ТСИС

(Технические средства информационных систем)

Программное обеспечение информационных систем (1-40 01 73)

Гр. 60325, 60326

Адресация. Режимы работы процессора. Управление памятью.

Лекция 6
(По материалам Мухаметова В.Н.)

Ковалевский Вячеслав Викторович

2016

Ковалевский Вячеслав Викторович

4096tb@gmail.com

Тема письма:
БГУИР. … .

Лекция 5. Структура процессора. Архитектуры CISC и RISC. Архитектура процессора Intel .

План лекции:

- Структура процессора. Шинная организация.
- Архитектуры CISC и RISC. Архитектура IA-32. Регистры процессора.
- Формат команды. Классификация команд. Особенности состава команд Intel.
- Взаимодействие с памятью и вводом-выводом. Цикл шины. Ввод-вывод: программный, по прерываниям и ПДП.

Экзаменационные вопросы:

- Буферные элементы. Шинная организация современного компьютера.
- Понятие архитектуры компьютера. Структура компьютера. Понятие о CISC и RISC.
- Регистры общего назначения и их особенности у Intel.
- Команда. Формат команды. Классификация команд. Особенности состава команд Intel.

Лекция 6. Адресация. Режимы работы процессора. Управление памятью.

План лекции:

- Адресация памяти. Непосредственная, прямая и косвенная адресация. Автоинкрементная и автодекрементная адресация. Строковые команды. Стек.
- Режимы работы процессора Intel.
- Сегментная и страничная организация доступа к памяти. Сегментация памяти в реальном режиме. Дескрипторы сегментов. Дескрипторные таблицы.
- Шлюзы. Виртуальная память. Подкачка страниц. Размеры страниц и расширение адреса.

Экзаменационные вопросы:

- Адресация памяти и ввода-вывода. Циклы обмена между процессором и памятью.
- Абсолютная, прямая и косвенная адресация. Автоинкрементная и автодекрементная адресация.
- Стек. Работа стека и его использование. Ввод-вывод: программный, по прерываниям и ПДП.
- Режимы работы процессора Intel. RM, VM, PM, SMM.
- Сегментная и страничная организация доступа к памяти. Сегментация памяти в реальном режиме. Страничная организация – реализация виртуальной памяти.
- Управление сегментами в защищенном режиме. Дескрипторные таблицы. Дескрипторы сегментов.

Общие сведения о микропроцессорах

Классификация микропроцессоров

Тактовая частота обработки информации

Тактом называют время между началом подачи двух последовательных импульсов электрического тока, синхронизирующих работу различных устройств компьютера. Специальные импульсы для отсчета времени для всех устройств вырабатывает тактовый генератор частоты, расположенный на системной плате.

Разрядность процессора

Разрядность процессора - это число битов, обрабатываемых процессором одновременно. Процессор может быть 8-, 16-, 32- и 64-разрядным.

Адресное пространство (адресация памяти)

Объем физически адресуемой МП оперативной памяти называется его адресным пространством. Он определяется разрядностью внешней шины адреса. Поэтому разрядность процессора часто уточняют записывая, например, 32/32, это значит МП имеет 32х разрядную шину данных и 32х разрядную шину адреса, т.е. одновременно обрабатывается 32 бита информации, а объём адресного пространства МП составляет 4 Гбайта.

Типы процессоров

Процессор	Год вып.	Разрядность: РОН/ШД/ША	Частота системной шины, МГц	Частота ядра, МГц	Встроенный кэш* (инструкции/данные)	Доп.возм.**
8086	1978	16/16/20	4.77-8		-	-
8088	1979	44059				
80286	1982	16/16/24	44044		-	-
80386DX	1985	32/32/32	16, 20, 33		-	-
80386SX	1988	32/16/24				-
80386SL	1991	32/16/24				SMM
i486DXn	1989	32/32/32	20, 25, 33	20-100	L1: 8K (4K/4K)	FPU
i486SXn	1991					-
i486SL	1992					SMM
Pentium	1993	32/64/32	50, 60, 66	60-200	L1: 16K (8K/8K)	SMM, FPU
Pentium MMX	1995	32/64/32	66	166, 200, 233	L1: 32K (16K/16K)	SMM, FPU, MMX
Pentium Pro	1995	32/64/36	50, 60, 66	150-200	L1: 16K (8K/8K)	SMM, FPU
					L2: 256K, 512K	

Типы процессоров

Pentium II, Pentium II Xeon	1997	32/64/36	66,1	233-450	L1: 32K (16K/16K) L2: 512K	SMM, FPU, MMX
Celeron (Covington)	1998	32/64/32	66	266, 300	L1: 32K (16K/16K)	SMM, FPU, MMX
Celeron (Mendocino)	1998	32/64/32	66	300-533	L1: 32K (16K/16K) L2: 128K	SMM, FPU, MMX
Pentium III, PIII-Coppermine	1999	32/64/36	100,133	450-600, 500-1130	L1: 32K (16K/16K) L2: 256K	SMM, FPU, MMX, SSE
Celeron (Coppermine)	2000	32/64/32	66,1	533-900	L1: 32K (16K/16K) L2: 128K	SMM, FPU, MMX, SSE
Pentium III Xeon	1999	32/64/36	100, 133	500-866	L1: 32K (16K/16K) L2: 256K-2M	SMM, FPU, MMX, SSE
Pentium 4	2000	32/64/36	4x100	1400+	L1: 20K (12K/8K) L2: 256K	SMM, FPU, MMX, SSE2

* Pentium II/III Xeon выпускаются с L2 кэшем до 2M. У Pentium 4: вместо кэша команд - кэш микроопераций

** SMM - процессор поддерживает режим системного управления

FPU - процессор содержит встроенный блок вещественной арифметики

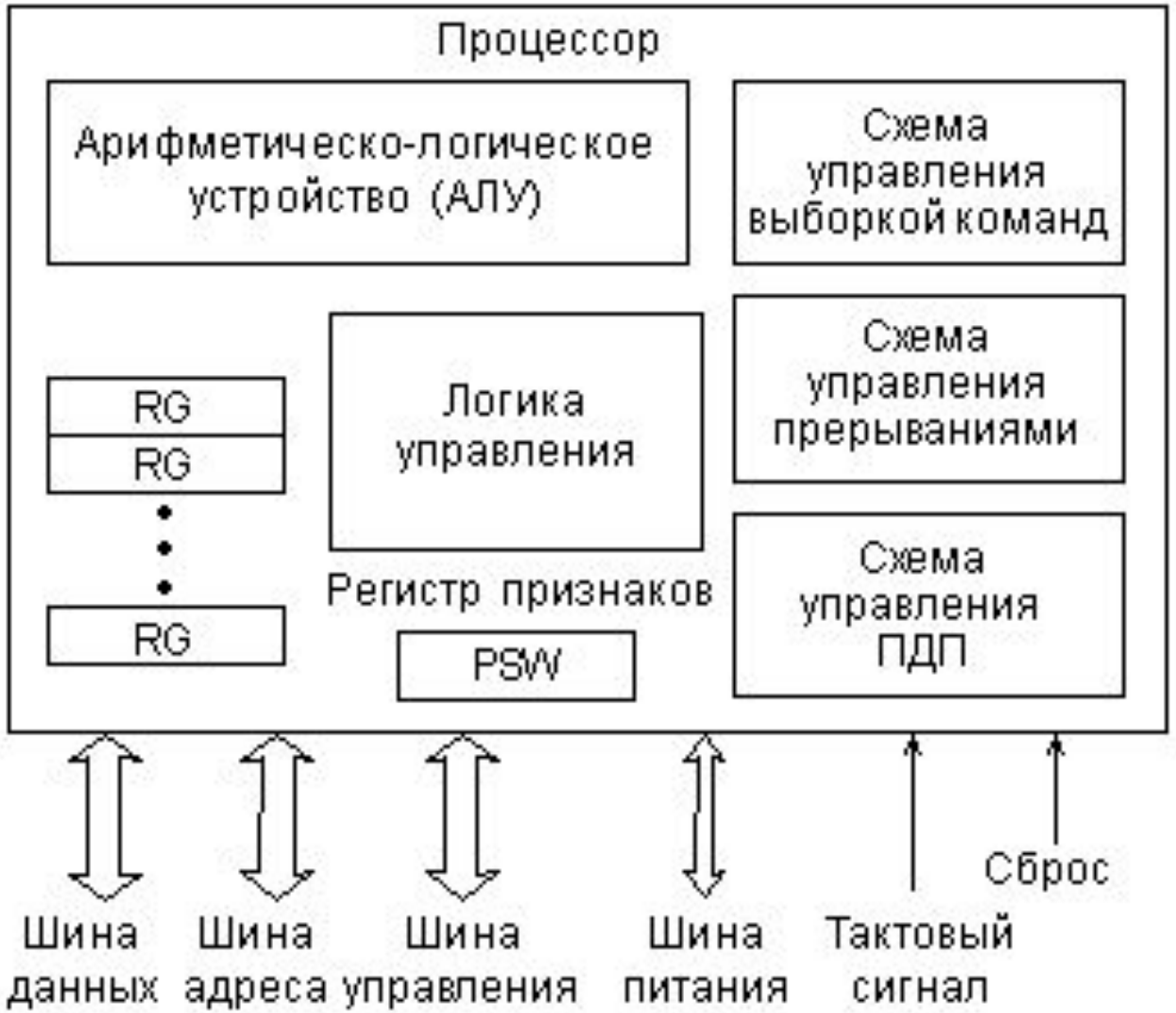
MMX - процессор поддерживает команды параллельной обработки пакетов целочисленных данных

SSE - процессор поддерживает команды параллельной обработки пакетов вещественных данных

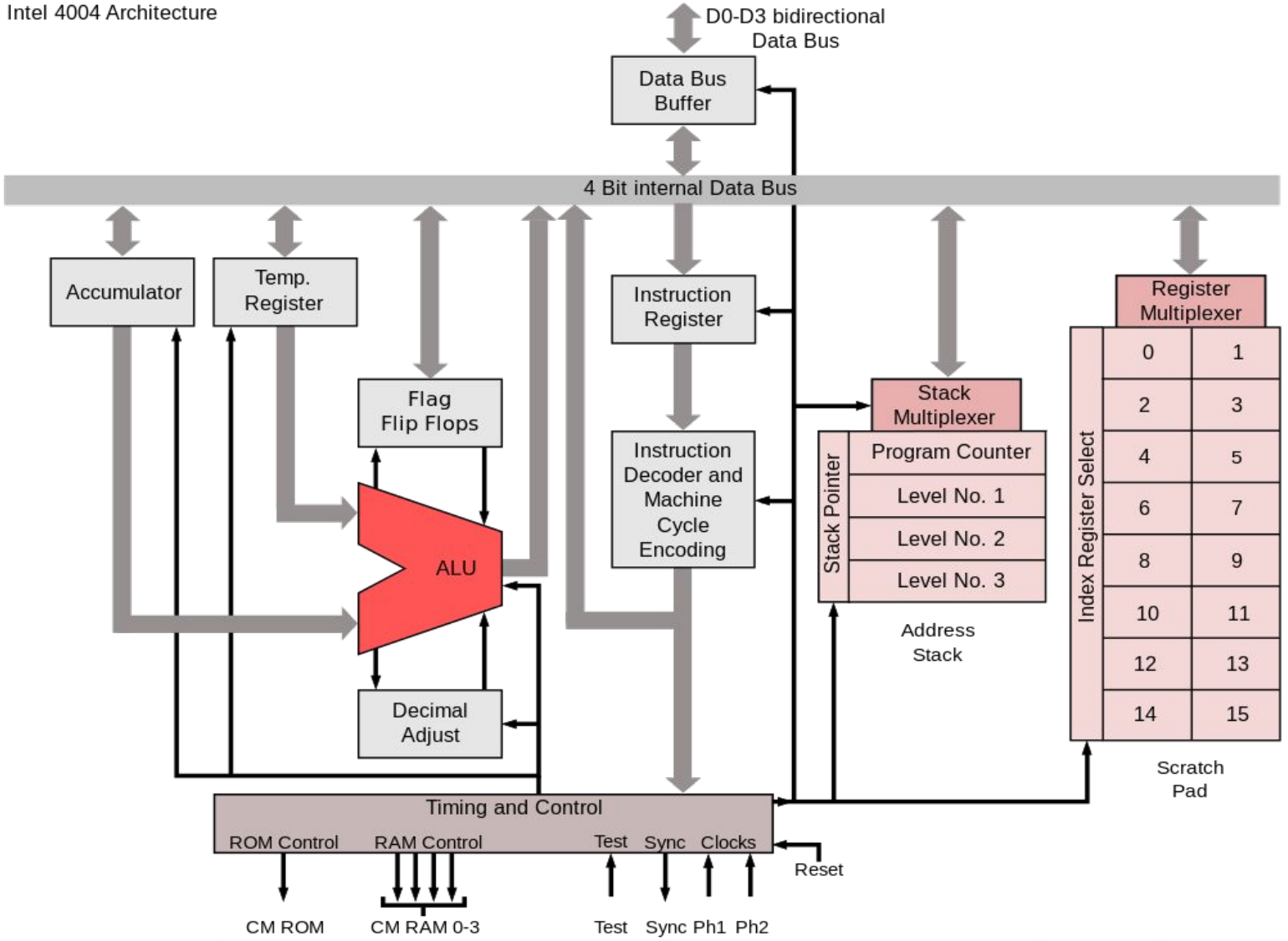
SSE2 - процессор поддерживает дополнительный набор команд SSE

