КОНСТРУИРОВАНИЕ ПРОГРАММ

Лекция № 03.1 Архитектура процессора

Преподаватель: Поденок Леонид Петрович, 505a-5

+375 17 293 8039 (505a-5)

+375 17 320 7402 (ОИПИ НАНБ)

prep@lsi.bas-net.by

ftp://student:2ok*uK2@Rwox@lsi.bas-net.by/

Кафедра ЭВМ, 2022

2022.02.23

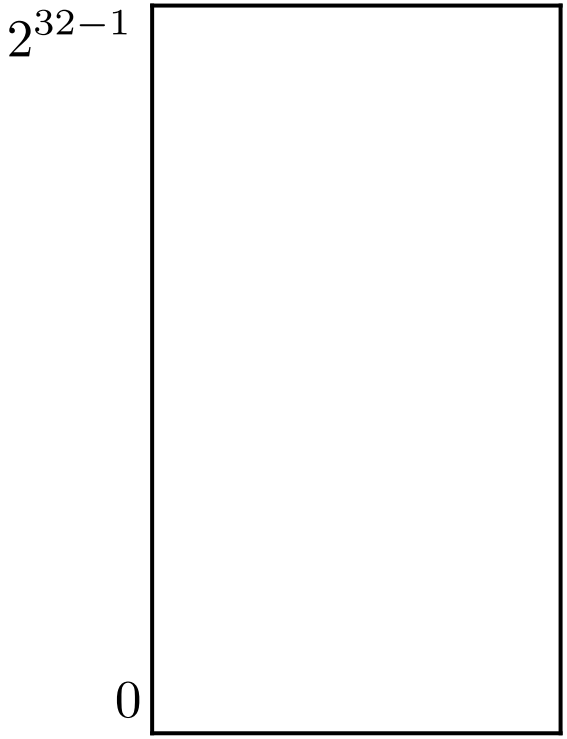
Оглавление

Базовое окружение исполнения программы (для не-64 бит режимов)

Основные регистры исполнения программы

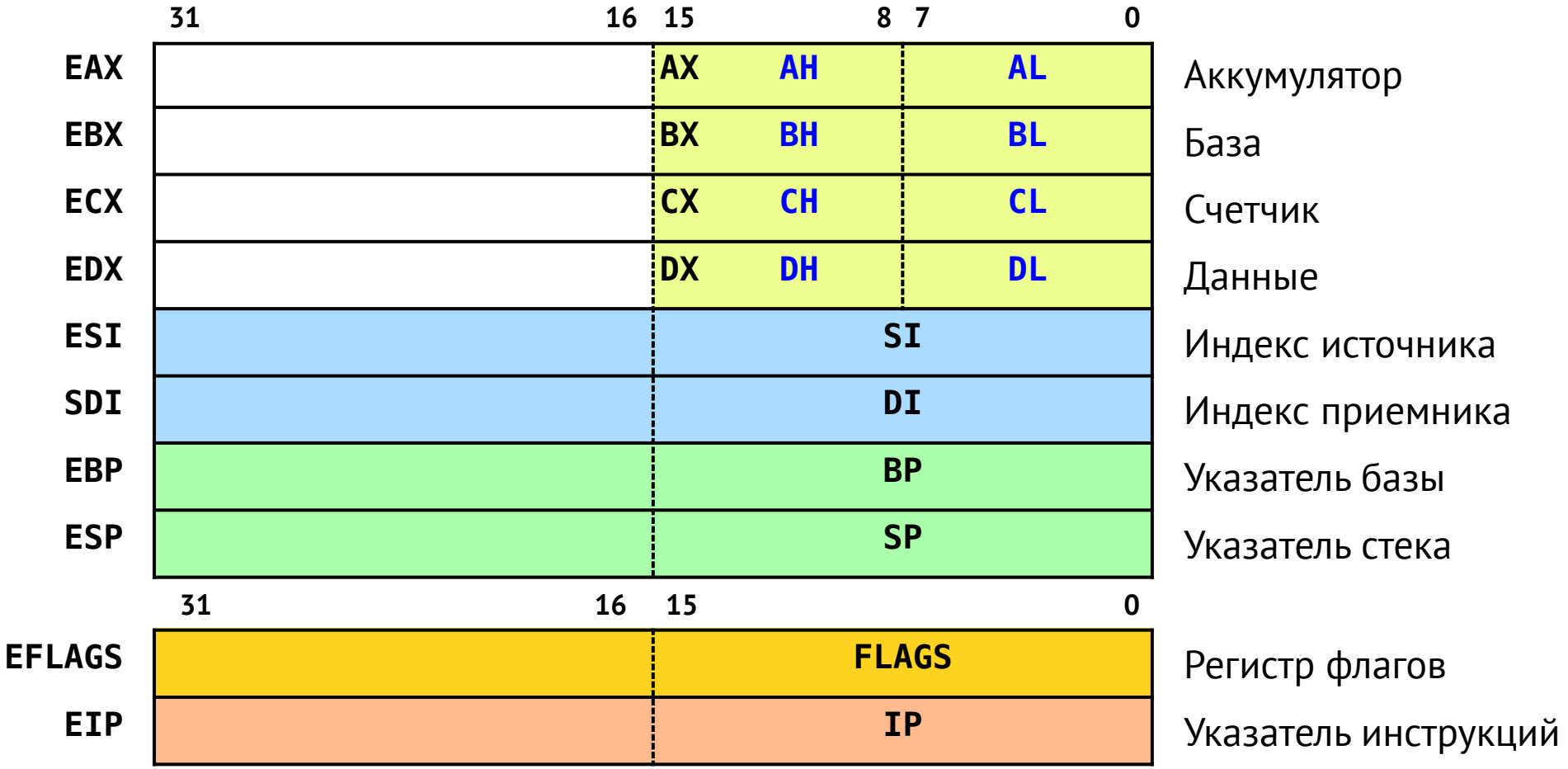
восемь 32-разрядных регистров	Регистры общего назначения EAX, EBX, ECX, EDX ESI, EDI, EBP, ESP
шесть 16-битных регистров	Сегментные регистры (селекторы) CS, DS, ES, SS, FS, GS
32 бита	Регистр флагов EFLAGS
32 бита	Регистр указателя инструкций EIP

