- Плоская модель памяти -

Процессор, имеющий n линий в шине адреса может адресоваться к 2^n ячейкам памяти. Каждая из 2^n комбинаций выбирает конкретную ячейку памяти. Такая модель памяти называется плоской (flat) или сплошной. Программы могут обращаться по адресам 0--2^n -1, что составляет линейное адресное пространство.

- Сегментная модель памяти -

Сегментация – такая организация памяти, при которой используется много независимых линейных адресных пространств, называемых сегментами.

Для программы адресное пространство разделено на сегменты (блоки смежных адресов) и программа может обращаться к данным только в этих сегментах. Внутри сегмента применяется линейная адресация и программа может обращаться к ячейкам 0, 1, 2 и т.д. Такая адресация осуществляется относительно начала сегмента и физический адрес, ассоциируемый с программным адресом, скрыт от программиста.

Программы обычно разделяются логически на сегменты кода, данных, стека. При этом упрощается изолирование программ друг от друга в мультизадачной среде.

Достоинства сегментации

- обеспечивает модульность программ
- придает коду и данным перемещаемость
- реализуют управление виртуальным адресным пространством, защиту памяти, многозадачность.

Причины сегментации:

- разнородность используемых структурных элементов с данными.
- проблема адресации

- Сегментация памяти в процессорах 8086 -

Разрядность адреса n=20.

Линейное адресное пространство 1Мбайт

Разрядность внутренних регистров, используемых, в том числе, и для вычисления адресов, 16.

Сегмент – блок сегментных ячеек памяти с максимальным размером 64к и начальным адресом, находящимся на границе параграфа.

Параграф – блок смежных ячеек из 16 байт. Граница параграфа имеет линейный адрес кратный 16, т.е. содержащий 4 нулевых младших разряда.

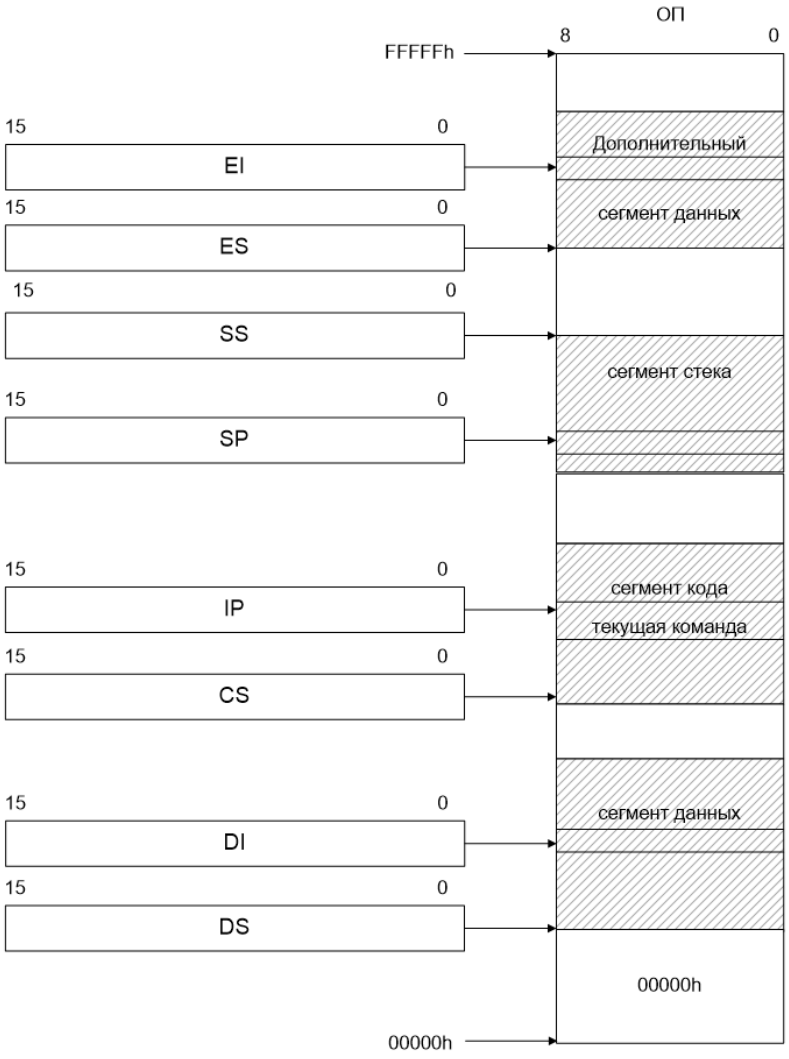
Для задания старших 16 разрядов начальных адресов сегментов используются сегментные регистры.

CS (code segment) – определяет сегмент кода.

DS (data segment) – определяет сегмент данных.

SS (stack segment) – определяет сегмент стека.

ES (extra segment) – определяет дополнительный сегмент данных.



Логический адрес сегмент: смещение
IP (Instruction Pointer) определяет текущую команду в сегменте кода
SP (Stack Pointer) определяет вершину адреса
Линейный адрес = (сегмент)*16 +смещение

Преобразование логического адреса в линейный в процессоре 8086.

линейный адрес = (сегмент)*16 +смещение

Пример

CS= 100h – определяет базовый адрес сегмента кода 1000h.

IP= 08h – определяет смещение текущей команды

линейный адрес текущей команды — адрес

$$\begin{array}{r} 1000h \\ +\ 80h \\ \hline 1008h \end{array}$$

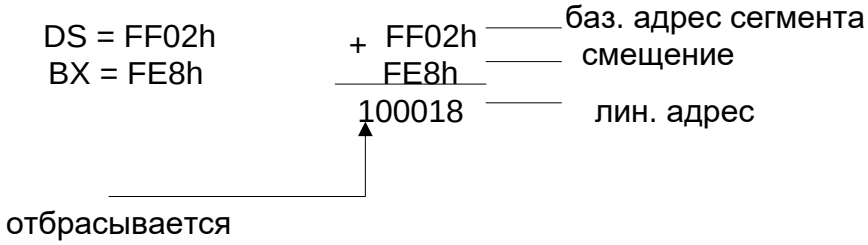
Преобразование логического адреса в линейный однозначно.

Обратное преобразование не однозначно.

1000h = 100h:08h = 0FFh:18h = 0FFh:28h…

Каждому линейному адресу соответствует 4к логических адресов.

Эффект «закругления» или «заворачивания» адреса.