Структура процессора

Структура процессора



Структура процессора Intel 4004



АРХИТЕКТУРА ПРОЦЕССОРА INTEL 8086

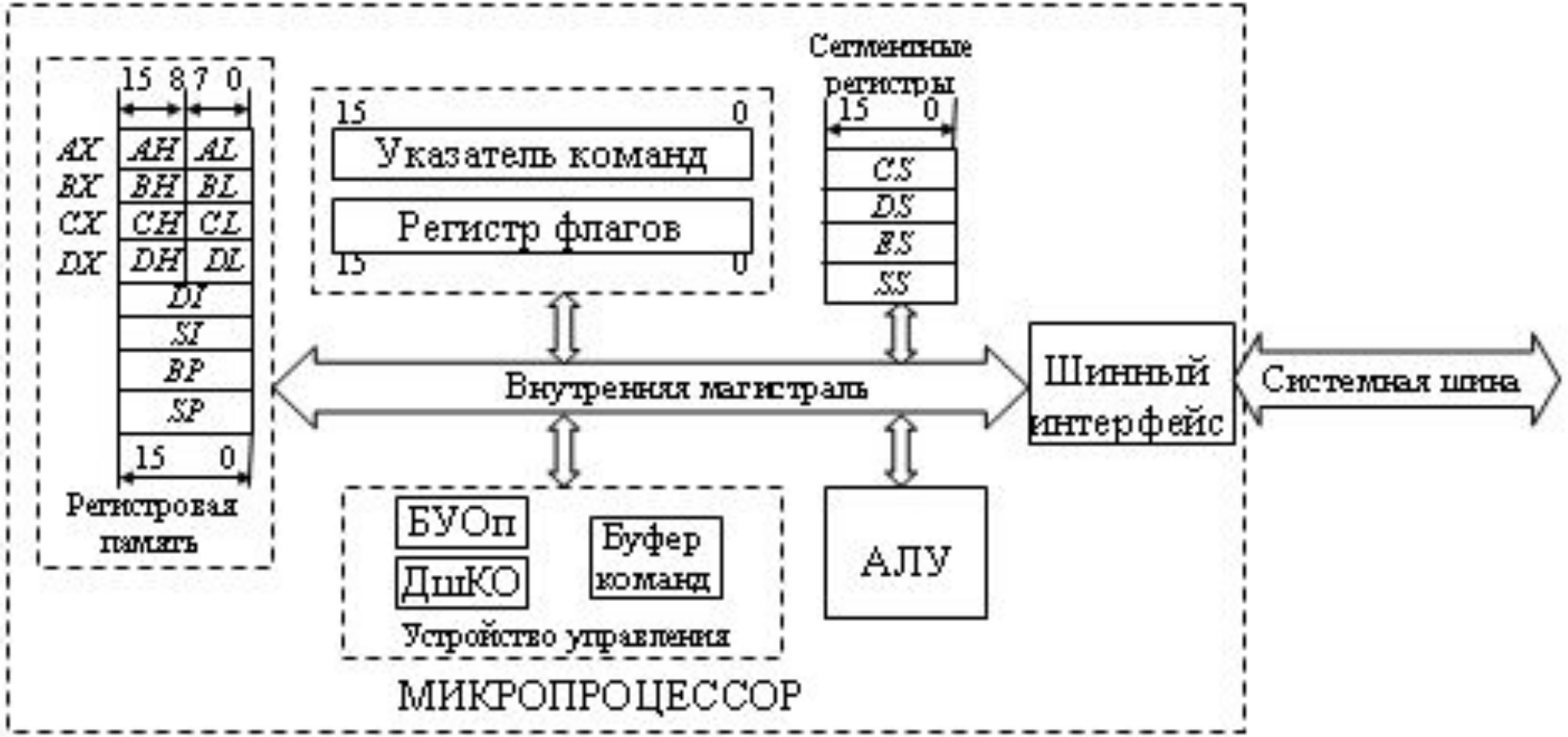
Микропроцессор Intel 8086 приспособлен для работы с несколькими процессорами в одной системе, причем возможно использование как независимых процессоров, так и сопроцессоров

Внешние шины адреса и данных в 8086 объединены, и поэтому наличие на шине в данный момент времени информации или адреса определяется порядковым номером такта внутри цикла. Процессор ориентирован на параллельное выполнение команды и выборки следующей команды

Микропроцессор i8086 состоит из трех основных частей: устройства сопряжения шины, устройства обработки и устройства управления и синхронизации

Устройство сопряжения шины состоит из шести 8-разрядных регистров очереди команд, четырех 16-разрядных регистров адреса команды, 16-разрядного регистра команды и 16-разрядного сумматора адреса.

Структура процессора Intel 8086



ПРОГРАММНАЯ МОДЕЛЬ ПРОЦЕССОРА 8086

Программная модель процессора - это функциональная модель, используемая программистом при разработке программ в кодах ЭВМ или на языке ассемблера. В такой модели игнорируются многие аппаратные особенности в работе процессора. В процессоре 8086 имеется несколько быстрых элементов памяти, которые называются регистрами. Каждый из регистров имеет уникальную природу и предоставляет определенные возможности, которые другими регистрами или ячейками памяти не поддерживаются.

Регистры разбиваются на четыре категории: регистры общего назначения, регистр флагов, указатель команд и сегментные регистры. Все регистры 16-разрядные.

Регистры i8086 (все 16-ти разрядные)

Регистры общего назначения (AX,BX,CX,DX)

Сегментные регистры:

- CS – для кодового сегмента
- DS – регистр дополнительного сегмента
- SS – сегментный регистр сегмента стека
- IP – указатель на инструкцию

Регистры указатели и индексные регистры:

- SP – Stek pointer
- BP – Base pointer
- DI – destination index
- SI – source index

Флаговый регистр

Регистры i8086

Регистры общего назначения

Восемь регистров общего назначения процессора 8086 (каждый разрядностью 16 бит) используются в операциях большинства инструкций в качестве источника или приемника при перемещении данных и вычислениях, указателей на ячейки памяти и счетчиков. Каждый регистр общего назначения может использоваться для хранения 16-битового значения, в арифметических и логических операциях, может выполняться обмен между регистром и памятью (запись из регистра в память и наоборот).

Регистр AX всегда используется в операциях умножения или деления и является также одним из тех регистров, который можно использовать для наиболее эффективных операций (арифметических, логических или операций перемещения данных).

Регистр BX может использоваться для ссылки на ячейку памяти (указатель), т.е. 16-битовое значение, записанное в BX, может использоваться в качестве части адреса ячейки памяти, к которой производится обращение.

Регистр CX – используется в качестве счетчика при выполнении циклов.

Регистр DX - это единственный регистр, которые может использоваться в качестве указателя адреса ввода-вывода в командах IN и OUT.

Регистр SI может использоваться, как указатель на ячейку памяти.

Регистр DI его можно использовать в качестве указателя ячейки памяти.

При использовании его в строковых командах регистр DI несколько отличается от регистра SI. В то время как SI всегда используется в строковых командах, как указатель на исходную ячейку памяти (источник), DI всегда служит указателем на целевую ячейку памяти (приемник).

Регистр SP называется также указателем стека. Стек - это область памяти, в которой можно

Сегментные регистры. Основной предпосылкой сегментации является следующее: процессор 8086 может адресоваться к 1 мегабайту памяти. Для адресации ко всем ячейкам адресного пространства в 1 мегабайт необходимы 20-разрядные сегментные регистры. Однако процессор 8086 использует только 16-разрядные указатели на ячейки памяти. Процессор 8086 использует двухступенчатую схему адресации. Каждый 16-разрядный указатель памяти или смещение комбинируется с содержимым 16-разрядного сегментного регистра для формирования 20-разрядного адреса памяти.

Аналогично регистрам общего назначения каждый сегментный регистр играет свою, конкретную роль. Регистр CS указывает на код программы, DS указывает на данные, SS - на стек. Сегментный регистр ES - это дополнительный сегмент, который может использоваться так, как это необходимо.

Формат регистра флагов Intel 8086

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
flags	**	**	**	**	OF	DF	IF	TF	SF	ZF	**	ZF	**	PF	**	CF

OF - флаг переполнения;
DF - флаг направления;
IF - флаг прерывания;
TF - флаг трассировки;
SF - флаг знака;
ZF - флаг нуля;
AF - флаг дополнительного переноса;
PF - флаг четности;
CF - флаг переноса;
** - бит не используется, состояние не определено

Сегментные регистры i8086

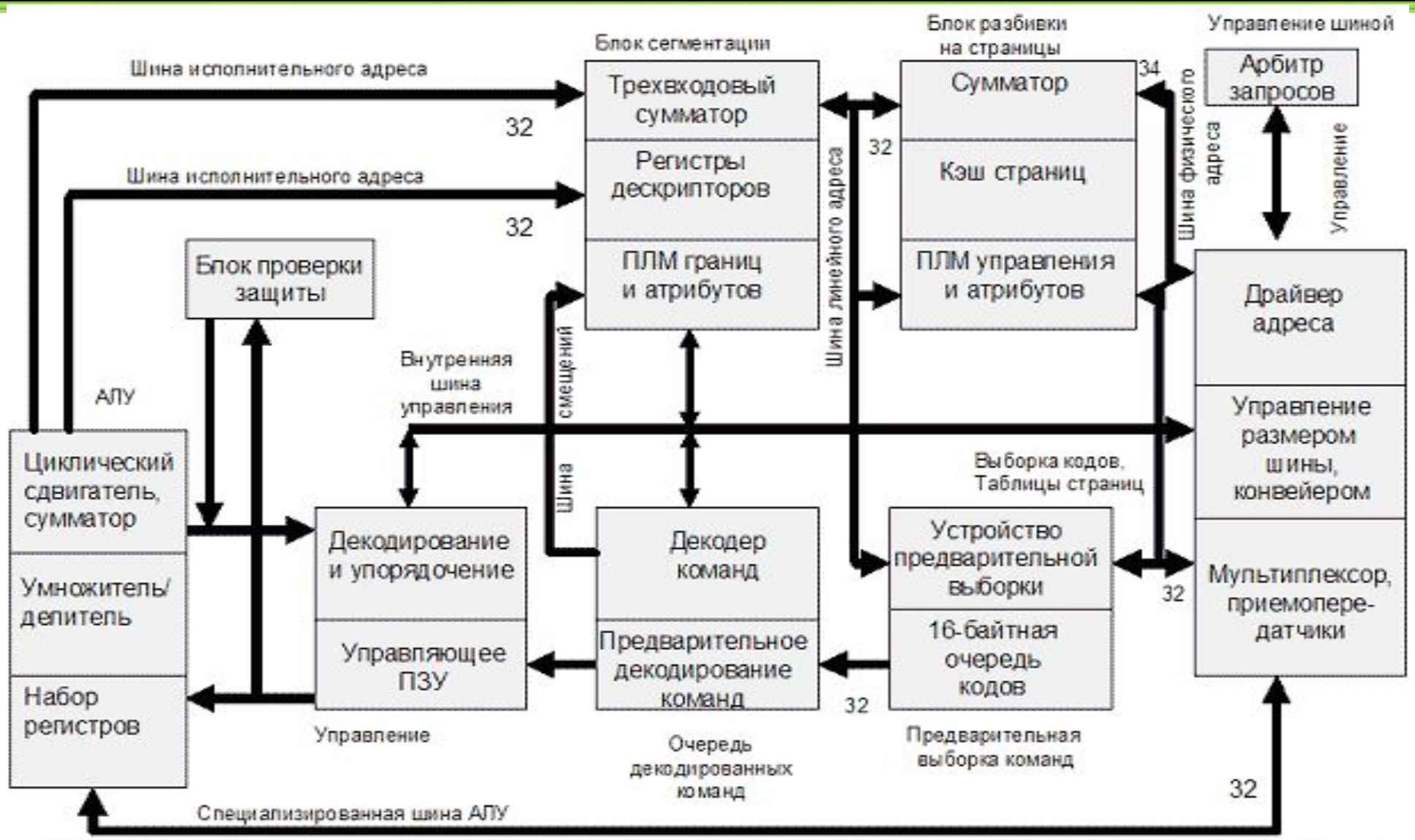
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cs																
ds																
es																
ss																

Формат регистра команд Intel 8086

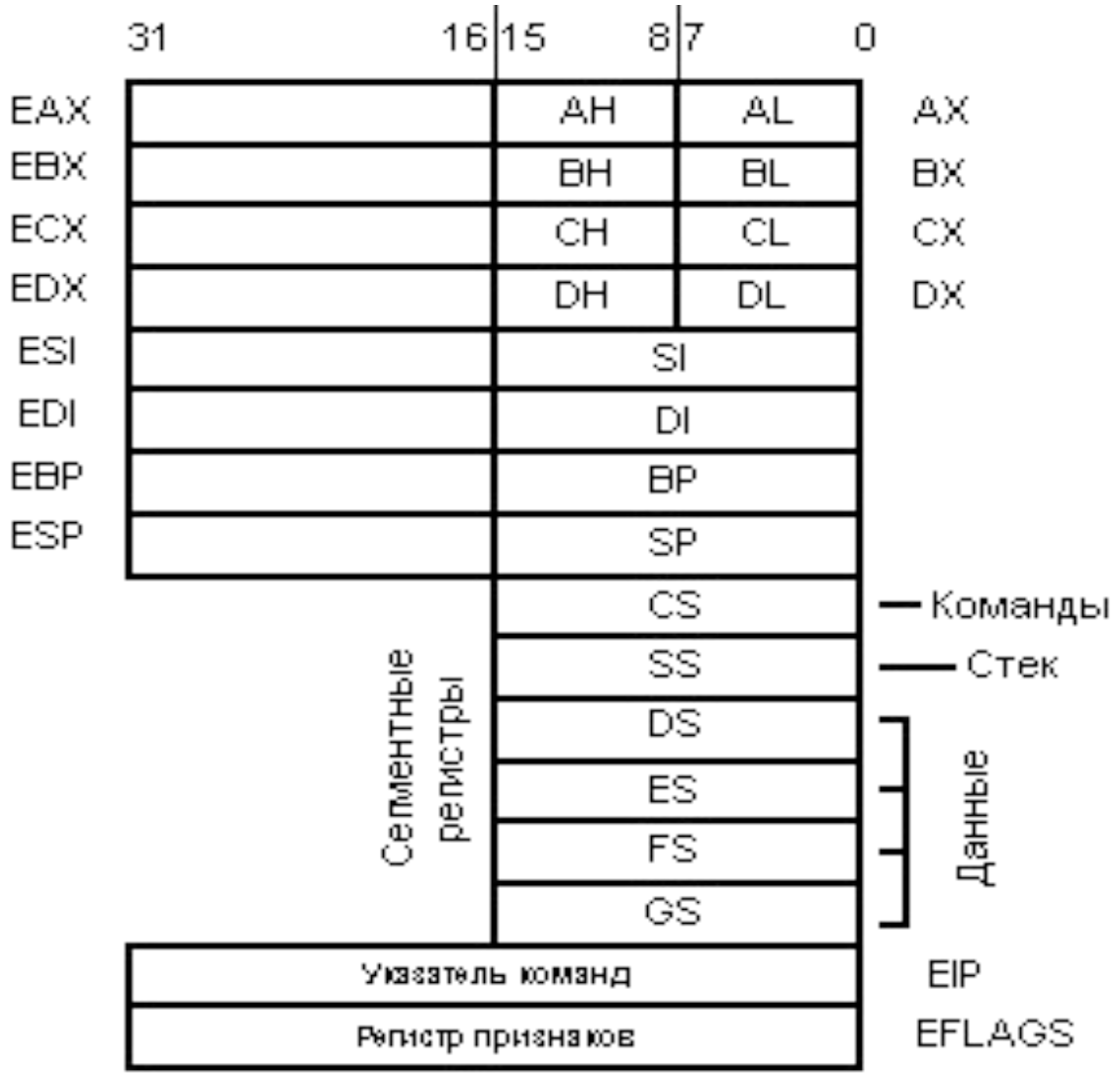
Указатель команд (регистр IP) всегда содержит смещение в памяти, по которому хранится следующая выполняемая команда. Когда выполняется одна команда, указатель команд перемещается таким образом, чтобы указывать на адрес памяти, по которому хранится следующая команда. Обычно следующей выполняемой командой является команда, хранимая по следующему адресу памяти, но некоторые команды, такие как вызовы или переходы, могут привести к тому, что в указатель команд будет загружено новое значение.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ip																