Адресное пространство*



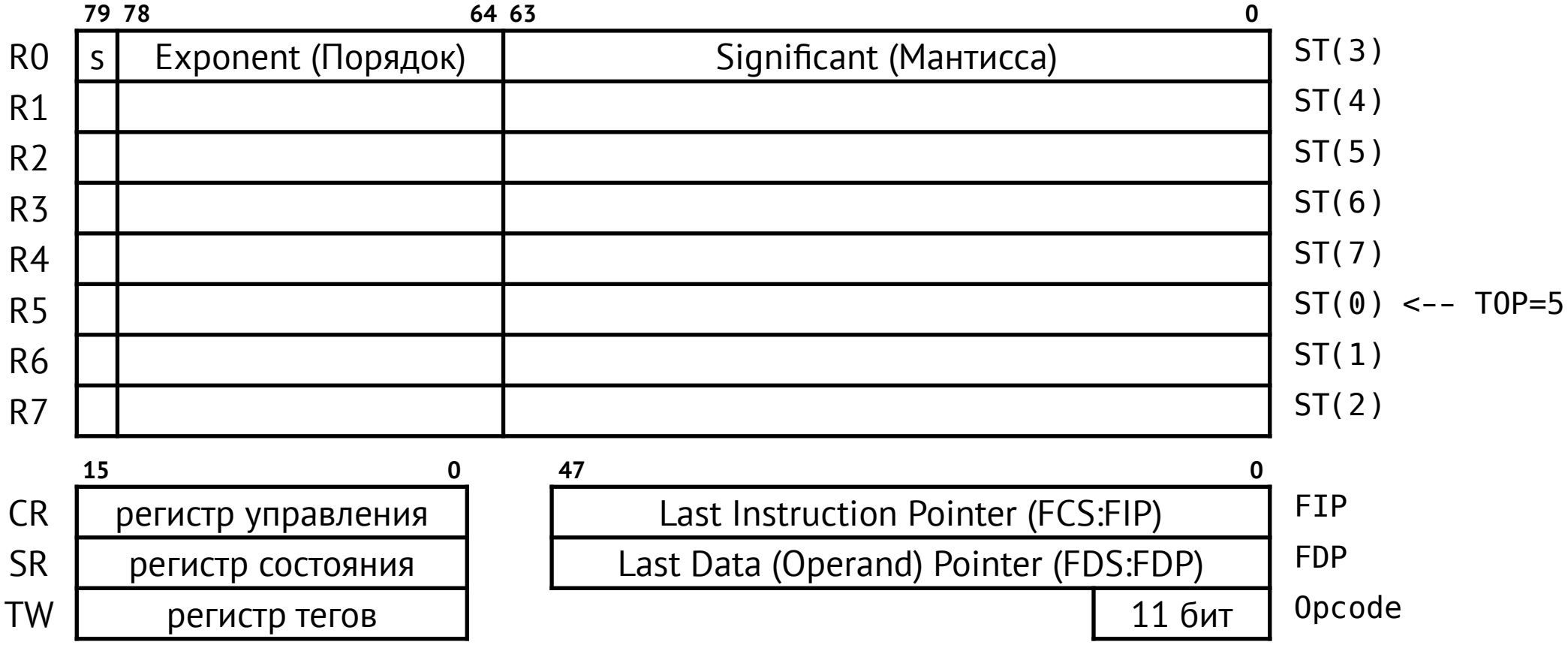
*) 4 Гбайт (PAE → 2^{36-1}). Может быть плоским или сегментированным. Стек.

Регистры общего назначения x86 (IA-32)



Регистры FPU

FPU содержит восемь регистров для хранения данных (bin, bcd) и пять (+1) вспомогательных.