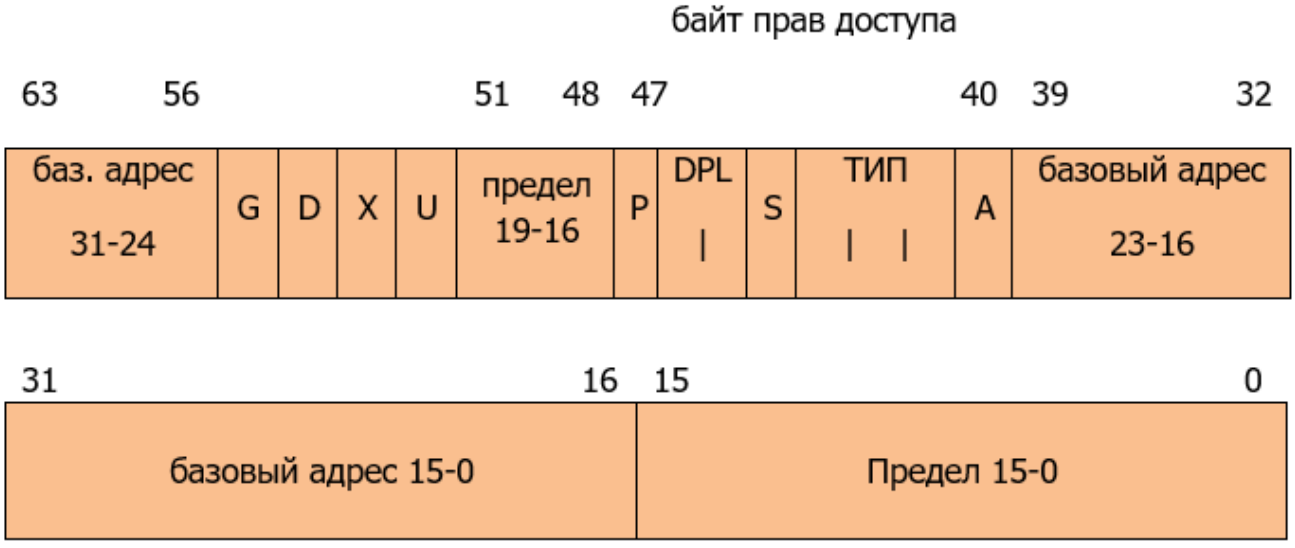
В процессорах, начиная с 286, этот эффект отсутствует, т.к. шина адреса имеет большую разрядность. И появляется возможность в реальном режиме адресовать сегмент размером 64к-16 за пределами 1Мбайта. Эта память называется **himemory**. Линейный адрес совпадает с физическим для процессора 8086.

- Недостатки сегментации в процессоре 8086 –

1. Сегменты имеют всего лишь два атрибута, начальный адрес и максимальный размер 64кбайта. Никаких аппаратных средств контроля правильности использования сегментов нет.
2. Размещение сегментов памяти произвольно. Они могут перекрываться частично, или полностью, или не иметь общих частей.
3. Программа может обраться к любому сегмента для записи, и считывания данных, и для выбора команд. Может нарушить область системных данных.
4. Нет никаких препятствий для обращения даже к физически не существующей памяти.

- Сегментация памяти в защищенном режиме -

Каждый сегмент характеризуется специальной 8-байтовой структурой данных, называемой дескриптором сегмента. (segment descriptor).



Базовый адрес 31-0 определяет местоположение сегмента в ОП.
Предел 19-0 – указывает на последний адресуемый байт сегмента.
G – бит гранулярности. Определяет единицу измерения поля предел.
G=0 (байтная гранулярность)
П=1 (страничная гранулярность)

1 страница = 4 кбайта = 212байт.

Пример предел = 10h

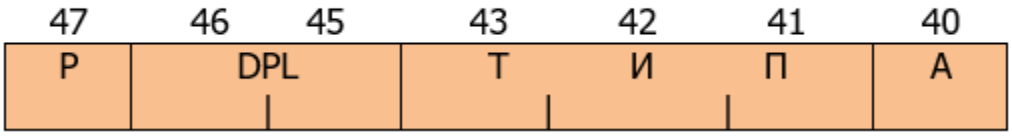
G=0

Размер сегмента равен 17 байтам, что соответствует смещению в сегмент 0,1,2,…,10h.

G=1

Размер сегмента равен 17 страницам. Последний адресуемый байт в сегменте 10FFFh.

Байт прав доступа



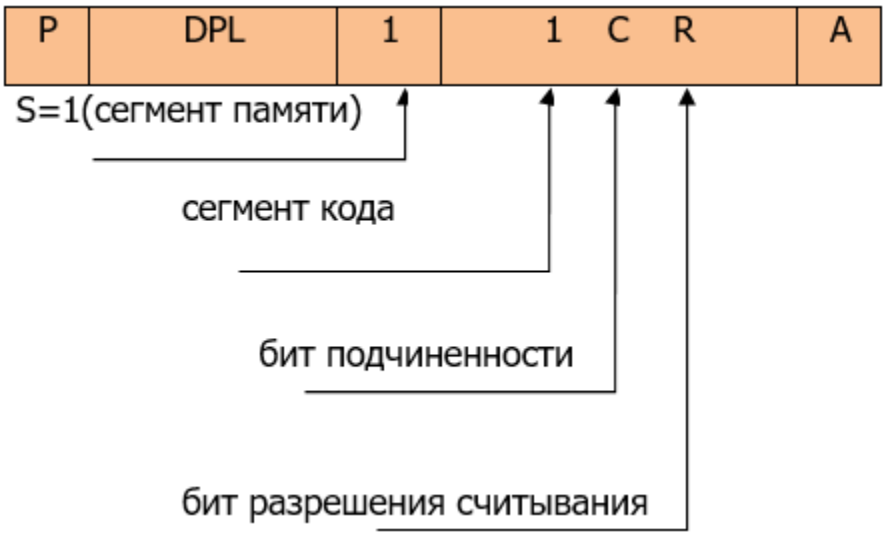
P (present) – бит присутствия. P=1 указывает на присутствие сегмента в ОП. P=0 указывает на то, что сегмент находится на дисковой памяти DPL (Descriptor Privilege Level) – уровень привилегий дескриптора. Определяет уровень привилегий той области памяти, которую описывает дескриптор.

S – бит, показывающий, что описываемый дескриптором сегмент является сегментом памяти (S=1) или системным объектом(S=0).