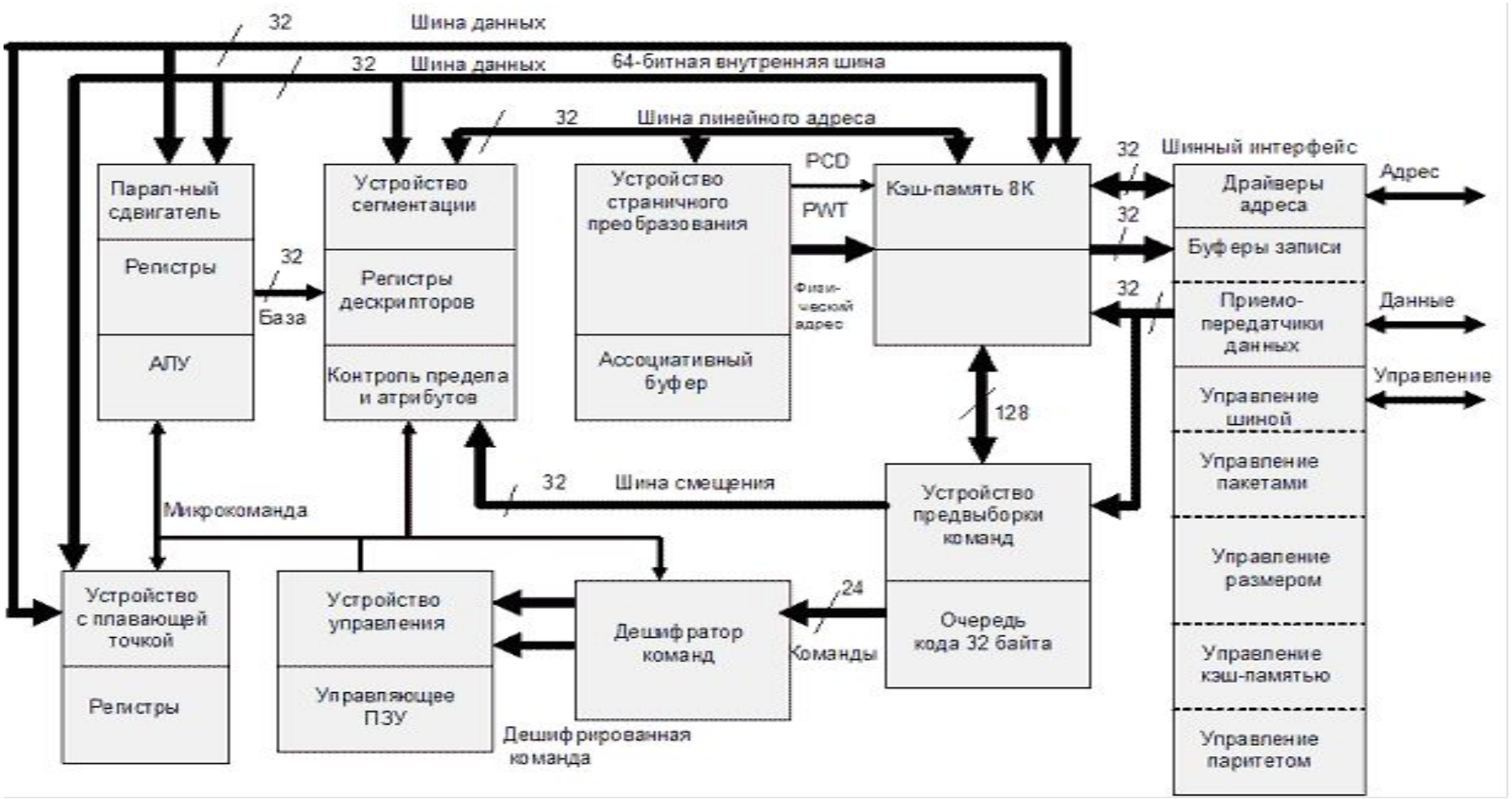
Структура процессора i386



Регистры процессора i386



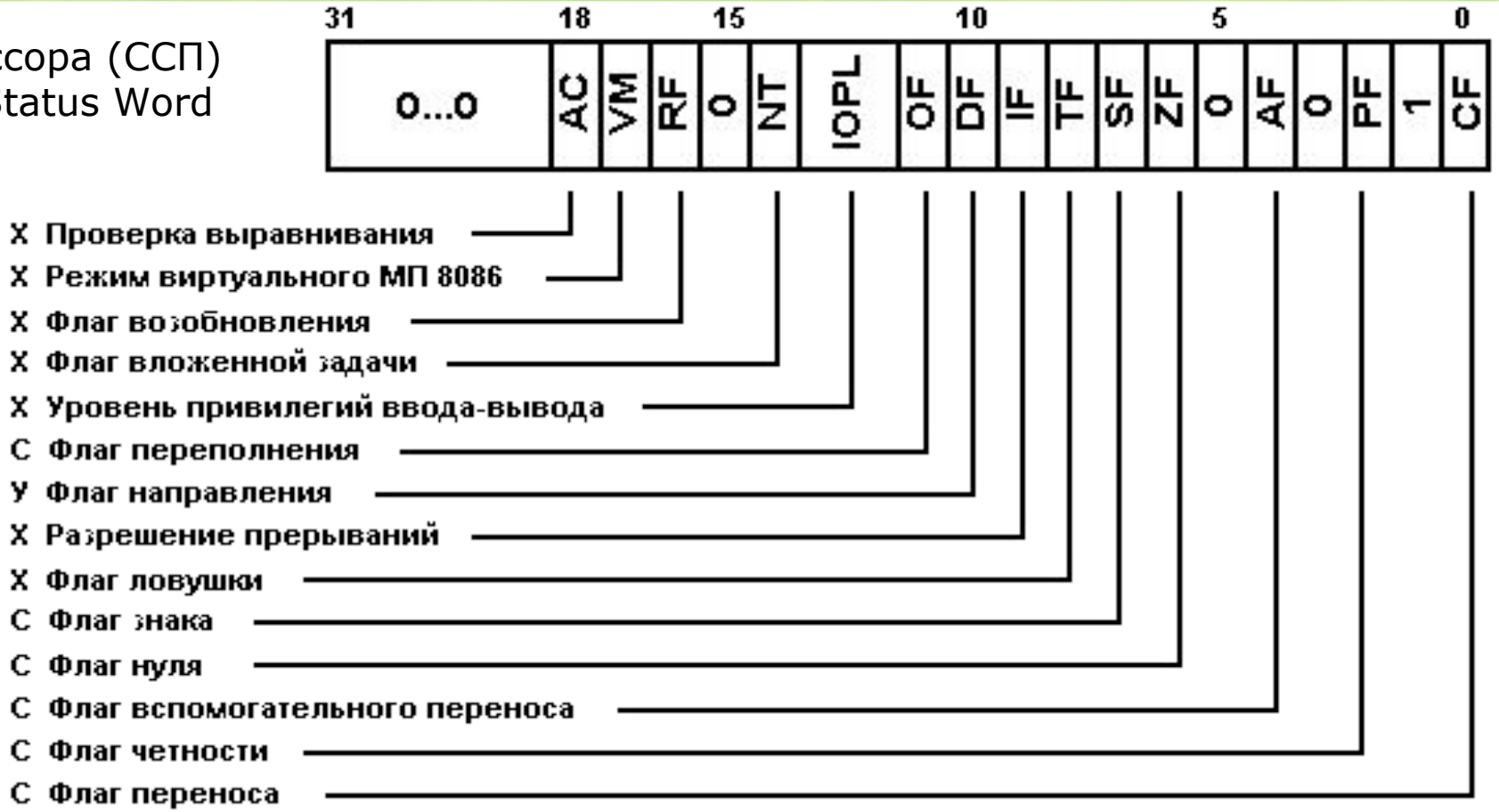
Структура процессора Intel 80486



Структура процессора Intel 80486

Регистр признаков IA-32 (PSW)

Слово состояния процессора (ССП)
англ. **PSW** — Processor Status Word



Структура регистров процессора IA-32 с плавающей точкой

Типы чисел 32-разрядного микропроцессора

Тип	Размер, байт	Диапазон	Обработка
Целые без знака	1	0...255	*АЛУ* ФТ
	2	0...65535	
	4	$0...4,3*10^{9}$	
Целые со знаком	1	-128...+127	*АЛУ* ФТ
	2	-32768...+32767	
	4	$2,1*10^{9}...+2,1*10^{9}$	
	8	$9,1*10^{18}...+9*10^{18}$	*FPU*
С плавающей точкой	4 (1+8+23)*мантисса*	$\pm 3,37*10^{35}$	*FPU*
	8 (1+11+52)	$\pm 1,67*10^{308}$ 308	
	10 (1+15+64)	$\pm 1,1*10^{4932}$	
Двоично- десятичные числа	1 распакованный	0...9	*АЛУ* ФТ
	1 упакованный	0...99	*АЛУ* ФТ
	10 упакованных	0..9...9(18 цифр)	*FPU*

IA-32

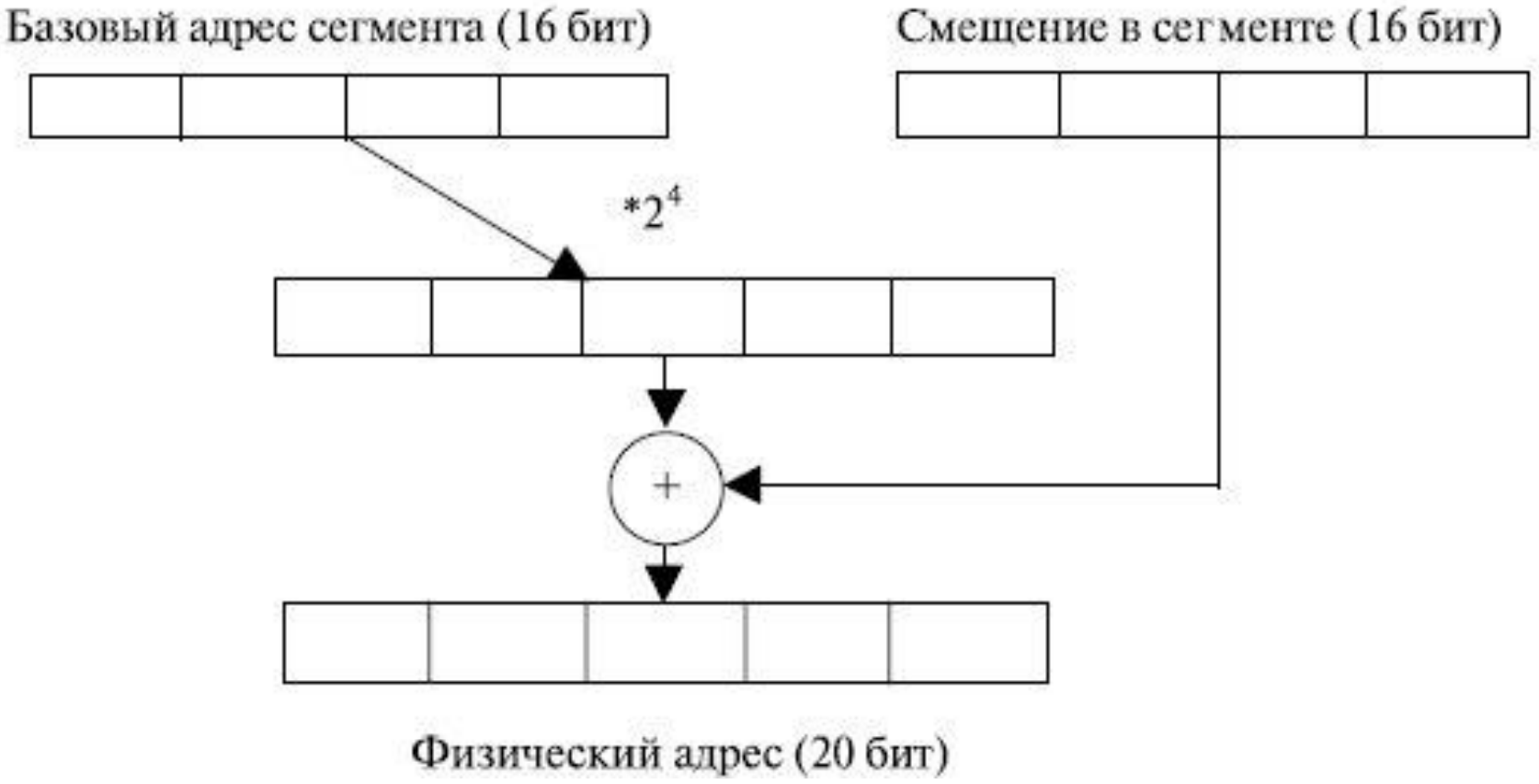
Read Address Mode (RM)– режим реального адреса. Однозадачность, отсутствие защиты и страничной организации, полностью совместимый с 8086, позволяющий адресовать до 1Мб физической памяти.

Protected mode – защищенный режим. Многозадачность. Механизм защиты на уровне сегментов. Поддержка подкачки страниц (paging).

Protected Virtual Address Mode (VM)– защищенный режим виртуальной адресации, позволяет для процессов 8086 адресовать до 4 Гбайт физической памяти, через которые при использовании механизма страничной адресации могут отображаться до 64 Тбайт виртуальной памяти для каждой задачи.