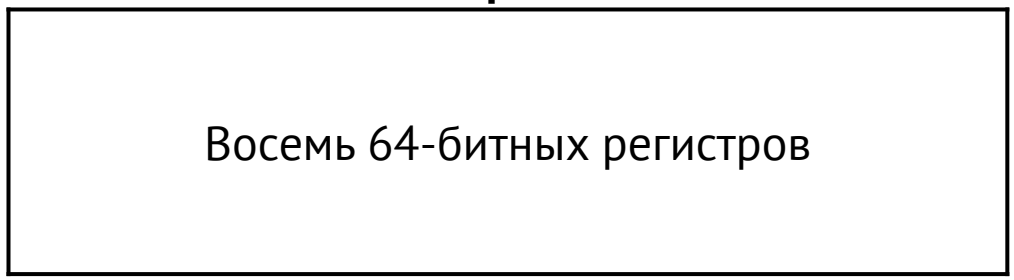


Регистры данных (R0 — R7) — не адресуются по именам, а рассматриваются в качестве регистрового стека, вершина которого называется **`ST(0)`**, более глубокие элементы — **`ST(1)..ST(7)`**. Регистры организованы в виде кольца.

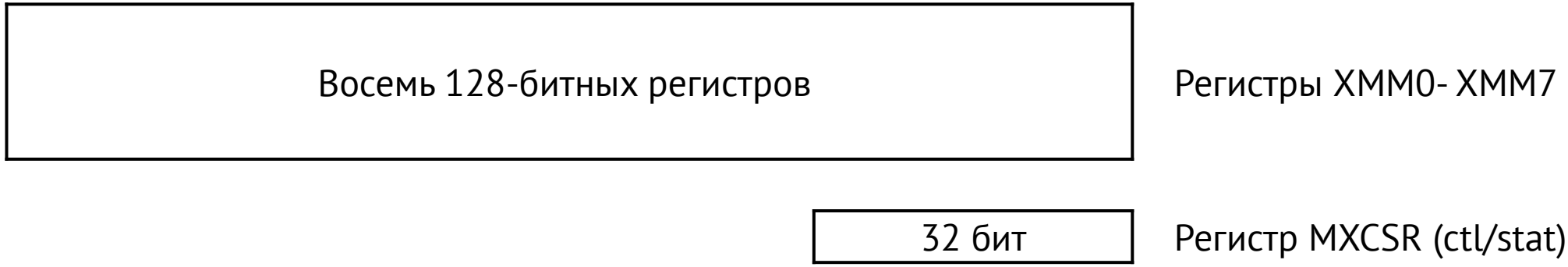
Если процессор поддерживает MMX, мантиссы из регистров стека доступны как MM0..MM7.

Регистры MMX и XMM

Регистры MMX



Регистры XMM



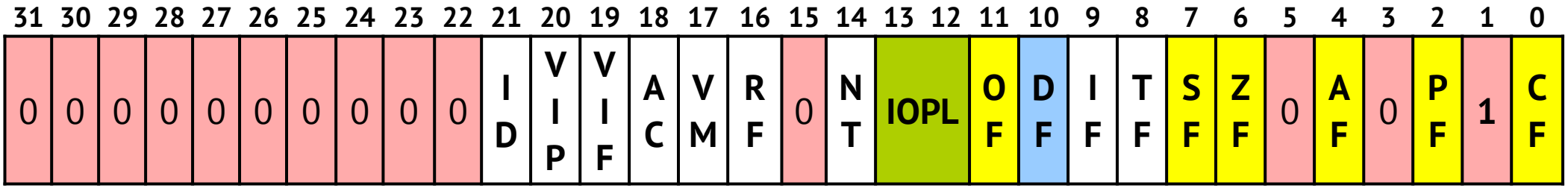
Multi**M**edia e**X**tensions

Streaming **S**IMD **E**xtension — потоковые расширения

SIMD — **S**ingle **I**nstruction — **M**ultiple **D**ata

Предназначено для современных приложений, работающих с 2-d и 3-d графикой, видео-, аудио- и другими видами потоковых данных.

Регистр флагов (EFLAGS Register)



Регистр EFLAGS содержит три группы флагов:

- группу флагов состояния;
- группу флагов управления;
- группу системных флагов.

При инициализации процессора **EFLAGS** устанавливается в значение **00000002H**. Биты 1, 3, 5, 15 и с 22 по 31 зарезервированы. ПО не должно ни полагаться на них, ни использовать.

Некоторые биты могут быть модифицированы прямо с использованием специальных инструкций.

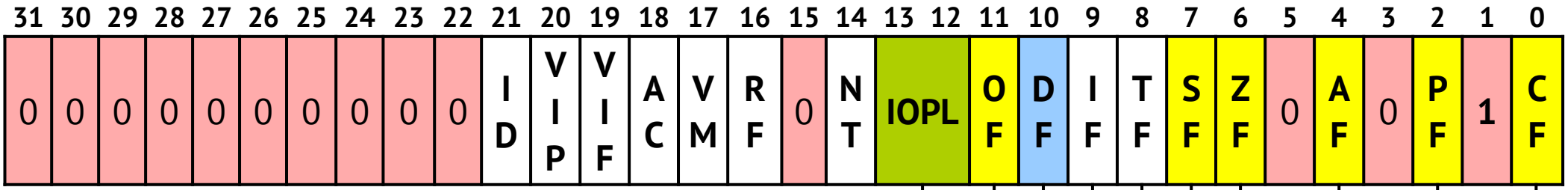
Инструкций, которые бы позволяли модифицировать или проверить EFLAGS целиком, не существует.