A (accessed) – бит доступа. Установка этого бита свидетельствует об обращении к соответствующему сегменту.

DPL (Descriptor Privilege Level) – определяет уровень привилегий той области памяти, которая описывает дескриптор в диапазоне от 00 до 11.

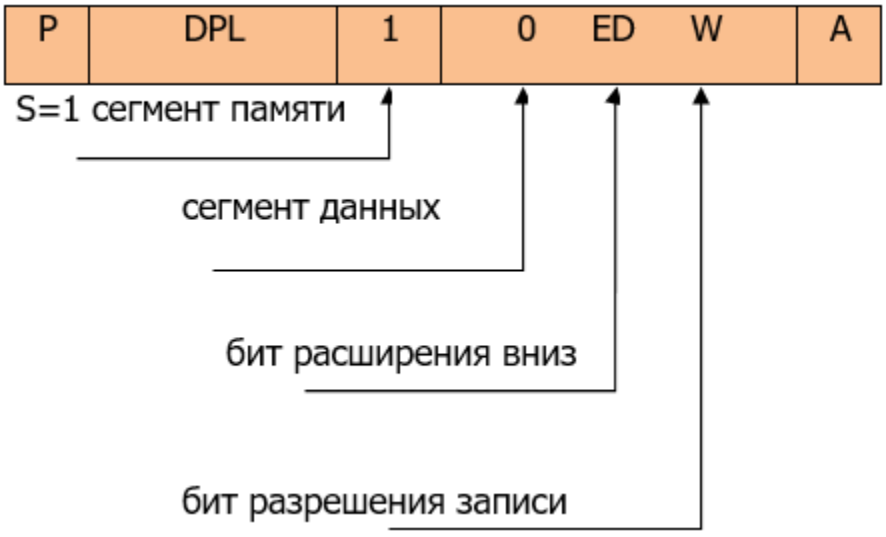
Поле ТИП определяет целевое использование сегмента.



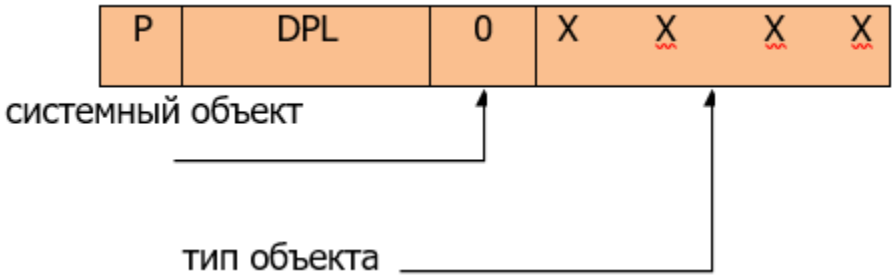
100 – сегмент кода, разрешено только выполнение.

101 – сегмент кода, разрешено выполнение и чтение.

110 – подчиненный сегмент кода, разрешено выполнение.
111 – подчиненный сегмент кода, разрешено выполнение и чтение.



000 – сегмент данных, разрешено только чтение.
001 – сегмент данных, разрешено чтение и запись.
010 – сегмент стека, разрешено только чтение (на практике не используется).
011 – сегмент стека, разрешено чтение и запись.



Если бит S=0, то бит A расширяет поле тип до 4 разрядов.
Бит D (default size) – размер по умолчанию.
D=0 операнды имеют размер 16 разрядов.
D=1 операнды имеют размер 32 разряда.

Бит U предназначен для использования системными программистами по их усмотрению.
Бит X зарезервирован для последующих процессоров.

- Дескрипторные таблицы -

Существуют дескрипторные таблицы трех типов.

GDT – глобальная дескрипторная таблица. Содержит дескрипторы общесистемных объектов. Ее местоположение в физической памяти определяет регистр GDTR.

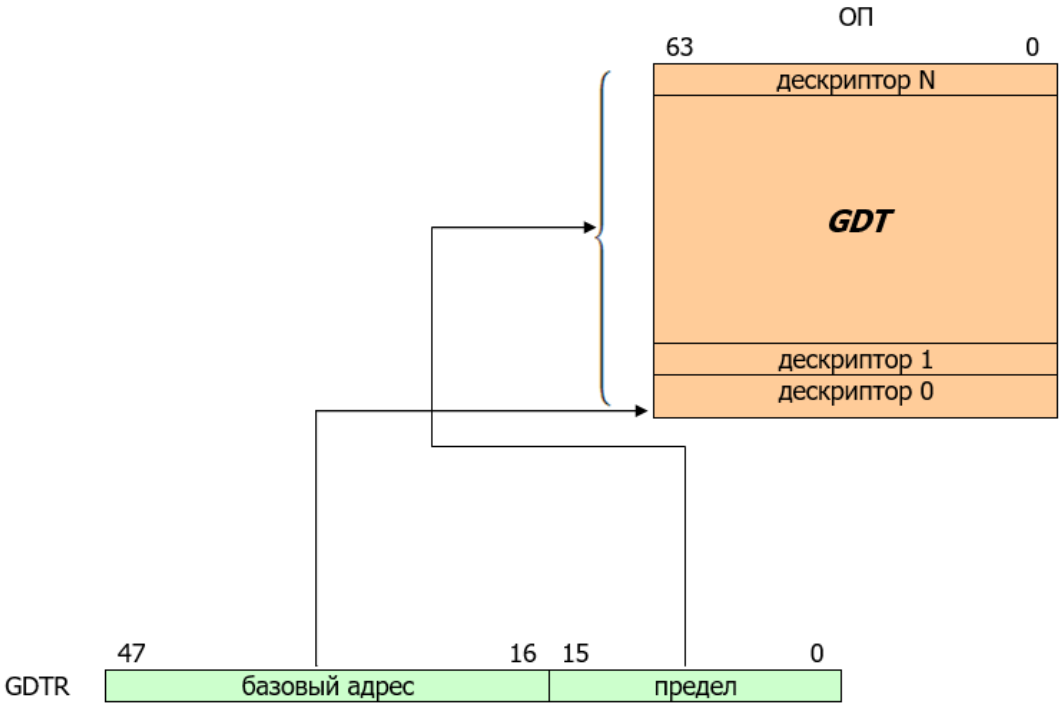
IDT – дескрипторная таблица прерываний. Предназначена для вызова процедур обработки прерываний и исключений. Местоположения IDT в физической памяти определяет регистр IDTR.