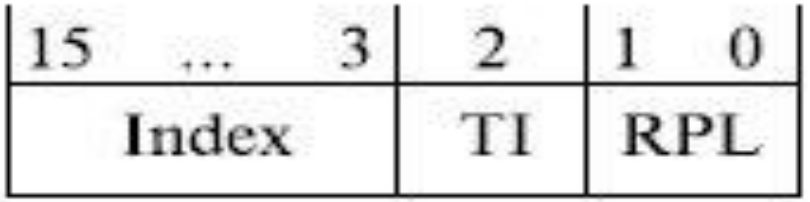
System Management Mode (SMM) - режим управления системой. Инициируется аппаратно. Используется для реализации системы управления энергосбережением.

Схема получения физического адреса в RM

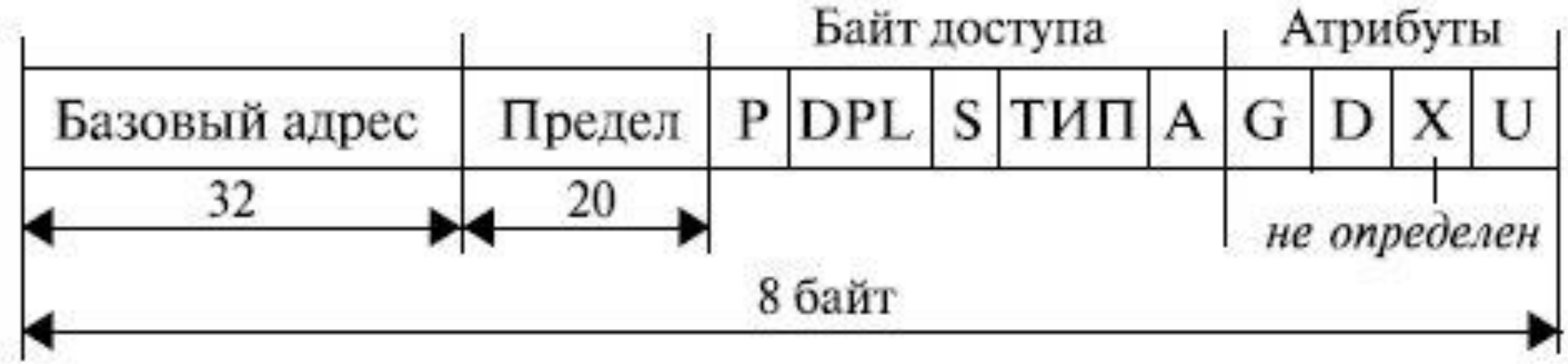


Структура регистров системных адресов и системных сегментов (РМ)

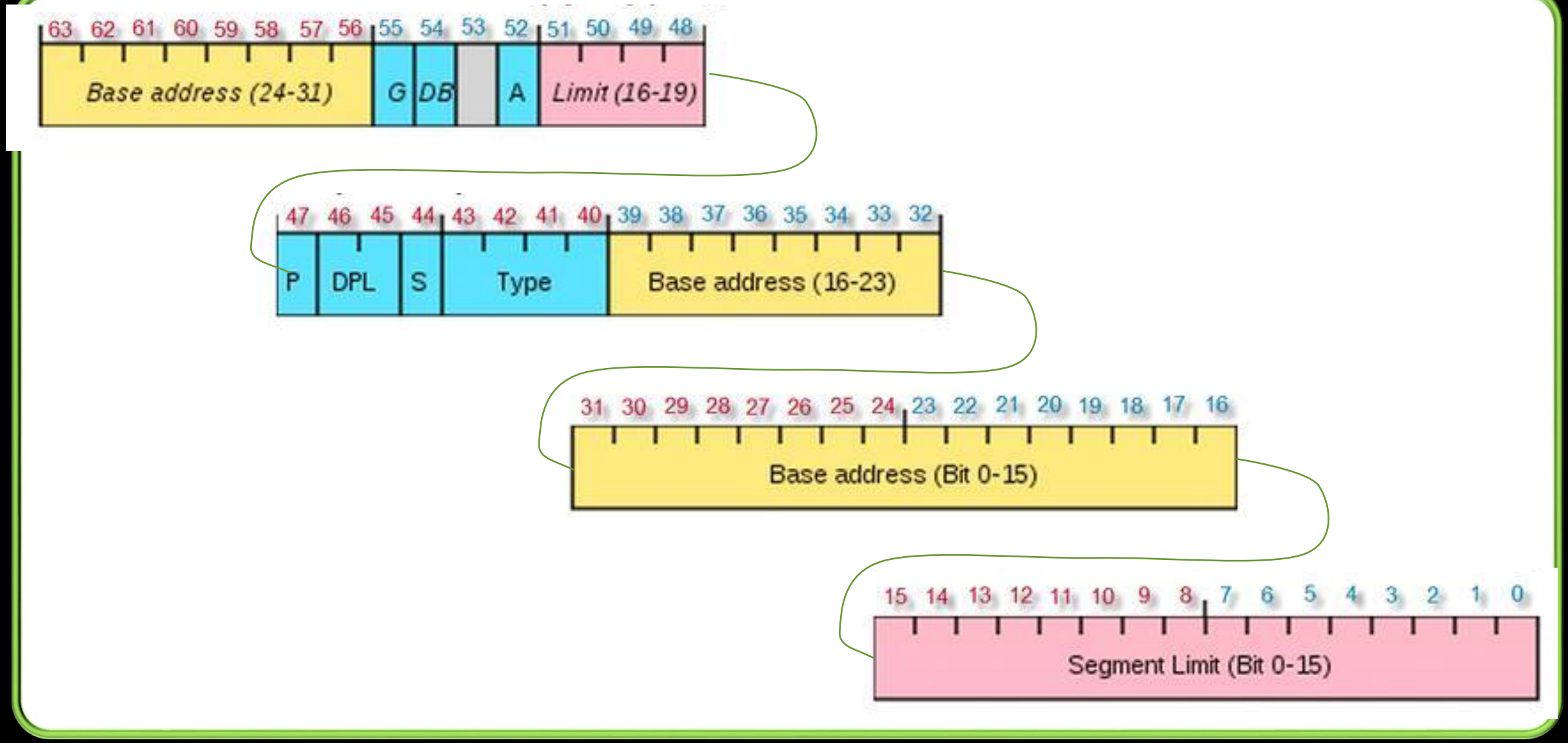
Структура селектора



Структура дескриптора сегмента



Формат дескриптора сегмента в МП



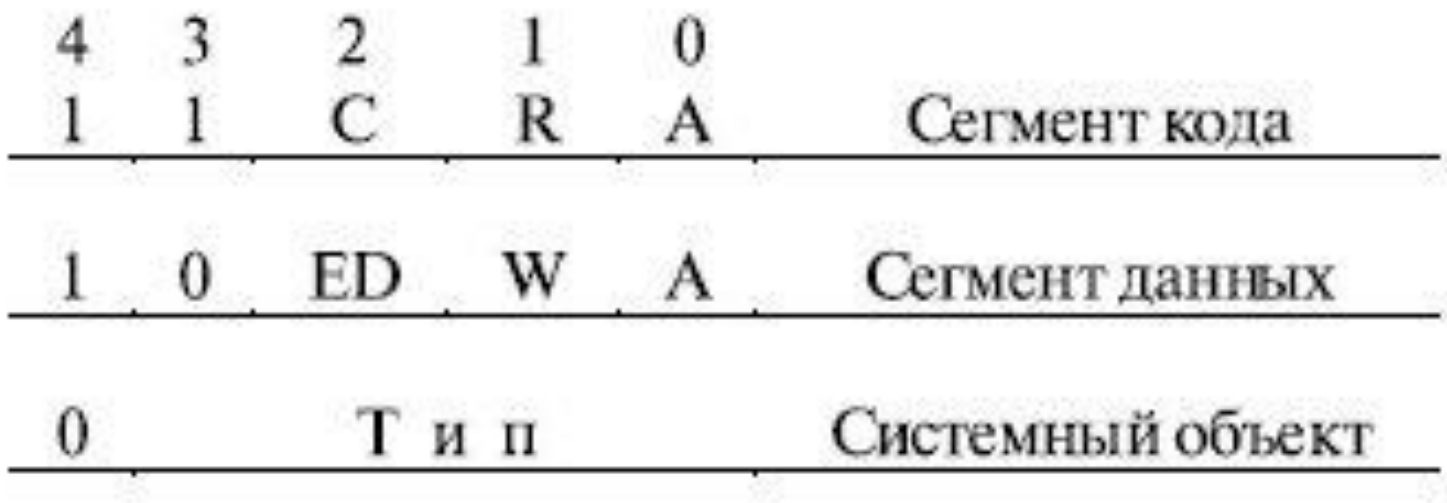
Размер сегмента в дескрипторе

$$G = \begin{cases} 0\text{-длина в байтах} \\ 1\text{-длина в страницах} \end{cases}$$

$$V_{\text{сегм макс}} = 2^{20}_{\text{стр}} * 2^{12}_{\text{байт}} = 2^{32}\text{байт}$$

$$D = \begin{cases} 0\text{-}16 \text{ разрядов} \\ 1\text{-}32 \text{ разряда} \end{cases}$$

Формирование типа физического адреса при сегментно-страничной организации памяти



1000 — сегмент данных, только считывание
1001 — сегмент данных, считывание и запись
1010 — сегмент стека, только считывание
1011 — сегмент стека, считывание и запись
1100 — сегмент кода, только выполнение
1101- сегмент кода, считывание и выполнение
1110 — подчиненный сегмент кода, только выполнение
1111 — подчиненный сегмент кода, только выполнение и считывание

Формат команды микропроцессора IA-32

ФОРМАТ КОМАНД МП 8086

Форматом команды называется распределение разрядов кода команды на группы. Число таких групп и их назначение зависит от типа микропроцессора. При любом формате команды обязательно наличие группы разрядов, называемой операционной частью команды или кодом операции (КОП).

Язык программирования наиболее полно учитывающий особенности "родного" микропроцессора и содержащий мнемонические обозначения машинных команд называется Ассемблером.

Текст программы на Ассемблере содержит:

а) команды или инструкции,
б) директивы или псевдооператоры,
в) операторы,
г) предопределенные имена

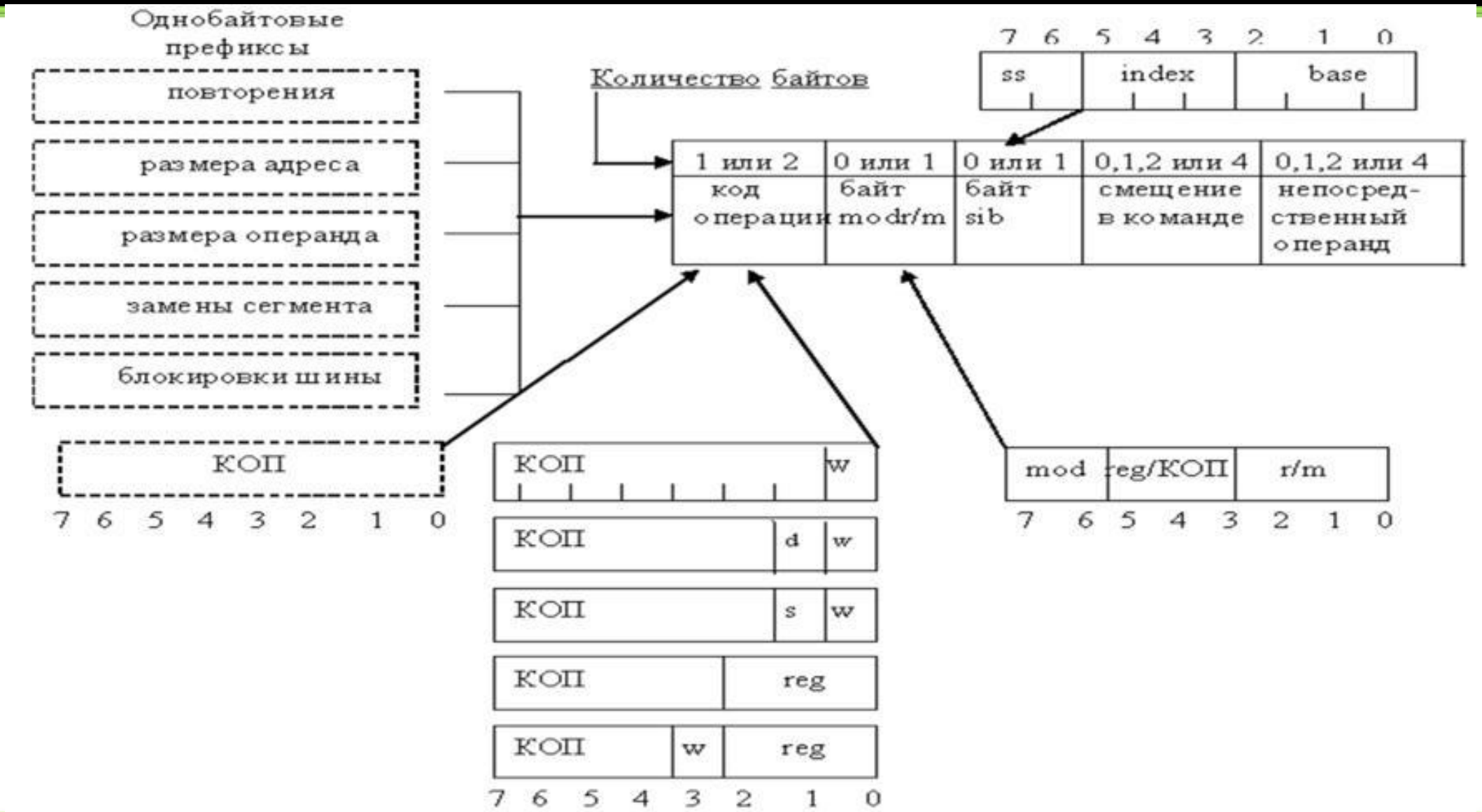
ФОРМАТ КОМАНД МП 8086

Действия обусловленные операциями перечисленными в пп. **б),в),г)** предыдущего слайда выполняются на этапе трансляции, т.е. являются командами Ассемблеру. Операции, называемые командами или инструкциями выполняются во время выполнения программы, т.е. являются командами микропроцессору.
Инструкция Ассемблера записывается на отдельной строке и включает до четырех полей, необязательные из которых выделены [ ]:

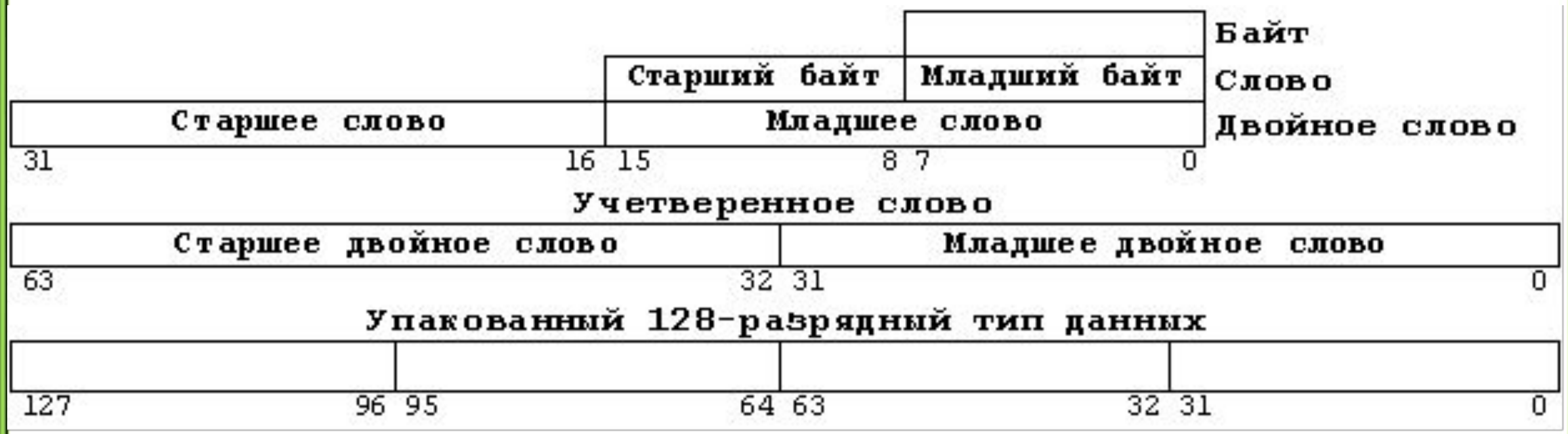
[метка:]	мнемоника_команды	[операнд(ы)]	[;комментарий]

Метка или символический адрес содержит до 31 символа из букв цифр и знаков ? @ . _ $. Причем цифра не должна стоять первой, а точка, если есть должна быть первой.
Мнемоника – сокращенное обозначение кода операции (КОП) команды, например мнемоника ADD обозначает сложение (addition).
Операндами могут быть явно или неявно задаваемые двоичные наборы, над которыми производятся операции

ФОРМАТ КОМАНД МП 8086



Основные типы данных микропроцессора

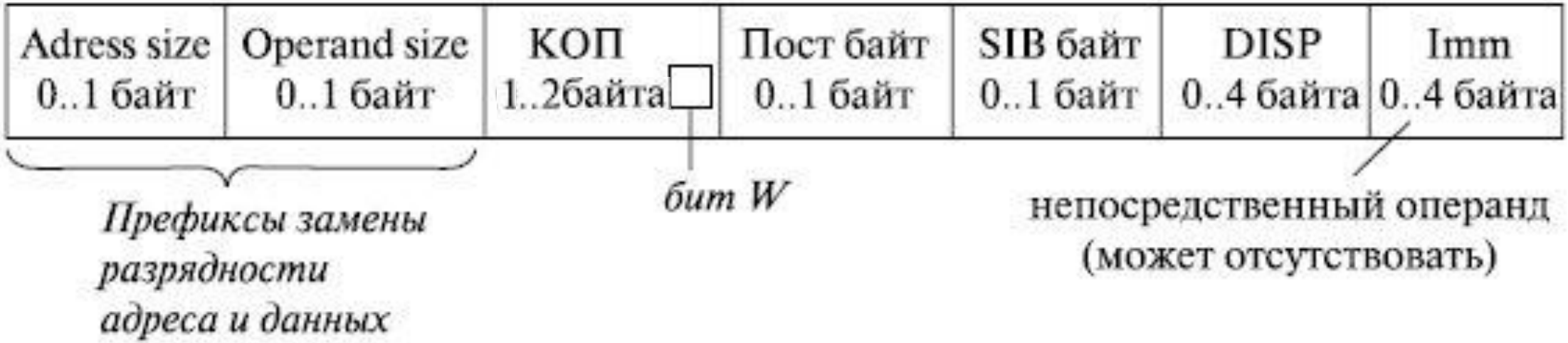


Формат команды микропроцессора IA-32

Инструкция микропроцессора может содержать следующие поля:

префикс	КОП	Mod R/M	SIB	Смещение	Непосредств. Операнд
0/1 байт	1/2 байта	0/1 байт	0/1 байт	0/1/2/4 байта	0/1/2/4 байта

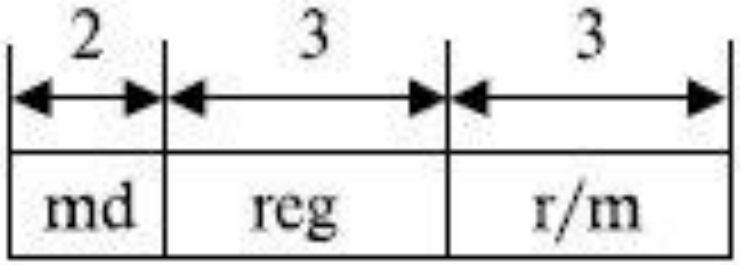
Формат команды 32-разрядного микропроцессора



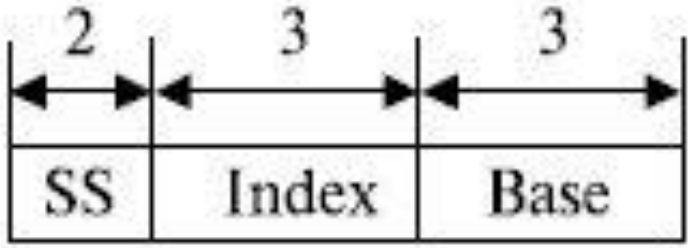
Формат команды 32-разрядного микропроцессора

Бит размерности D в дескрипторе сегмента	0	0	0	0	1	1	1	1
Префикс размерности Операнда*	-	-	+	+	-	-	+	+
Префикс размерности адреса*	-	+	-	+	-	+	-	+
Разрядность операнда (бит) **	16/8	16/8	32/8	32/8	32/8	32/8	16/8	16/8
Разрядность адреса (бит)	16	32	16	32	32	16	32	16

Формат команды 32-разрядного микропроцессора



Формат постбайта



Формат SIB-байта

$$(\text{смещение в сегменте}) = [base] + [index]^{ss} + disp$$

Смещение в сегменте (EA)

Смещение в сегменте (эффективный или исполнительный адрес - EA) может быть вычислено на основе значений регистров общего назначения и/или указанного в коде инструкции относительного смещения, при этом любой или даже несколько из указанных компонентов могут отсутствовать:

EA = BASE + (INDEX*SCALE) + DISPLACEMENT

Такая схема позволяет в языках высокого уровня и на языке Ассемблера легко реализовать работу с массивами.

Методы адресации

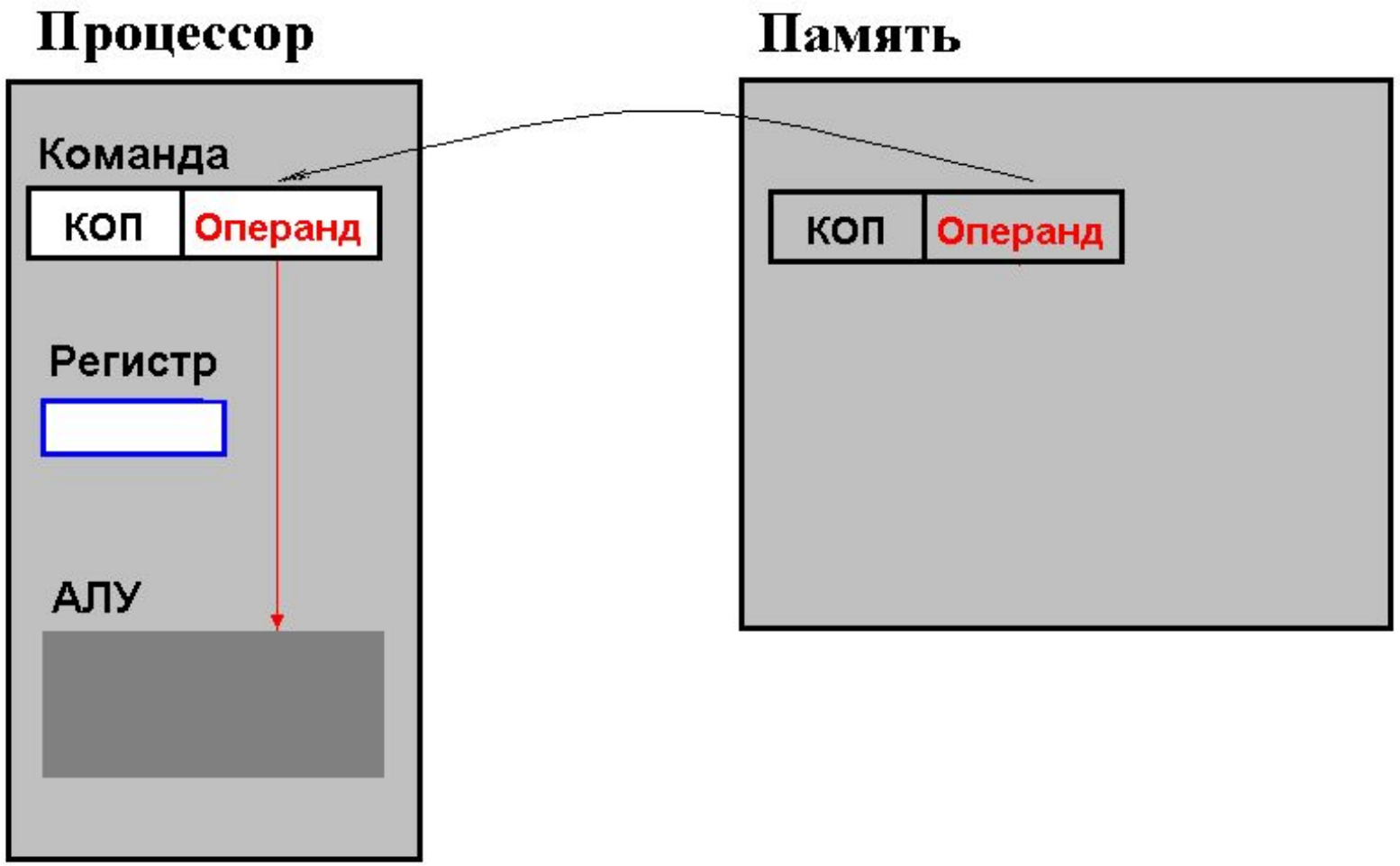
Режимы адресации

Режим	Адрес
Прямая адресация (Direct mode)	EA = Displacement
Косвенная регистровая адресация(Register Indirect Mode)	EA = Base
Базовая адресация (Based Mode)	EA = Base + Displacement
Индексная адресация (Index Mode)	EA = Index + Displacement
Масштабированная индексная адресация (Scaled Index Mode)	EA = Scale * Index + Displacement
Базово–индексная адресация (Based Index Mode)	EA = Base + Index
Масштабированная базово–индексная адресация (Based Scaled Index Mode)	EA = Base + Scale * Index
Базово–индексная адресация со смещением (Based Index Mode with Displacement)	EA = Base + Index
Масштабированная базово–индексная адресация со смещением (Based Scaled Index Mode with Displacement)	EA = Base + Scale * Index + Displacement

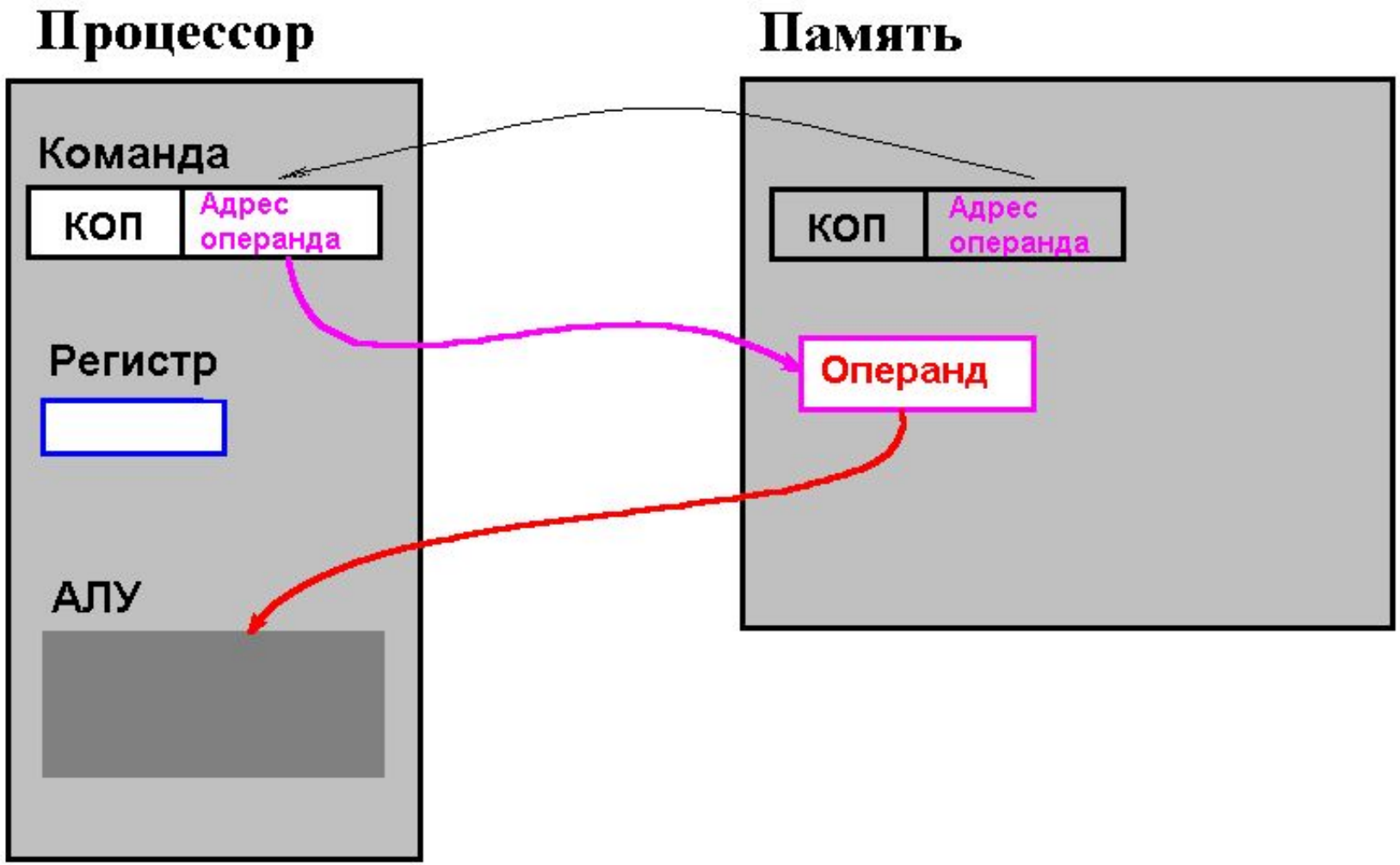
Адресация по Intel

- Непосредственная
- Прямая
- Регистровая (прямая)
- Косвенные:
  - Индексная
  - Базовая
  - Базово-индексная
  - Индексная со смещением
  - Базовая со смещением
  - Базово-индексная со смещением