Есть ряд команд, позволяющих перемещать группы флагов из/в стек программы или в регистр EAX: **LAHF**, **SAHF**, **PUSHF**, **PUSHFD**, **POPF** и **POPFD**. После того, как содержимое сохранено, флаги можно протестировать.

При приостановке программы процессор автоматически сохраняет состояние флагов в TSS приостановленной задачи. При продолжении или первоначальном старте программы процессор автоматически восстанавливает состояние флагов из TSS приостановленной процедуры (задачи).

Когда возникает прерывание или исключение, автоматически сохраняет состояние флагов в стеке прерванной процедуры.

Регистр флагов (EFLAGS Register)



IOPL — привилегии ввода/вывода

OF — переполнение

DF — флаг направления (STD, CLD)

IF — маска прерываний (CLI, STI)

TF — пошаговый режим

SF — знак результата

ZF — нулевой результат

AF — BCD перенос

PF — бит четности

CF — перенос (STC, CLC, CMC)

— Системный флаг — Флаг состояния — Флаг управления — Не используется

Флаги состояния

Биты 0, 2, 4, 6, 7 и 11 регистра EFLAGS указывают результаты выполнения арифметических инструкций, таких как ADD, SUB, MUL и DIV:

CF (бит 0) — Carry flag

Флаг переноса устанавливается, если генерируется перенос из наиболее значащего бита или заем в него. В противном случае очищается (сбрасывается).

Этот флаг указывает на переполнение для беззнаковой целочисленной арифметики. Он также используется при реализации арифметики многократной точности.

PF (бит 2) — Parity flag

Флаг четности устанавливается, если наименее значимый (least-significant) байт результата содержит четное количество установленных бит (1). В противном случае очищается (сбрасывается).

AF (бит 4) — Auxiliary Carry flag

Дополнительный флаг переноса устанавливается, если арифметическая операция генерирует перенос или заем бита 3, в противном случае очищается (сбрасывается). Флаг используется в двоично-десятичной арифметике (BCD — binary-coded decimal).

ZF (бит 6) — Zero flag —

Флаг нуля устанавливается, если результат равен нулю, в противном случае очищается (сбрасывается).

SF (бит 7) — Sign flag

Флаг знака устанавливается в состояние наиболее значимого бита результата, который в дополнительном коде представляет собой знак целого знакового числа. 0 указывает на положительное, 1 — на отрицательное значение.

OF (бит 11) — Overflow flag

Флаг переполнения устанавливается, если целочисленный результат является либо слишком большим положительным целым, либо слишком малым отрицательным, чтобы поместиться в операнде назначения. В противном случае очищается (сбрасывается). Флаг указывает на возникновение переполнения для целочисленной знаковой арифметики в дополнительном коде (two's complement).

Из перечисленных флагов только **CF** может быть модифицирован непосредственно, используя инструкции STC, CLC и CMC instructions. Также инструкции манипулирования битами BT, BTS, BTR и BTC копируют указанный бит в **CF**.

Флаг управления

DF (бит 10) — Direction Flag

Флаг направления управляет поведением строковых инструкций **MOVS**, **CMPS**, **SCAS**, **LODS** и **STOS** — установка **DF** вызывает автодекремент адреса (обработку в направлении от более высоких адресов к более низким). Очистка (сброс) — автоинкремент (обработку в направлении от более низких адресов к более высоким).

Устанавливают и очищают **DF**, соответственно, инструкции **STD** и **CLD**.

Системные флаги и поле IOPL

Системные флаги и поле привилегий ввода-вывода управляют поведением операционной системы и операциями по загрузке программ и их выполнением. Прикладным программам не следует их использовать.

TF (бит 8) — Trap flag — Устанавливает пошаговый режим выполнения программы для отладки. Очистка запрещает данный режим.

IF (бит 9) — Interrupt enable flag — Управляет реакцией процессора на запрос маскируемых прерываний. Установка разрешает, а очистка — запрещает (CLI, STI).

IOPL (биты 12 и 13) — I/O privilege level field — Указывает привилегии ввода/вывода для текущей выполняющейся программы или задачи. Чтобы программа могла получить доступ к адресному пространствы ввода/вывода, CPL (текущий уровень привилегий) должен быть меньше, либо равен IOPL.

Инструкции POPF и IRET могут модифицировать это поле только в привилегированном режиме (CPL = 0).

NT (bit 14) — Nested task flag — Управляет процессом взаимодействия прерываний и вызываемой задачи.

RF (bit 16) — Resume flag — Управляет реакцией процессора на отладочное исключение.

VM (bit 17) — Virtual-8086 mode flag — Устанавливается для разрешения режима виртуального 8086, очищается для возврата в защищенный режим.

AC (bit 18) — Alignment check (or access control) flag — используется для управления выравниванием. Если бит AM bit в CR0 установлен, разрешается проверка выравнивания данных в непривилегированном режиме (user-mode).

VIF (bit 19) — Virtual interrupt flag и **VIP (bit 20) — Virtual interrupt pending flag** — используются для управления поведением процессора при обработке прерываний.

ID (bit 21) — Identification flag — Используется при управлении памятью в защищенном режиме.