LDT – локальная дескрипторная таблица. Содержит дескрипторы сегментов, доступные только для одной конкретной задачи. Для локализации LDT используется 16разрядный регистр LDTR, который косвенно через GDT определяет LDT. В мультизадачной среде таблиц LDT может быть множество. Активной является только одна таблица, селектор которой содержится в LDTR.

Для работы с регистрами дескрипторных таблиц используются команды:

LGDT mem48
LIDT mem48
LLDT reg16/ mem16

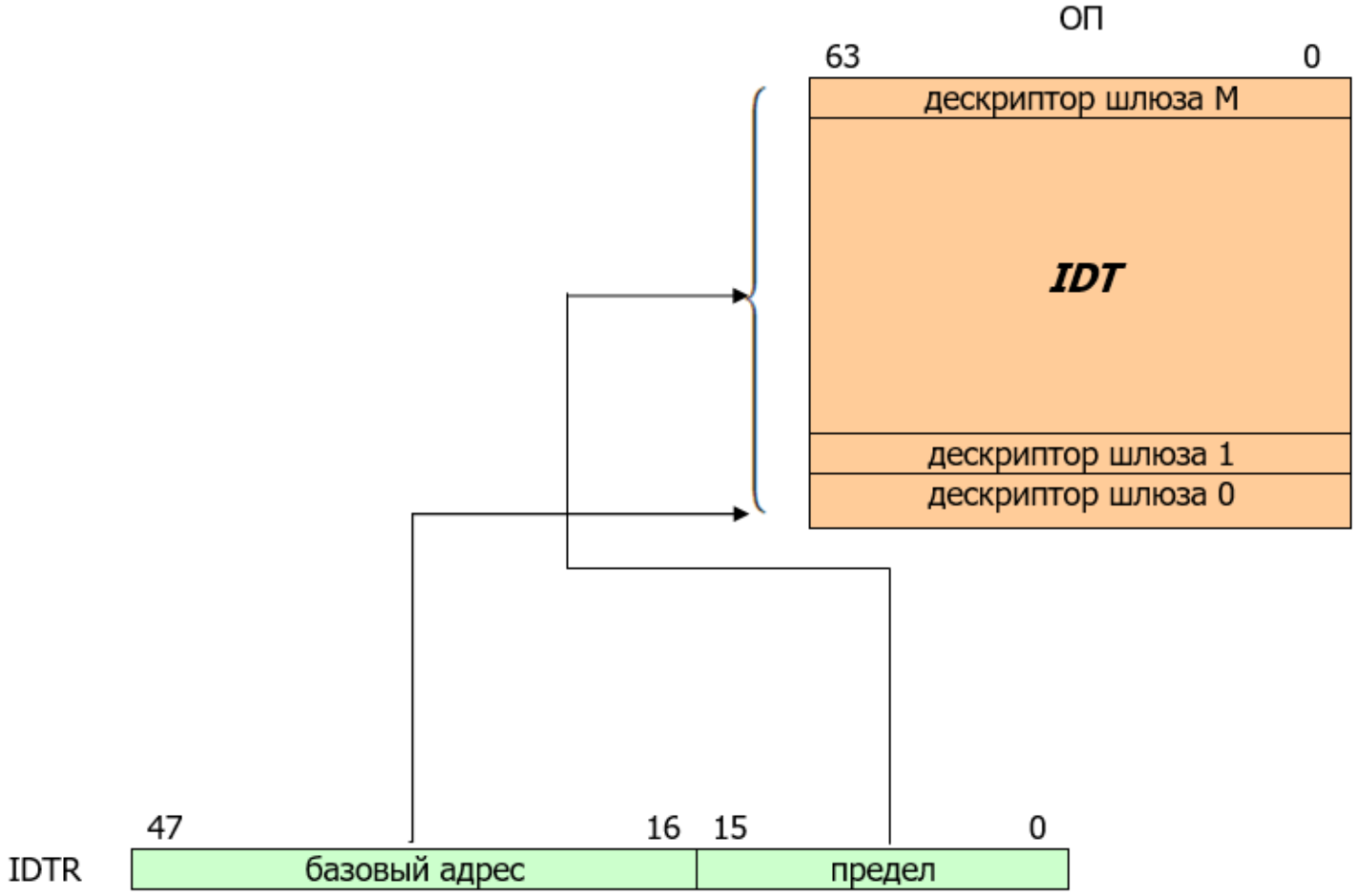
SGDT mem48
SIDT mem48
SLDT reg16/ mem16



- Регистр IDTR -

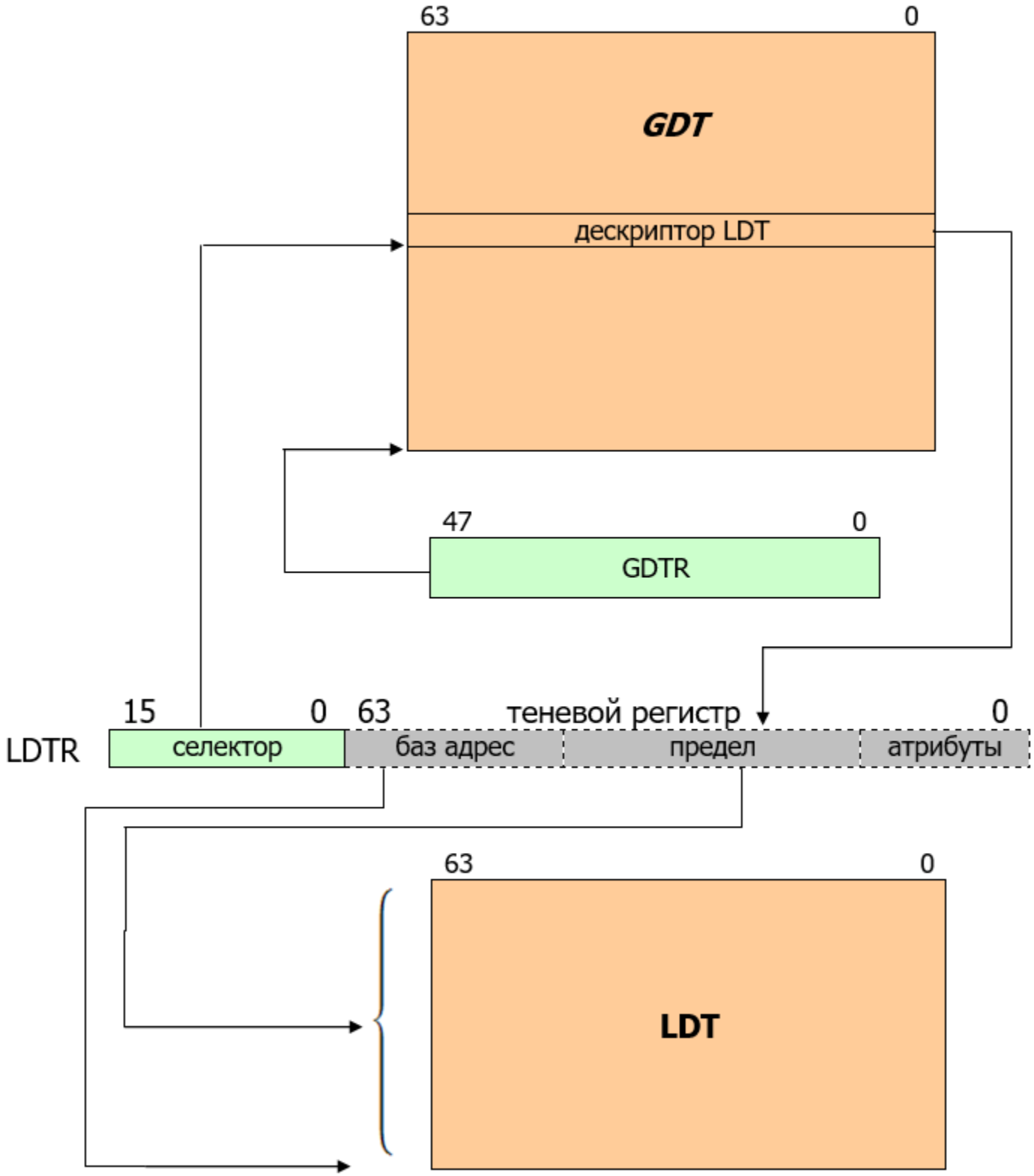
IDTR (Interrupt Descriptor Table Register).
Определяет местоположение в ОП дескрипторной таблицы прерываний.