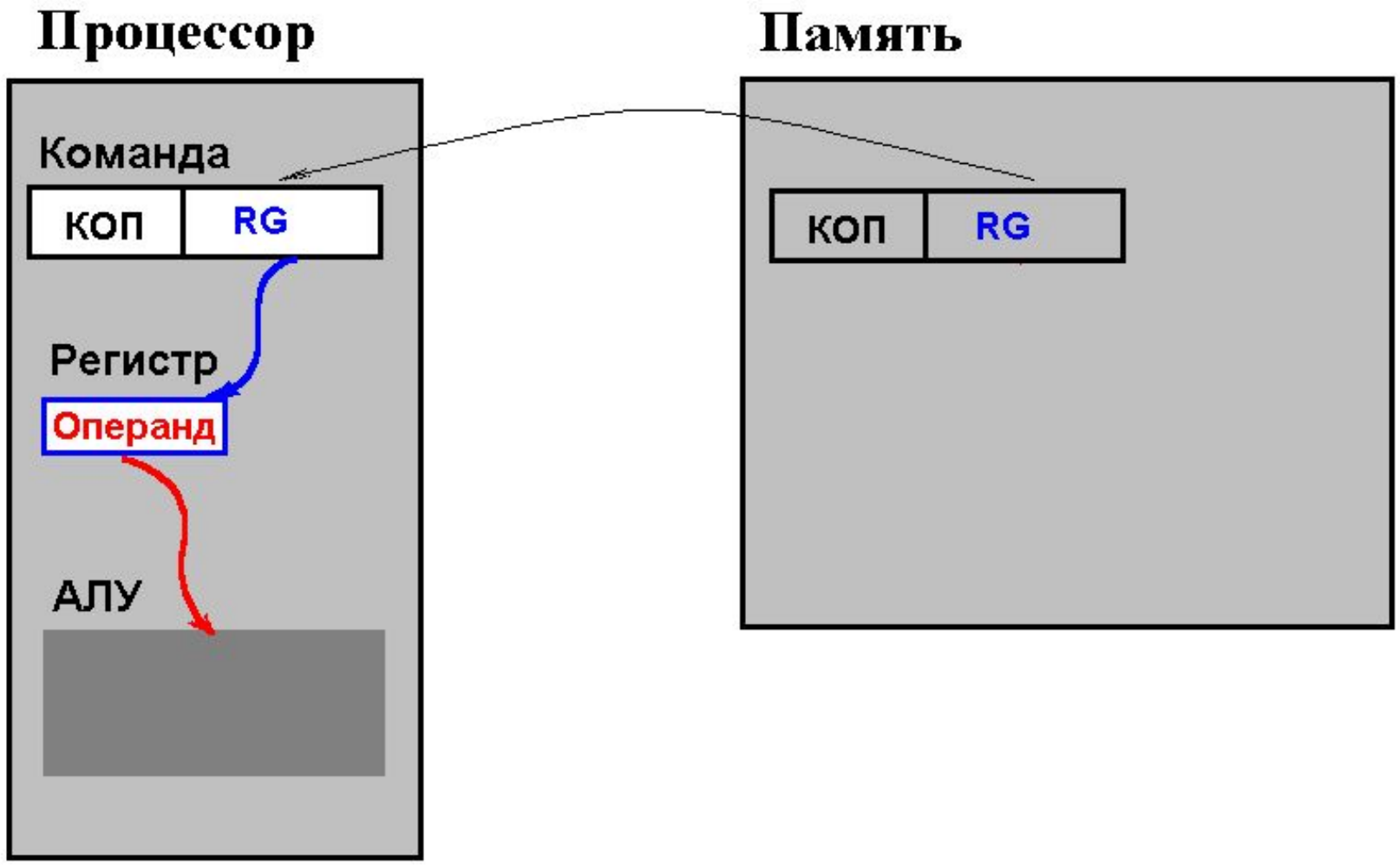
Непосредственная адресация



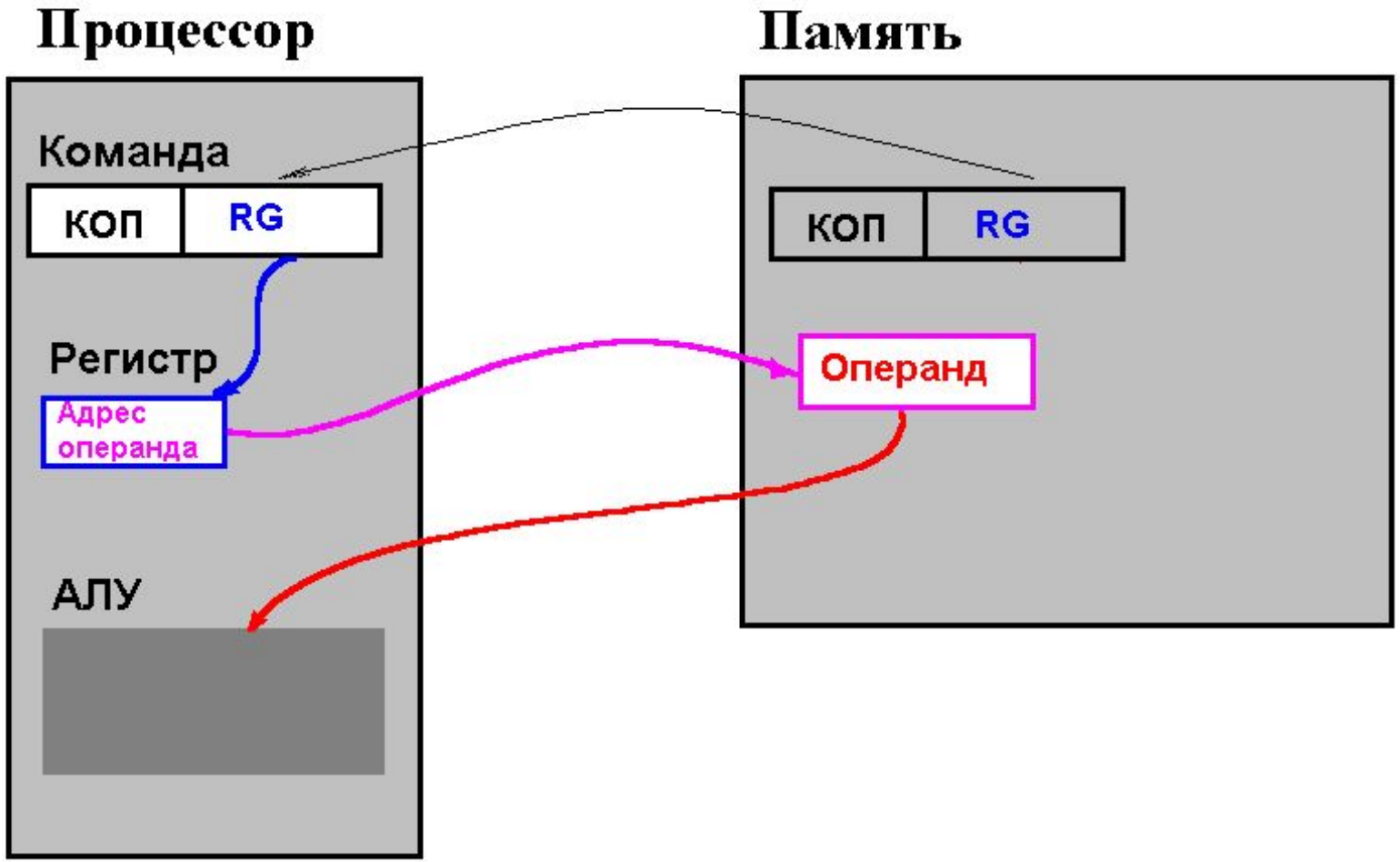
Прямая адресация



Регистровая адресация

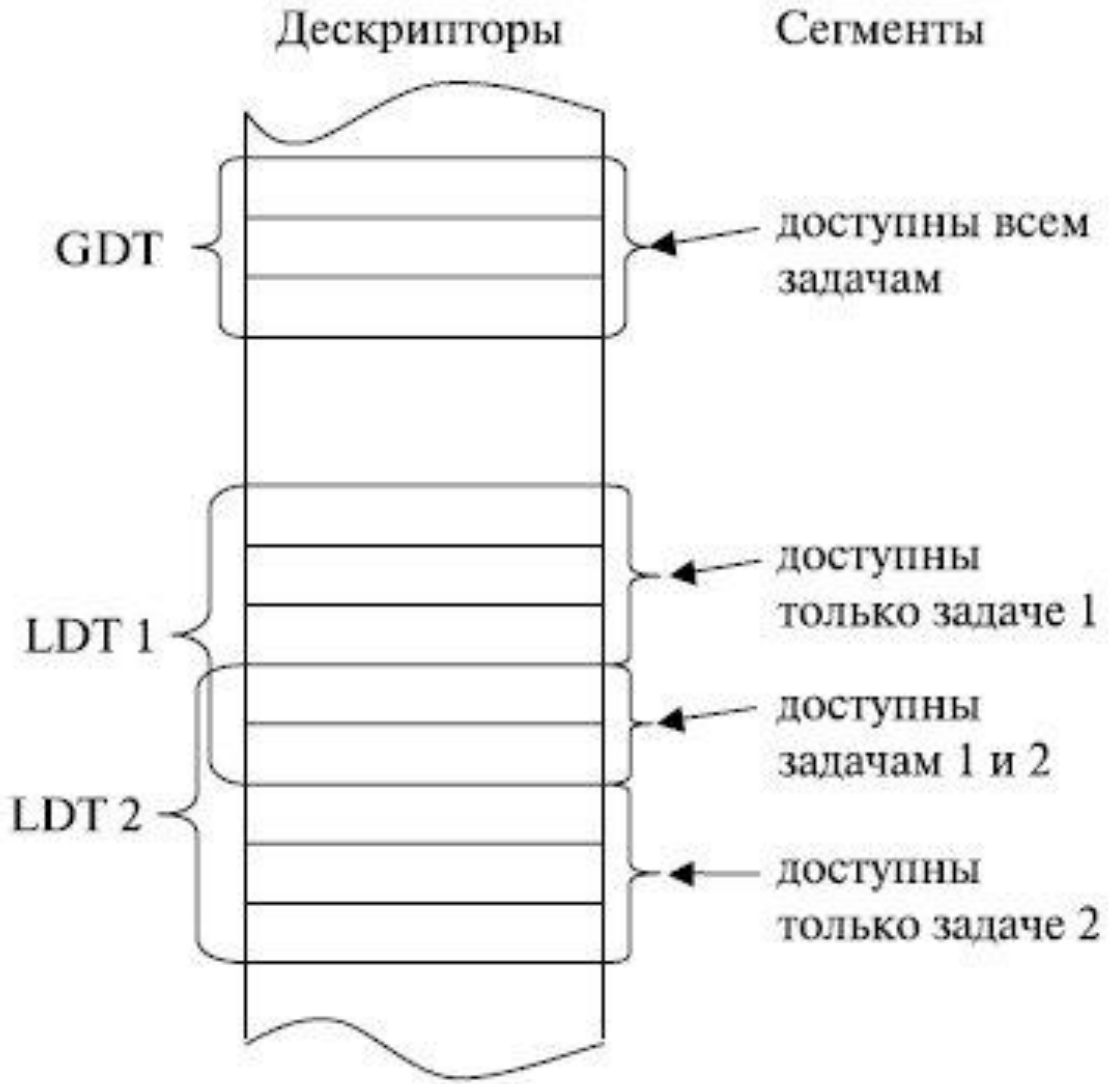


Косвенная адресация



Сегментная адресация памяти

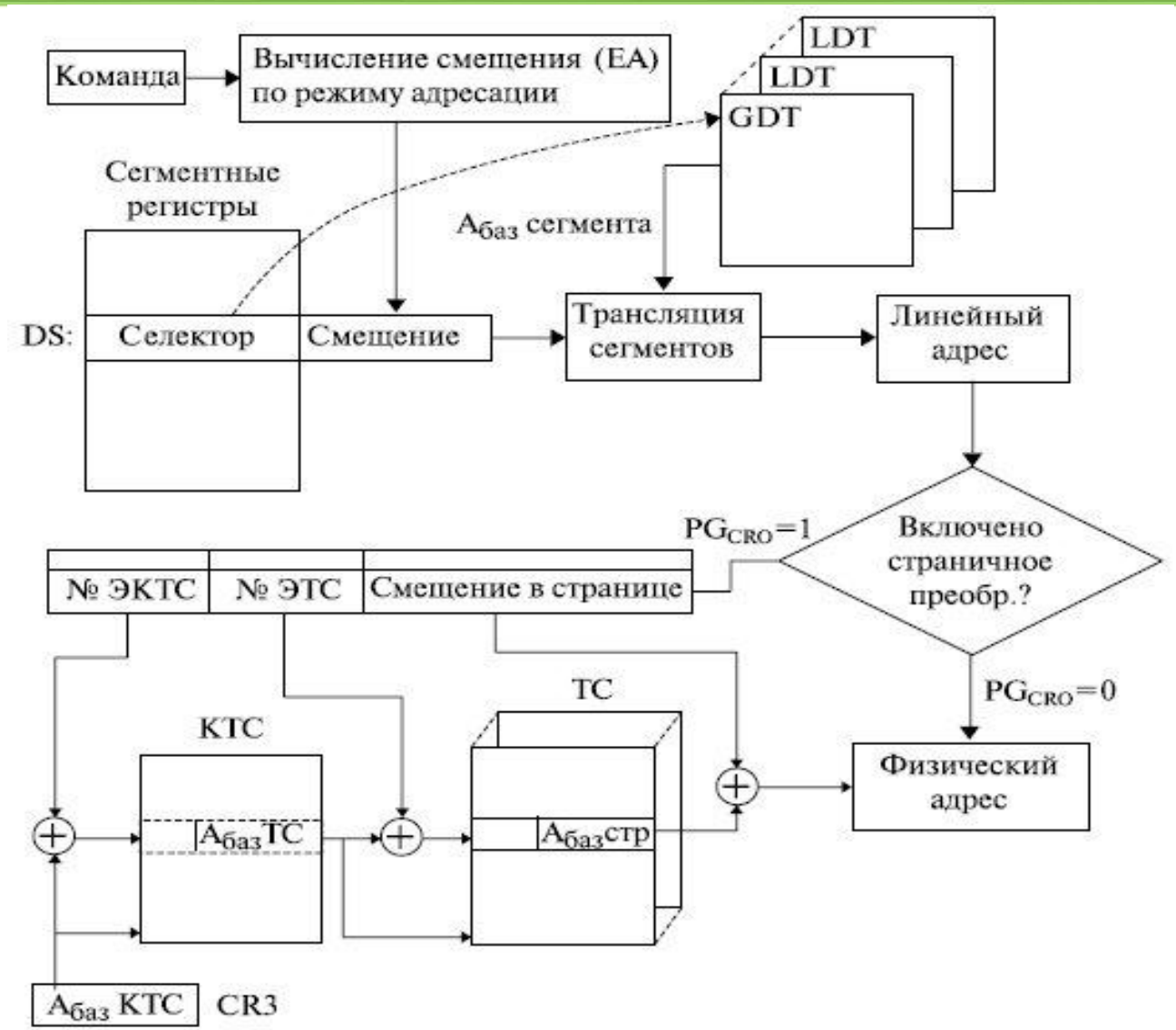
Описание сегментов в таблицах дескрипторов