Указатель инструкций

Регистр EIP содержит смещение следующей инструкции, подлежащей исполнению, в текущем сегменте кода. Его значение изменяется от границы одной инструкции к другой в порядке последовательности команд. Также продвигается вперед или назад при выполнении инструкций **JMP**, **Jcc**, **CALL**, **RET** и **IRET**.

Регистр **EIP** не доступен программе напрямую — он управляется неявно с помощью инструкций передачи управления, таких как **JMP**, **Jcc**, **CALL** и **RET**, прерываний и исключений.

Единственный способ прочитать регистр EIP состоит в выполнении инструкции CALL и после этого прочитать в стеке адрес возврата.

Регистр **EIP** может быть загружен непосредственно с помощью модификации адреса возврата в стеке и вызова инструкций возврата (**RET** или **IRET**).

Сегментные регистры x86_64 (16 бит)

Сегментные регистры отвечают за разбиение памяти на сегменты

15 Видимая часть 0		Скрытая часть
Селектор сегмента	CS	Базовый адрес, предел, информация о доступе
-//-	SS	-//-
-//-	DS	-//-
-//-	ES	-//-
-//-	FS	-//-
-//-	GS	-//-

(¤) Дескриптор сегмента

Формат поля доступа

15 7	14 6	13 5	12 4	11 3	10 2	9 1	8 0	в слове / в байте	
P	DPL		**1**	1	C	R	A		Поле доступа сегмента кода
P	DPL		**1**	0	D	W	A		Поле доступа сегмента данных
P	DPL		**0**	TYPE					Поле доступа системного сегмента

S – системный/несистемный (код или данные) объект.

R – разрешение чтения;

W – разрешение записи в сегмент;

D – направление расширения сегмента (0 – обычный, 1 – стек);

Для сегмента кода может быть создан второй дескриптор (алиас), который помечен, как сегмент данных с разрешением записи.

C – бит подчинения;

P – бит присутствия сегмента в памяти;

A – бит обращения к сегменту;

DPL – уровень привилегий дескриптора.

(¤) Дескриптор сегмента (8 байт)

L (только 64-битный режим) 0 - процессор в режиме совместимости.
D (Default size). 0 — 16-разрядный объект; 1 — 32-разрядный.
G (Granularity). Если бит G = 0, то сегмент имеет байтную гранулярность предела сегмента, иначе - страничную (одна страница - это 4 Кб).
AVL (доступность). Для использования программным обеспечением (оборудованием не используется)

Все дескрипторы хранятся в глобальной и локальных дескрипторных таблицах **GDT** и **LDT**.
Размер данных или адреса по умолчанию, определяемый битом **D**, может быть изменен в конкретном случае с помощью префикса.

(¤) Глобальная дескрипторная таблица

Глобальная дескрипторная таблица (далее GDT) хранится в памяти.

GDT можно расположить по любому адресу в памяти, но рекомендуется располагать её по адресу, выровненному на границу 8 байт. В GDT может храниться до 8192 дескрипторов. Нулевой дескриптор, т.е. дескриптор, определённый в самом начале GDT, процессор не использует. Адрес таблицы и её лимит хранятся в регистре процессора **GDTR**.

Лимит таблицы нужен для того, чтобы не произошло обращения за границу таблицы. Регистр GDTR в защищённом режиме имеет размер 6 байт.

(¤) Формат регистра GDTR (6 байт)

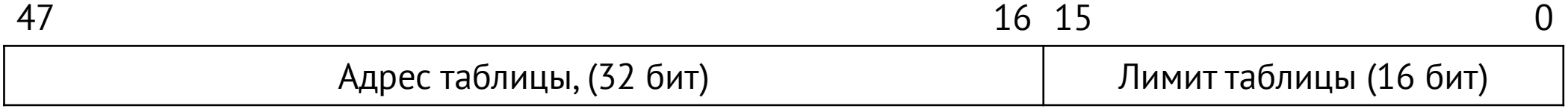
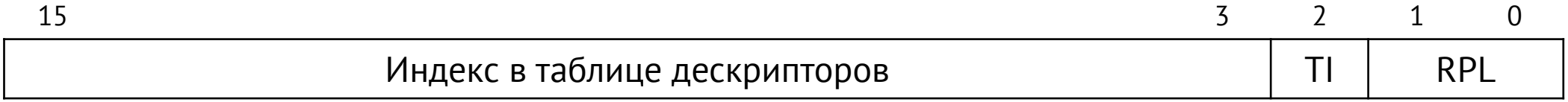


Таблица может быть не больше чем 64 Кб, отсюда максимальное количество дескрипторов 8192 (65536/8=8192), точнее 8191, т.к. нулевой дескриптор не используется.

Для загрузки регистра GDTR используется команда **lgdt**, для чтения - **sgdt**.

(¤) Формат селектора



Селектор - это индекс дескриптора в GDT

Селектор имеет размер 2 байта и содержится в сегментом регистре или задаётся непосредственным значением.

Бит TI указывает, к какой таблице следует обратиться (0 — GDT, 1 — LDT).

Поле RPL обозначает запрашиваемый уровень привилегий.