Таблица IDT содержит дескрипторы шлюзов, необходимые для выполнения процедур обработки прерываний(исключений).



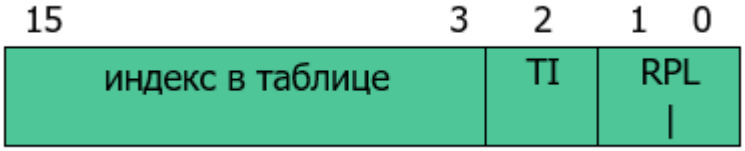
- Регистр LDTR -

Определяет косвенно через GDT местоположение в оперативной памяти LDT.
Локальная дескрипторная таблица содержит дескрипторы сегментов(кода, стека, данных), относящиеся к текущей задаче.
Каждая задача имеет свою LDT.



- Селекторы сегментов -

Селекторы загружаются в сегментные регистры и косвенно через дескрипторные таблицы определяют сегменты.



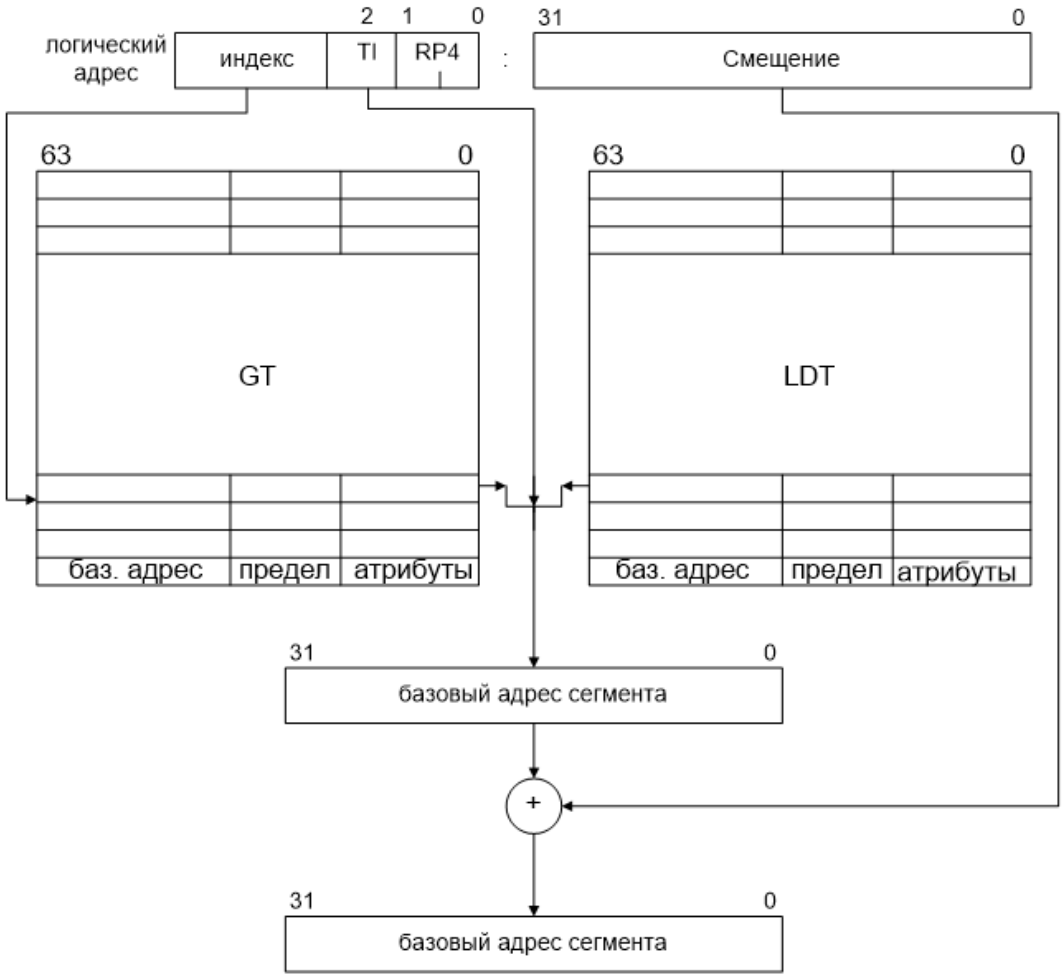
RPL (Request Privilege Level) – запрашиваемый уровень привилегий. Используется в механизме защиты по привилегиям.