15 ... 3	2	1 0
Index	TI	RPL

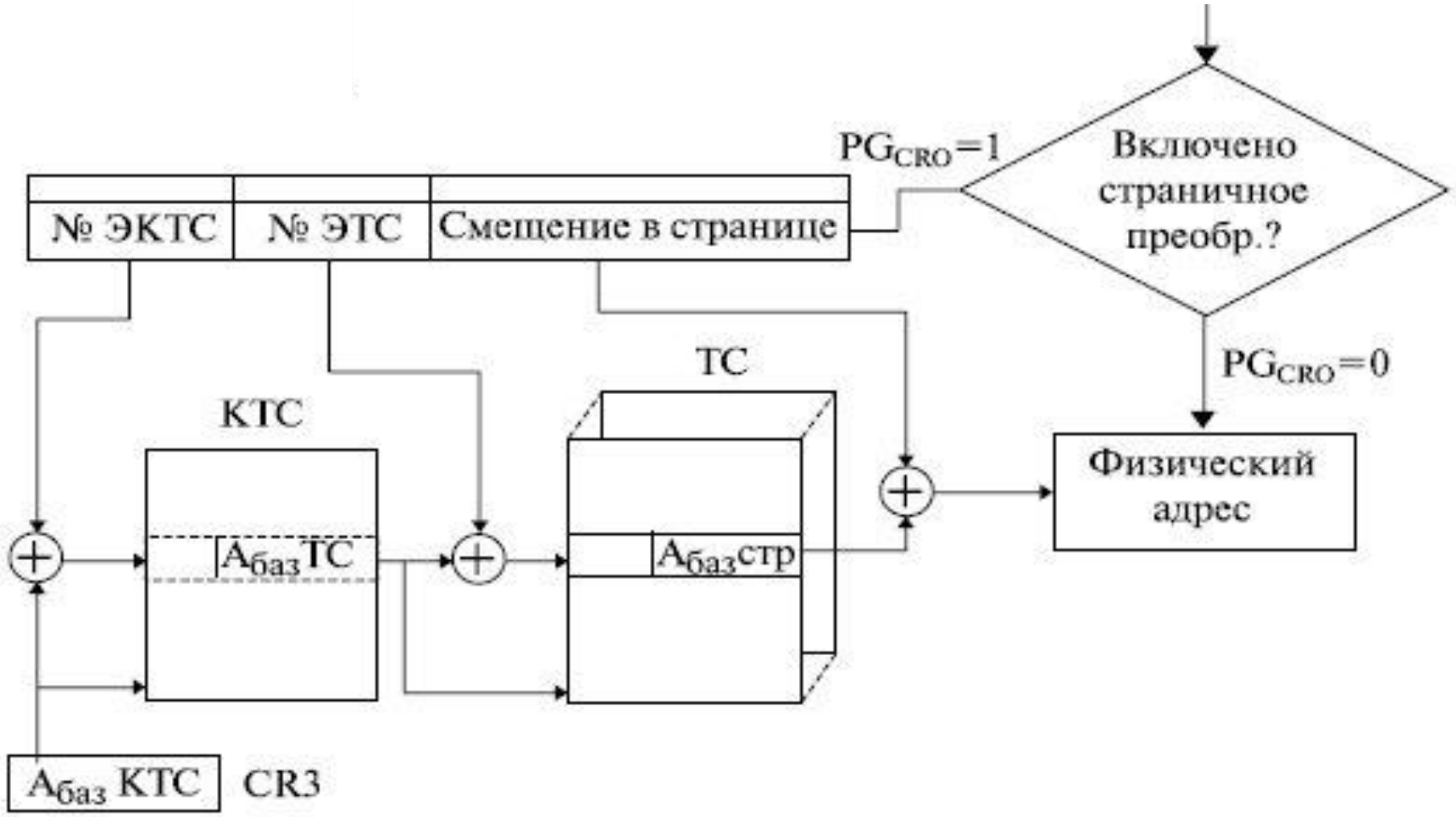
Формат селектора

Формирование физического адреса при сегментно-страничной

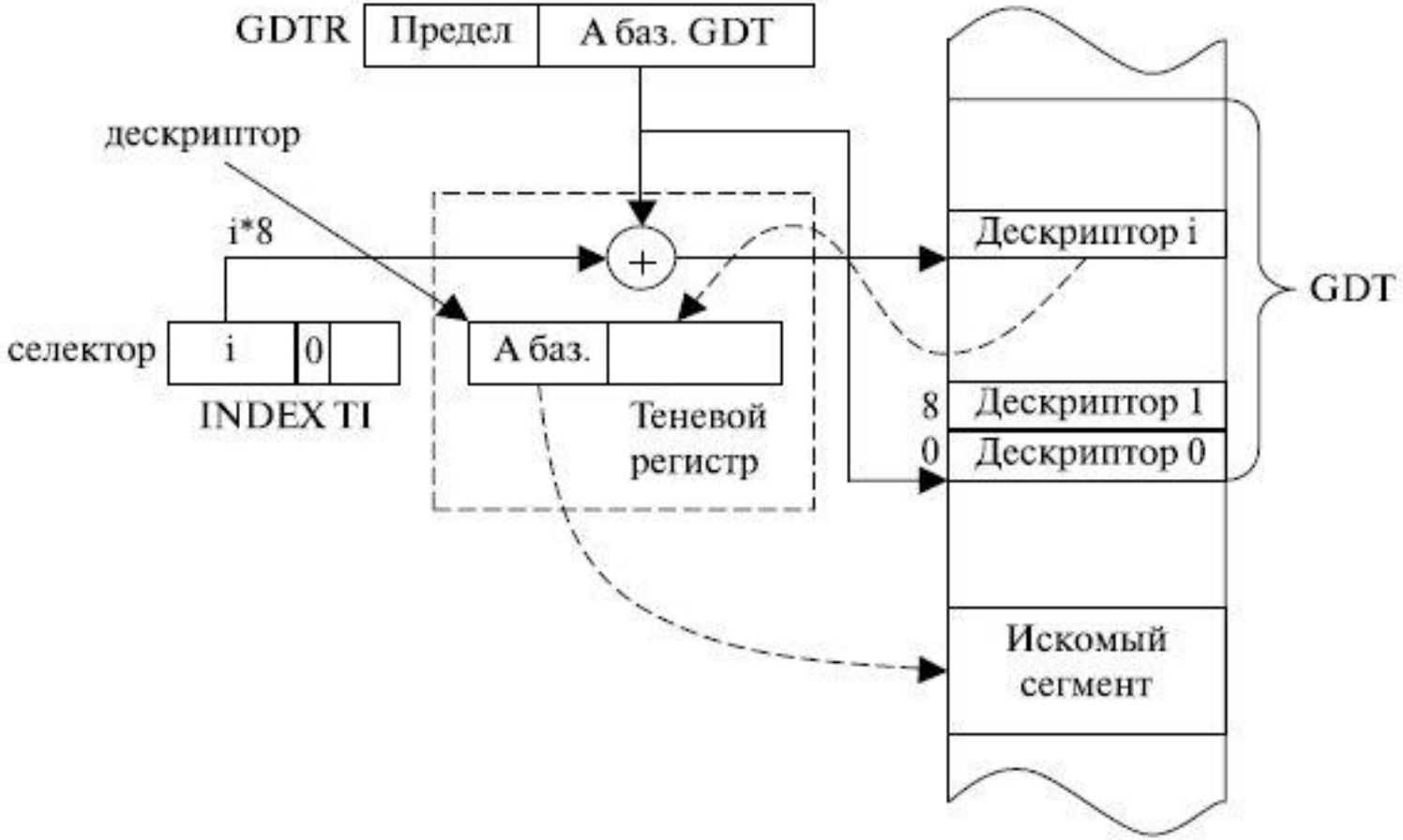


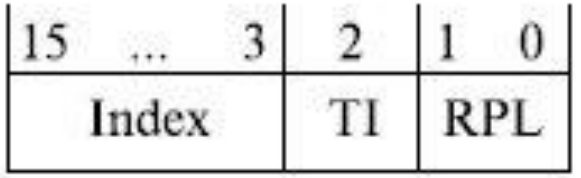
Формирование физического адреса при сегментно-страничной организации

Формирование физического адреса при сегментно-страничной организации



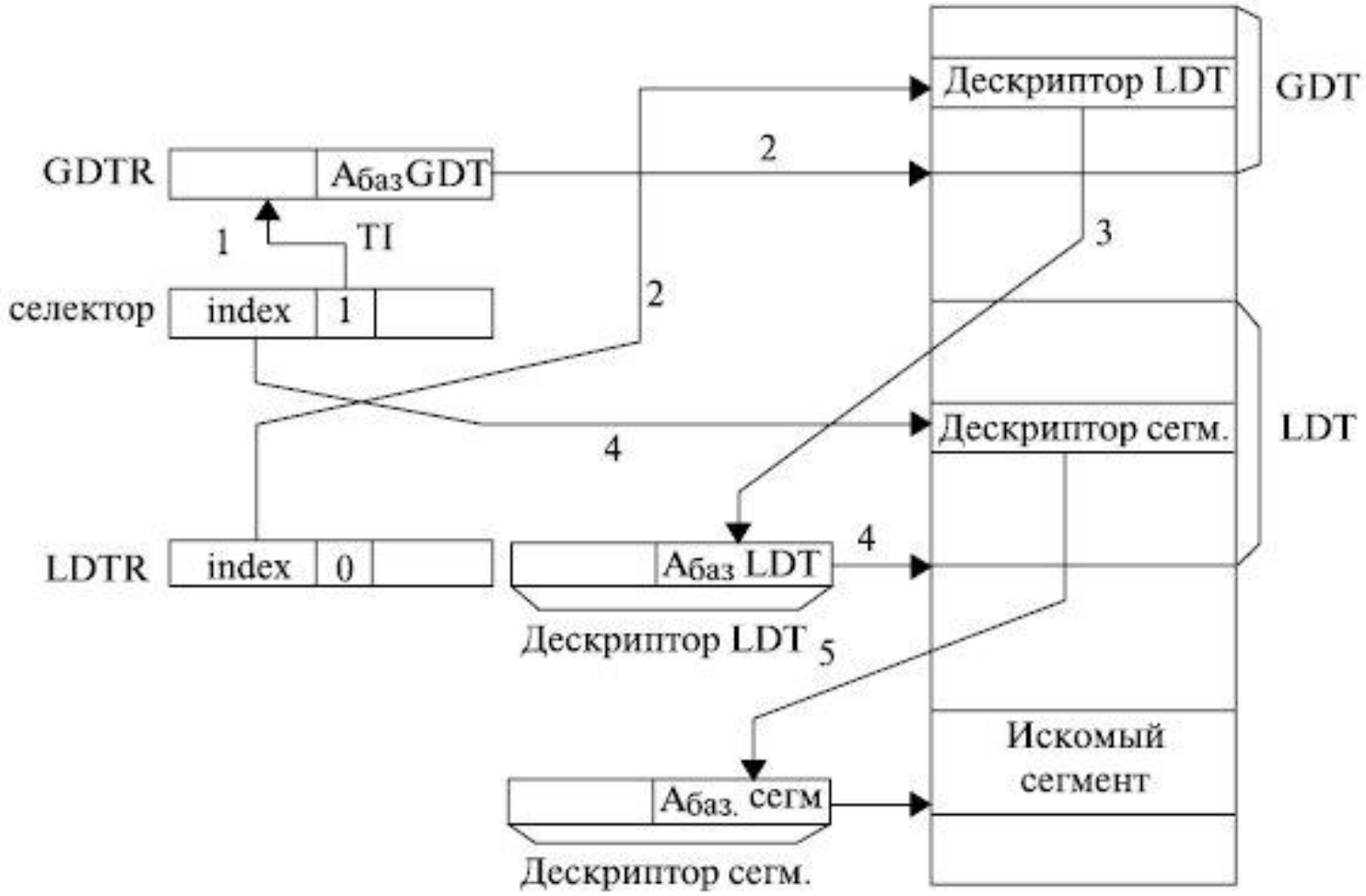
Получение дескриптора, находящегося в глобальной таблице дескрипторов GDT

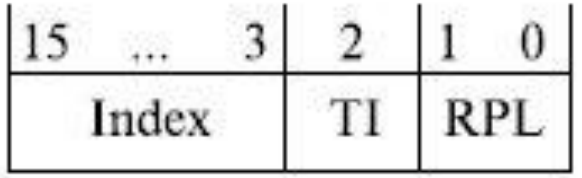




Формат селектора

Получение дескриптора, находящегося в локальной таблице дескрипторов LDT





Формат селектора

получения адреса операнда на примере команды

MOV EAX, [ECX+*ESI*+20h]