У каждого сегментного регистра есть теневая часть, доступная только процессору, размером 8 байт, и при каждой загрузке селектора в регистр процессор проверяет установки защиты - если всё прошло успешно, то он загружает весь дескриптор в теневую часть сегментного регистра. В дальнейшем при каждом обращении к памяти процессор обращается не к таблице, а к теневой части сегментного регистра. И если изменить в таблице её используемый дескриптор, то никаких изменений не произойдёт, т. к. надо еще раз загрузить требуемый селектор в соответствующий сегментный регистр.

Проверка лимитов сегментов

Проверка предела дескриптора сегмента не позволяет программе обращаться за пределы сегмента. Значение предела зависит от флага гранулярности дескриптора.

Предел определяет максимальное значение смещения в сегменте.

Для сегментов данных, расширяющихся вниз, предел определяет последний адрес, доступ к которому запрещён внутри сегмента. Допустимыми будут адреса в диапазоне от (предел + 1) до FFFFh, если флаг D=0, и в диапазоне от (предел + 1) до FFFFFFFFh, если D=l. Максимальный размер такие сегменты имеют с пределом равным нулю. Помимо проверок обычных сегментов, процессор также проверяет пределы дескрипторных таблиц GDT, IDT, LDT и текущего сегмента TSS, не позволяя обращаться за их пределы

Проверки типов

Дескриптор сегмента содержит информацию о типе в двух элементах:

- флаг S (тип дескриптора);
- поле типа.

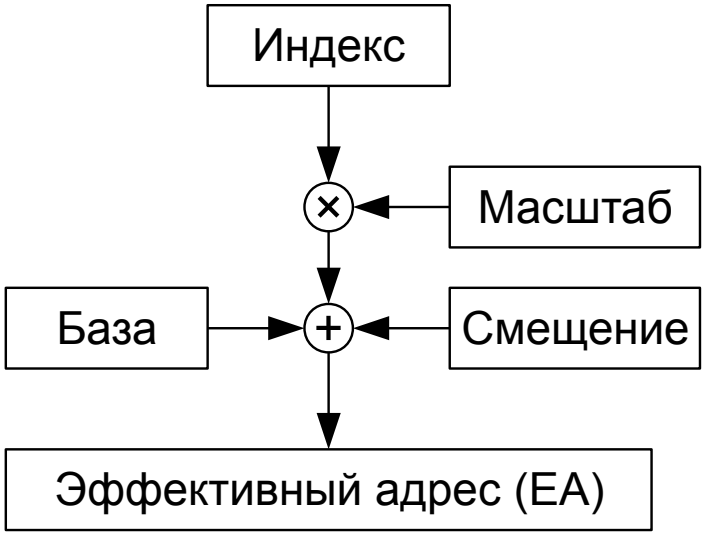
Процессор использует эту информацию для определения ошибочных действий программ, когда они пытаются использовать сегмент или шлюз неправильным или несоответствующим образом.

Флаг S определяет, что описывает дескриптор: системный объект или сегмент кода либо данных. Поля 9-11 содержат 4 дополнительных бита, определяющие различные типы несистемных дескрипторов.

Адресация памяти в x86 (IA32)

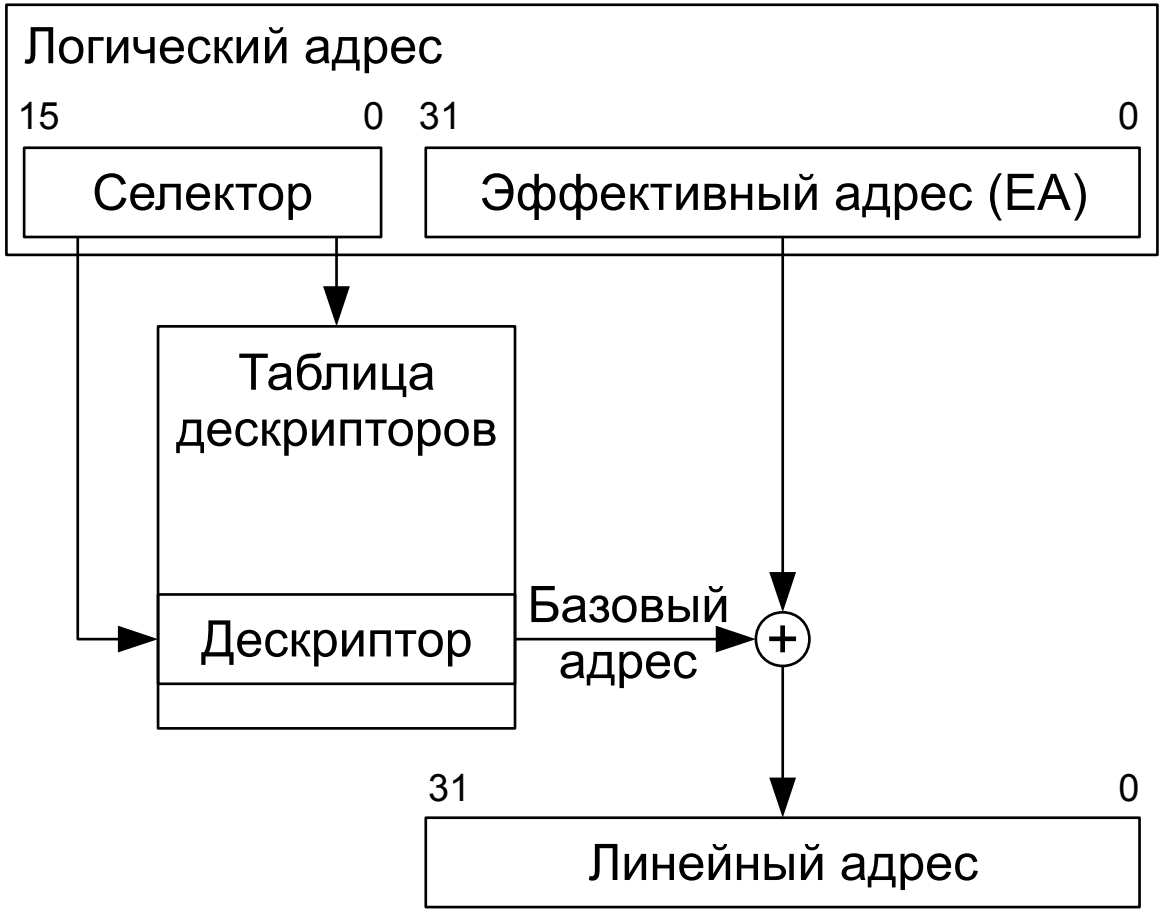
Существует четыре вида адресов:
1. эффективный (адрес относительно начала сегмента).
2. логический (сегмент и эффективный адрес);
3. линейный (или виртуальный);
4. физический — в системной памяти;