TI (Table Indicator) – показывает в какой дескрипторной таблице находится дескриптор. При TI=0 – дескриптор в GDT. При TI=1 – дескриптор в LDT.

Индекс (порядковый номер) определяет выбираемый дескриптор в таблице.

Базовый адрес дескриптора = базовый адрес таблицы + индекс * 2^3.

- Формирование линейного адреса –



Пример вычисления линейного адреса в защищенном режиме.

mov eax, [ecx][esi+20h]

32битный операнд, адресуемый в сегменте данных с помощью базово-индексной адресации со смещением, пересылается в регистр eax.

Пусть DS = 00..0110xxB

Содержимое DS указывает на то, что дескриптор сегмента данных находится в GDT (TI=0) под номером 3. (011)

Шаги выполнения команды:

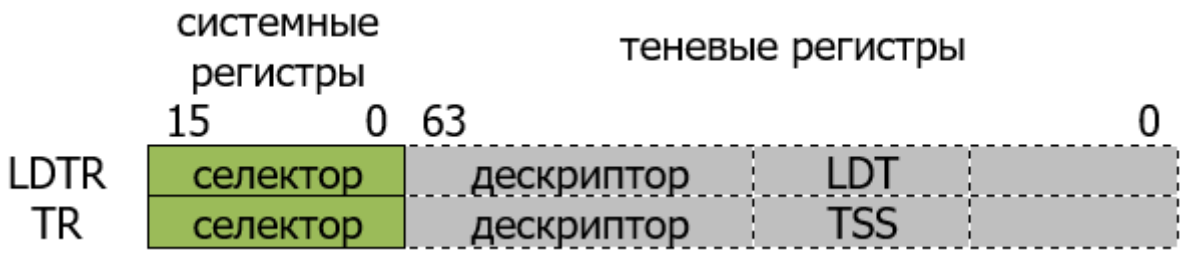
1. Вычислить эффективный адрес EA=(ECX)+(ESI)+20h
2. Выбрать третий дескриптор из GDT и считать в процессор дескриптор сегмента данных.
3. Вычислить линейный адрес операнда путем сложения базового адреса сегмента из дескриптора и эффективного адреса.
4. Обратиться к ОП по линейному адресу и передать двойное слово в EAX

Недостаток. Чтобы прочитать 4 байта из ОП приходится дополнительно прочитать 8-байтный дескриптор из ОП. Если TI=1, придется дополнительно прочитать 8-байтный дескриптор из ОП. Если TI=1, придется дополнительно прочитать еще и дескриптор из LDT из GDT.

- Кэширование дескрипторов в процессоре -

Чтобы всякий раз при обращении к сегменту не считывать из ОП дескриптор, дескрипторы запоминаются в теневых (shadow) регистрах, ассоциируемых с каждым из шести сегментных регистров.

Таким же образом используются регистры LDTR и TR.



В дескрипторах LDT и TSS поле S=0поле ТИП = 0010b = 2h- для дескриптора LDT поле ТИП =M0B0h, где M – определяет разрядность процессора, B – бит занятости сегмента.

Перед загрузкой селектора в сегментный регистр осуществляется проверка на выход за пределы дескрипторной таблицы, контролируются привилегии и правильность использования сегмента.

Команды работы с дескрипторами

lar reg16/32, reg16/mem16 – (load access right)

загрузка прав доступа (байт AR и байт с битами GD X U) в регистр получатель

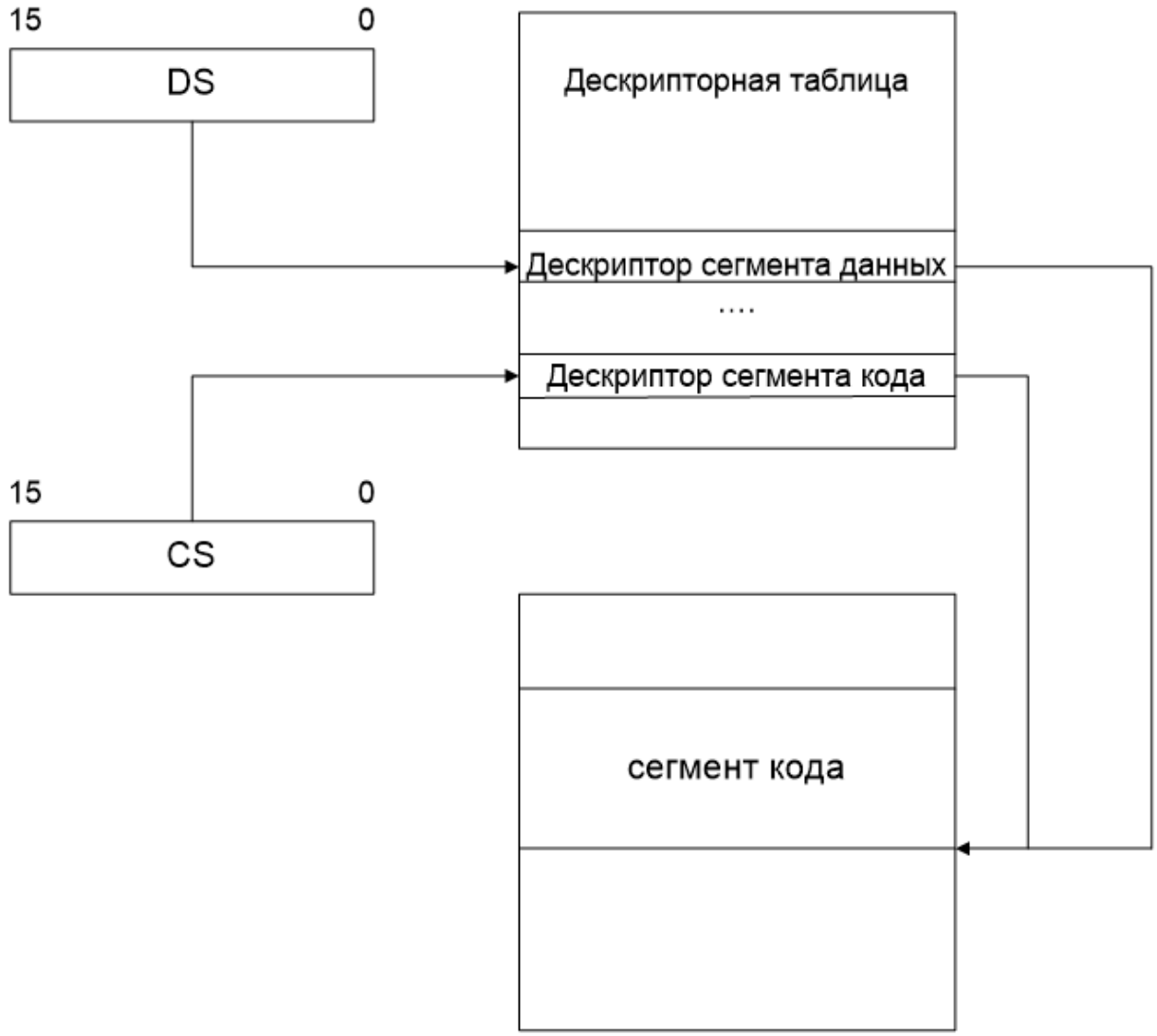
lsl reg 32, reg 16/mem16 – (load segment limit) загрузка предела сегмента

verr reg 16/mem16 – (verify for read) проверка допустимости чтения из сегмента.

