page Directory Physical Base Address). Его младшие 12 бит всегда нулевые (каталог выравнивается по границе страницы).



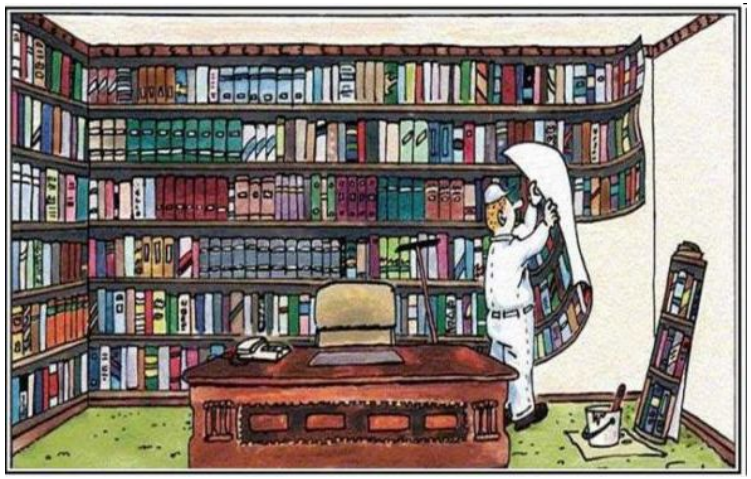
Используемая литература:

- Книга «*Assembler*. Учебник для ВУЗов», автор Юров
- Книга «Процессоры *Pentium4, Athlon* и *Duron*», авторы Михаил Гук, Виктор Юров
- Книга «Модернизация и ремонт ПК»,автор Скотт Мюллер
- Книга «Архитектура ЭВМ», автор Танненбаум
- Книга «Организация ЭВМ», автор Хамахер



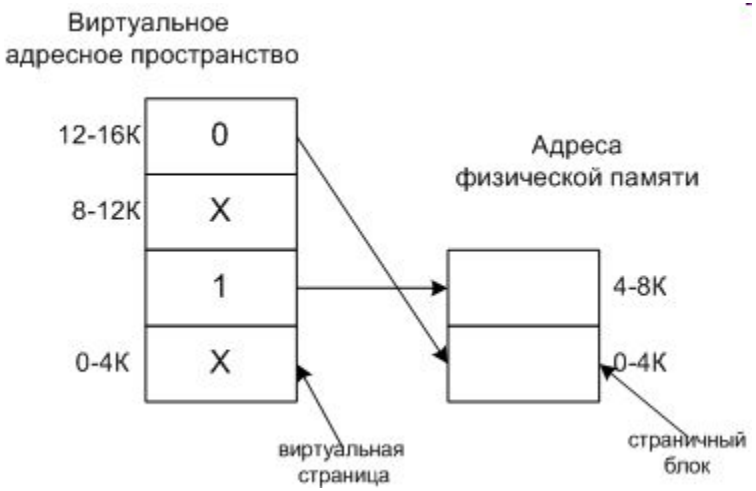


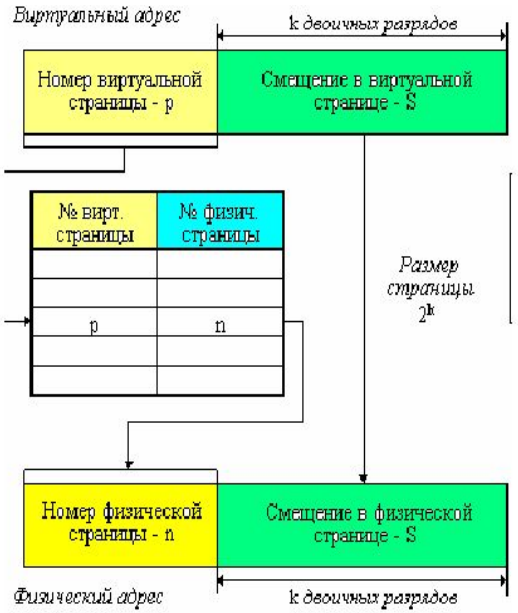
ЭВУ
SUPER!!!



В следующей серии

Виртуальная память





Основные используемые Интернет-ресурсы:

http://x86.migera.ru/text/glava_5.html
http://dims.karelia.ru/x86/multi_3.shtml
http://zzak.ru/protection/4_3_5.htm
http://sasm.narod.ru/docs/pm/pm_tss/chap_4.htm
http://www.internals.com/articles/protmode/introduction.htm
http://board.sysbin.com/viewtopic.php?t=1997
http://wiki.osdev.org/GDT_Tutorial#Flat_Setup
http://habrahabr.ru/tag/переключение%20задач/