Формирование эффективного адреса



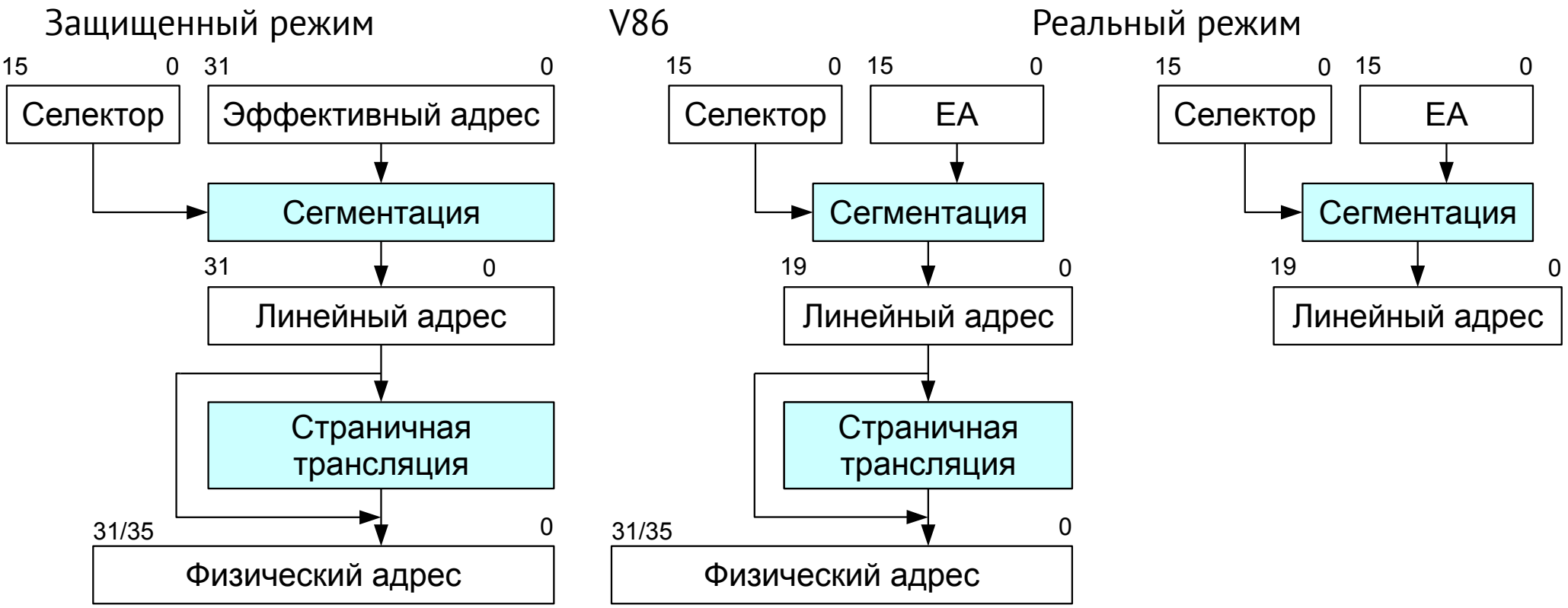
База — может содержаться в любом из РОН;
Индекс — может содержаться в любом из регистров, за исключением ESP;
Смещение — содержится в коде команды;
Масштаб — содержится в коде команды (1, 2, 4, 8).