verw reg16/mem16 – (verify for write) проверка допустимости записи в сегмент.

- Дескрипторы псевдонимы -

Одну и ту же область памяти можно описать с помощью нескольких дескрипторов. В этом случае дескрипторы называются псевдонимы.



Используя дескриптор-псевдоним, можно модифицировать код программы в сегменте кода.

Достоинства сегментной памяти в защищенном режиме:

- Возможность реализации виртуальной памяти.
- Реализация механизма защиты по привилегиям. Позволяет строить надежные защищенный системы.
- Автоматическое обнаружение и обработка программных ошибок, в связи с неверными указателями или нарушениями пределов.

Недостатки сегментации памяти:

- Необходимость выполнения служебных функций при загрузке селекторов в сегментные регистры.
- Ограниченные возможности по созданию пределов сегментов для сегментов, больших 1 Мбайта.

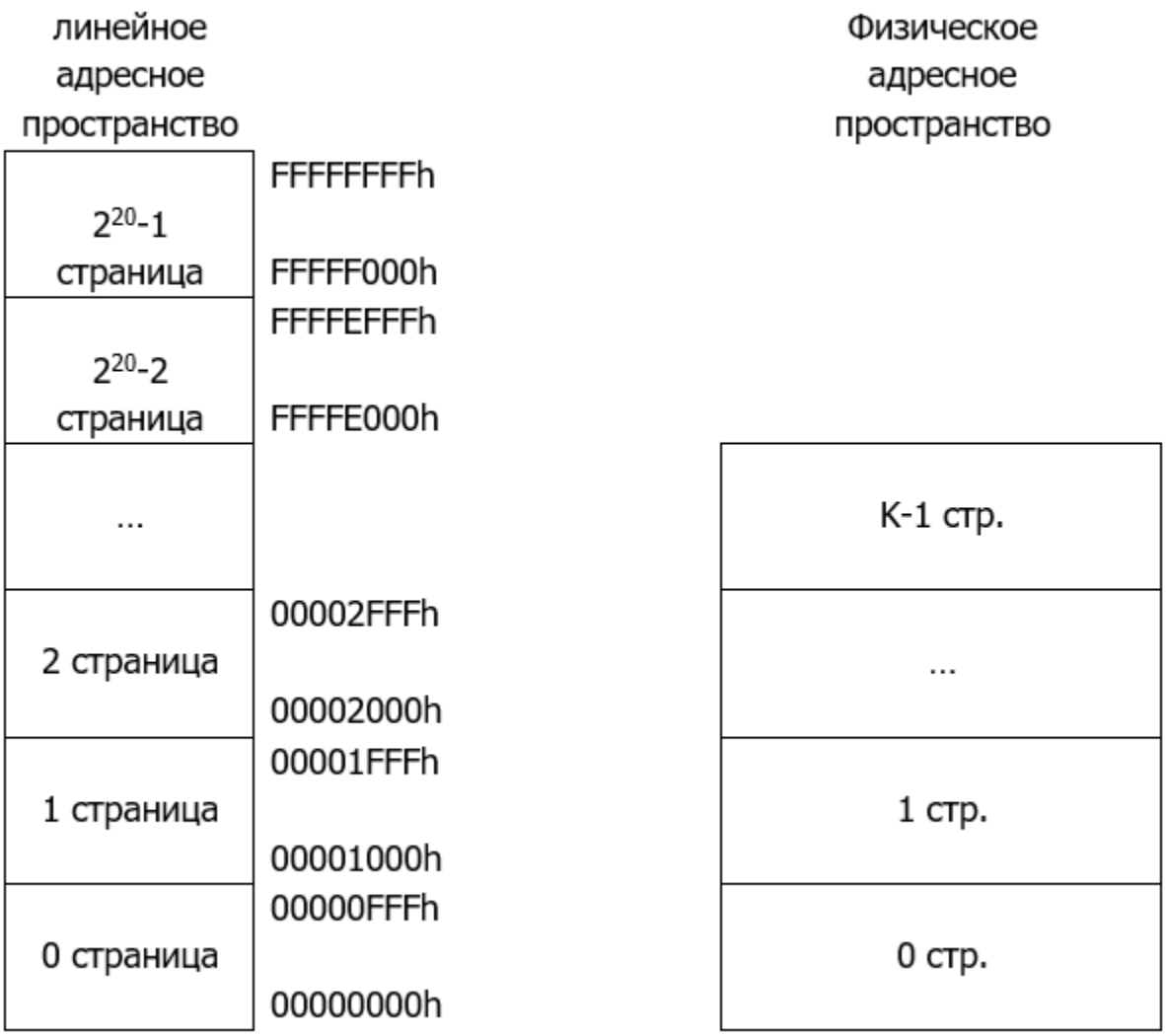
- Страничная организация памяти -

Линейное адресное пространство разбивается на страницы фиксированного размера (4 кбайта, 2 Мбайта, 4 Мбайта). Причем страницы выравниваются на границе страницы.

Физическое адресное пространство разбивается на такие же, выровненные страницы. Так как линейное адресное пространство намного больше пространства физически реализованной памяти, то только часть его страниц находится в физической памяти. Отсутствующие страницы обычно хранятся во внешней памяти, которой чаще всего служат накопители на жестких магнитных дисках.

Понятие виртуальной памяти состоит в том, что прикладная программа не касается процесса страничного преобразования адреса и может использовать все адресное пространство. Процессор формирует особый случай неприсутствия (страничное нарушение), когда программа обращается к странице, отсутствующей в физической памяти. При обработке этого особого случая операционная система загружает затребованную страницу из внешней памяти.

- Страничная память: виртуальная и физическая -



Линейное адресное пространство 2^{32} байт = 4Гбайта

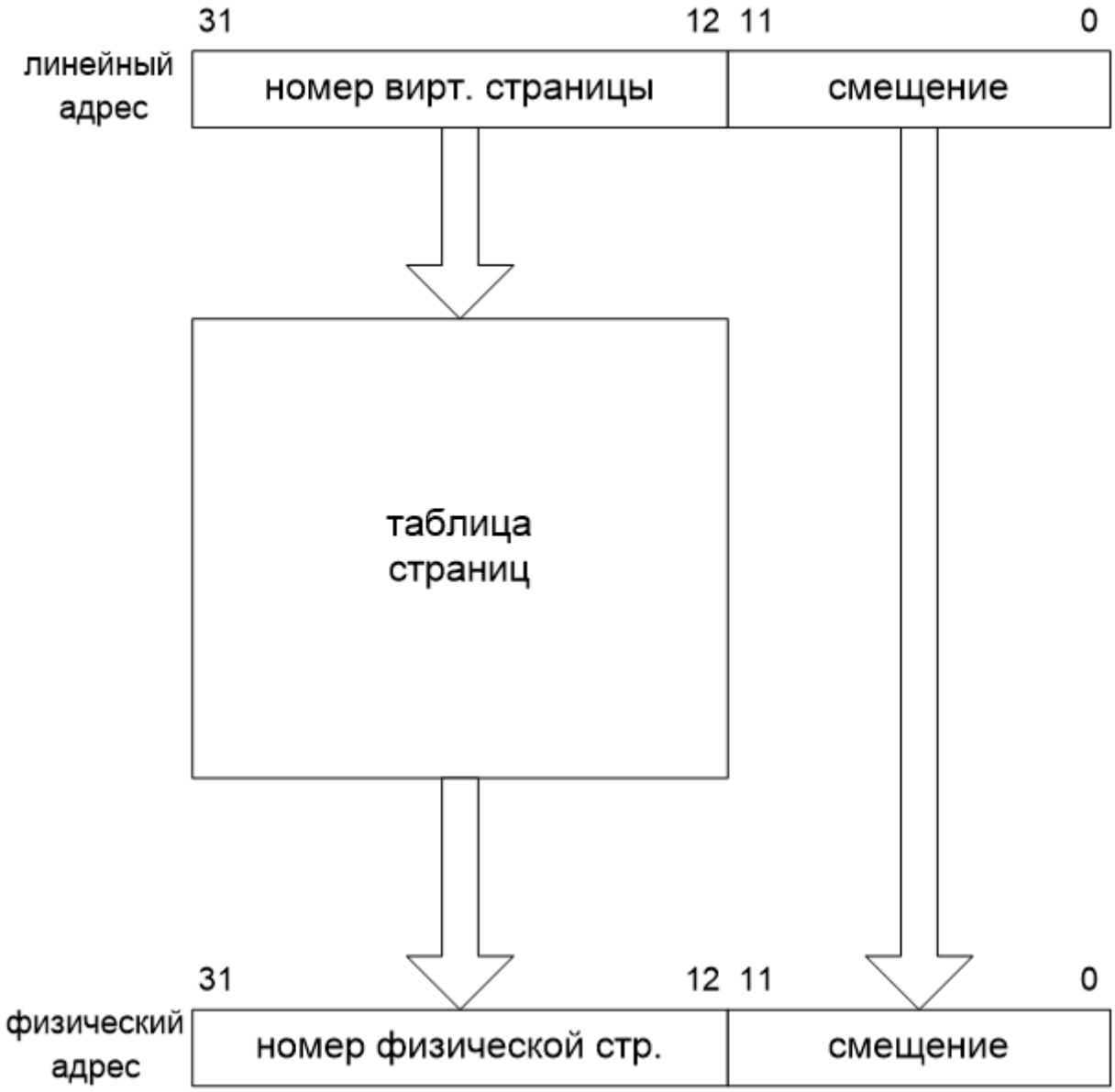
Размер страницы 4 кбайта = 2^{12}байт

Число страниц в виртуальной памяти $2^{32}/2^{12}=2^{20}$

Число страниц в физической памяти K (K«2^{20})

Физическое адресное пространство K*2^{12}байт

- Одноэтапное преобразование линейного адреса в физический –



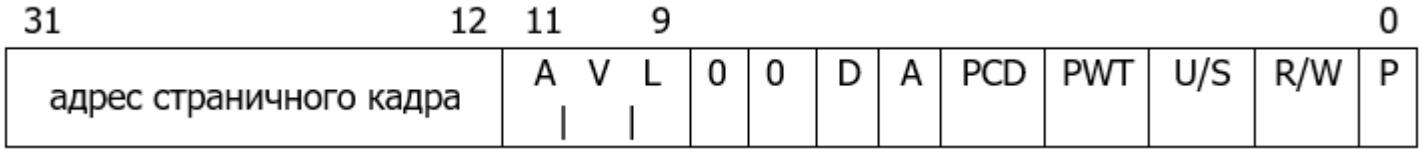
Размер таблицы страниц 2^{20}-элементов

Каждый элемент должен содержать базовый адрес физической страницы (20 разрядов) и дополнительную информацию (всего 4 байта).

С учетом этого размер таблицы 4 мегабайта.

В мультизадачной среде такая таблица может потребоваться для каждой задачи, что недопустимо.

- Формат элементов таблиц страниц -



P(present) – бит присутствия. Показывает, находится ли страница в физической памяти.

Когда P=0, страницы в физической памяти нет, и остальная часть элемента таблиц страниц может хранить информацию о ее местоположении на дисковом пространстве.

A (accessed) – бит обращения. Сообщает об обращении к таблице.

D (Dirty) – бит мусора. Показывает, что осуществлялась запись в страницу.

R/W (Read/Write) – разрешает доступ только по считыванию (R/W=0) или по считыванию и записи при R/W=1.

U/S (User/Supervisor) – определяет два уровня привилегий для защиты на уровне страниц. U/S=0 задает уровень супервизора для операционной системы и других системных программ и системных данных. U/S=1 для прикладных кода и данных.

PCD (Page Cache Disable) – бит запрещения кэширования.

PWT(Page Write Through) – бит сквозной записи. Управляет режимом кэширования(1-сквозная запись, 0-обратная запись).