Формирование линейного адреса (сегментация)



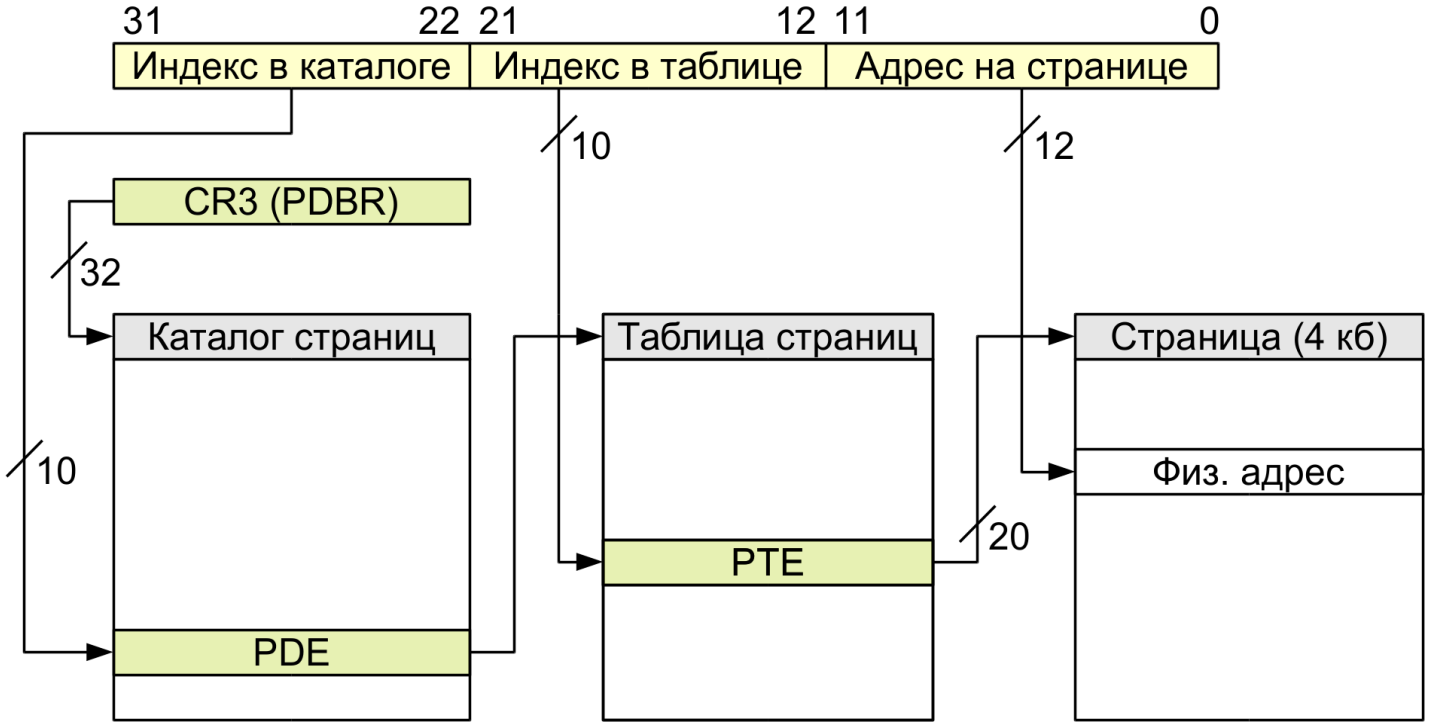
1. из селектора извлекается поле индекса;
2. по индексу находится соответствующий дескриптор;
3. из дескриптора извлекается поле адреса базы.
4. К адресу базы добавляется смещение.

Формирование физического адреса



(¤) Трансляция виртуального адреса в физический в обычном режиме

При каждом обращении к памяти виртуальный адрес делится на три части:



Первая часть — индекс (PDE — Page Directory Entry) элемента в каталоге страниц (Page Directory). Из этого элемента извлекается физический адрес таблицы страниц (Page Table). **Вторая часть —** индекс (PTE — Page Table Entry) в таблице страниц. Из этого элемента извлекается физический адрес страницы. **Третья часть** интерпретируется как смещение в этой странице.

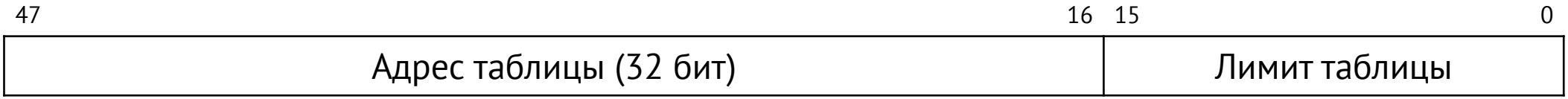
(¤) Регистр CR4

Регистр CR4 управляет дополнительными возможностями процессора, а также возможностями, которыми не управляет регистр CR0.

(¤) Регистры GDTR и IDTR

Регистры GDTR и IDTR служат для указания параметров глобальных таблиц: глобальной дескрипторной таблицы (GDT) и глобальной таблицы прерываний (IDT).

Формат регистров GDTR и IDTR



Адрес таблицы является абсолютным виртуальным адресом, он не зависит от содержимого каких-либо сегментных регистров.

(¤) Регистры LDTR и TR

Регистры LDTR и TR имеют размер 16 бит и хранят селекторы локальной дескрипторной таблицы (LDT) и текущей выполняющейся задачи (TSS) в GDT.

Также в этих регистрах есть теневые части, которые содержат сами дескрипторы в целях минимизации обращений к глобальной дескрипторной таблице (GDT).