селектор *по* умолчанию находится в *сегментном регистре* *DS*

Пусть (*DS*) = 0000 0000 0001 10XX

1) Образовать *эффективный адрес* (вычислить смещение в сегменте):

EA = (ECX)+(*ESI*)+20h

2) Выбрать 3-й дескриптор (Index = 3) из *GDT* (TI = 0)

Для этого:

a) считать базовый адрес глобальной таблицы дескрипторов ($A_{базGDT}$) из GDTR

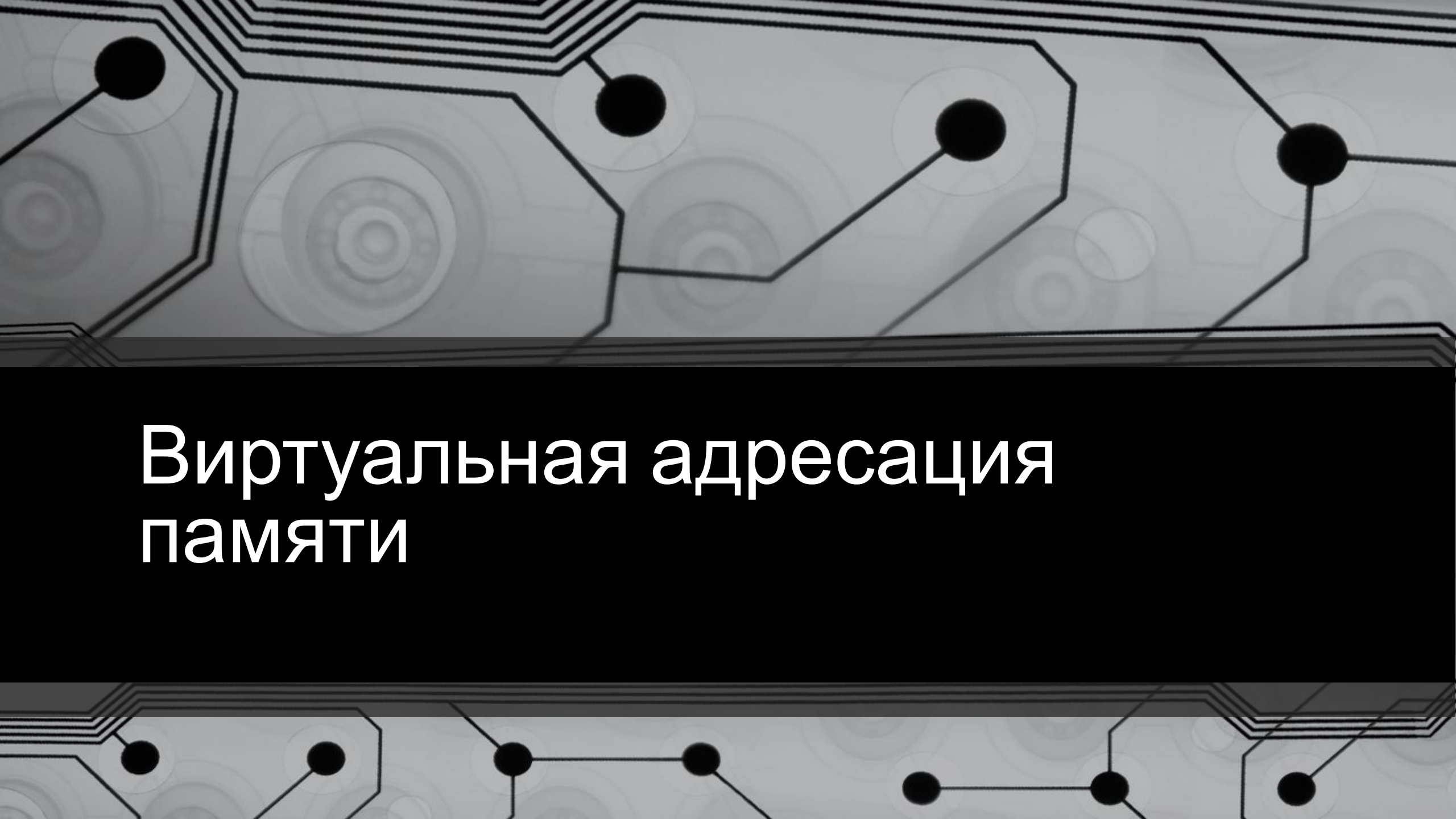
b) вычислить $A_{базGDT} + (Index) * 8$

c) обратиться по полученному адресу в память и считать нужный дескриптор

3) Получить **линейный адрес:** $ЛА = EA + A_{баздескр.3}$, где $A_{баздескр.3}$ -базовый адрес сегмента из считанного дескриптора с номером 3

4) Так как при сегментной организации адресного пространства **линейный адрес** равен физическому, следует обратиться к памяти по сформированному адресу и передать двойное **слово** в EAX

При TI = 1 потребовалось бы еще одно обращение к памяти для считывания дескриптора *LDT* из *GDT*

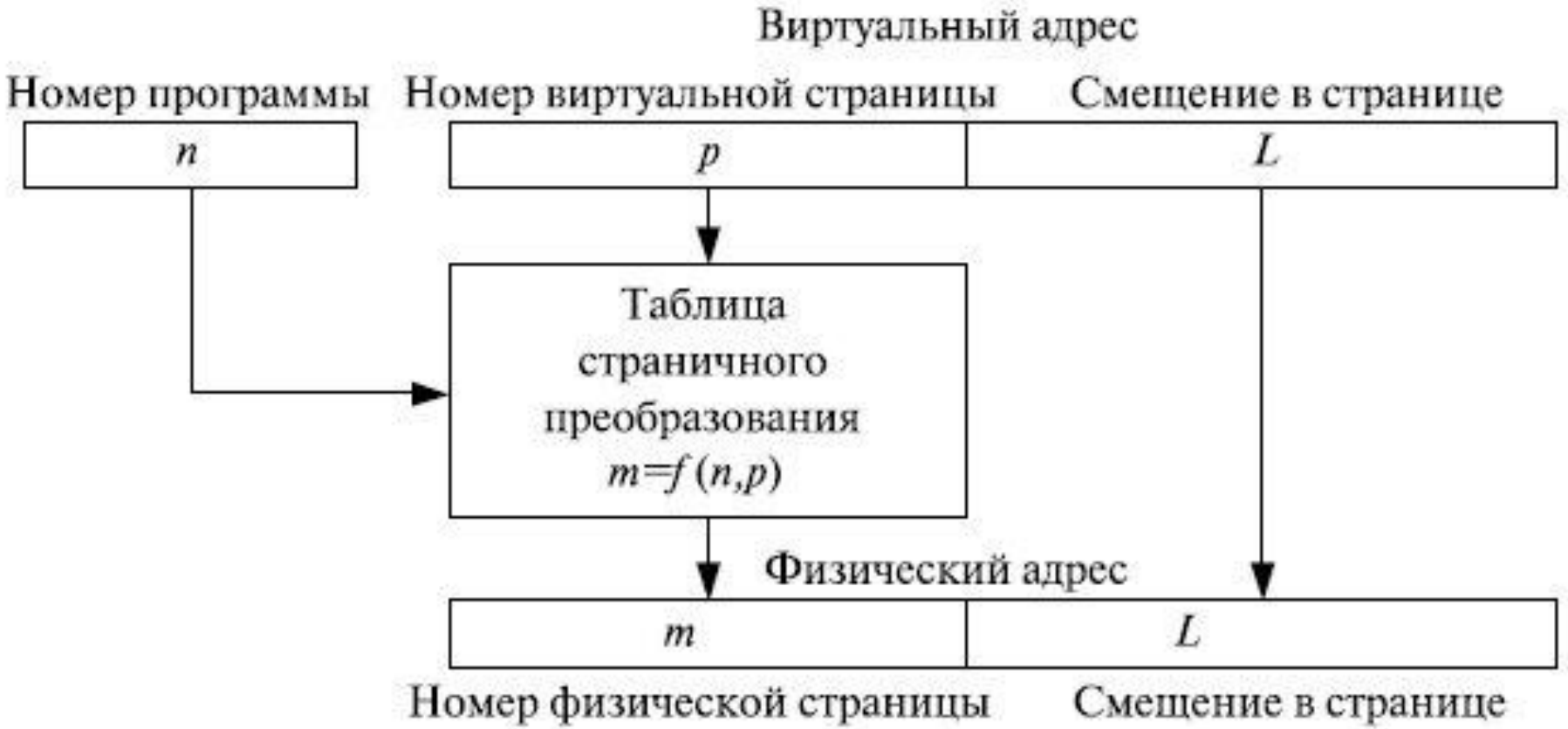


Виртуальная адресация памяти

Принцип преобразования виртуального страничного адреса в

$V_{\text{вирт}} >> V_{\text{физ}} V_{\text{вирт}} = 2^{L\text{ша}}$

$V_{\text{вирт}} = 2^{32} = 4\text{Гбайт}$



Пример преобразования адреса виртуальной страницы в адрес физической страницы.

Пусть *компьютер* использует *адресное пространство*, предполагающее *разбиение* на страницы объемом:

$$V_{стр} = 1i$$

и имеет оперативную *память в* страницах:

$$V_{озу} = 3$$

Пусть на компьютере одновременно выполняются четыре программы, имеющие следующее количество страниц:

$$V^A = 2, V^B = 1, V^C = 3, V^D = 2.$$

Переключение между программами происходит через время кванта

$$t^k = 1.$$

Время выполнения каждой страницы любой программы составляет

$$t = 3t^k.$$

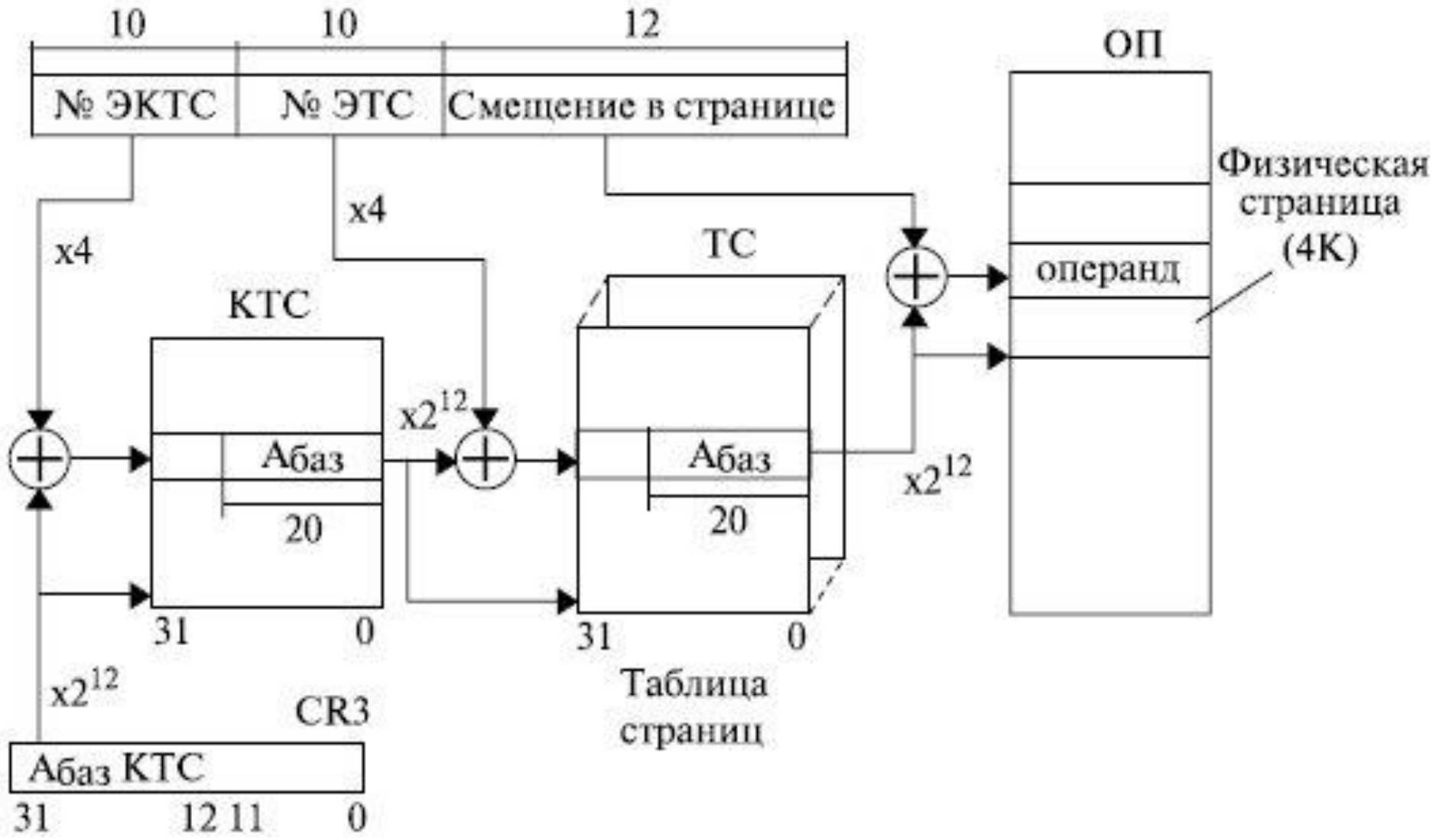
Полагаем, что страницы программ загружаются в оперативную *память по* мере необходимости и *по* возможности в свободные области *ОЗУ*.

Если вся *память* занята, то новая страница замещает ту, к которой дольше всего не было обращений.

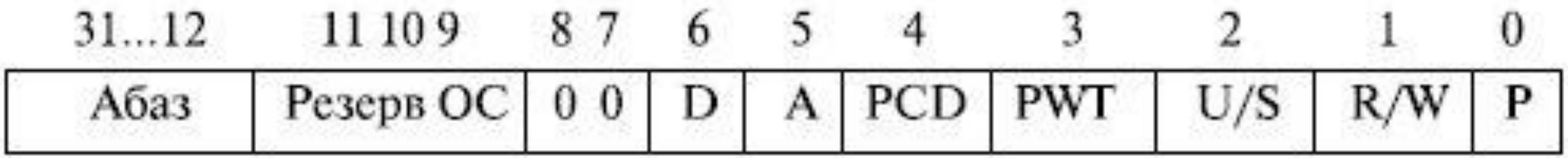
Пример страничного распределения памяти в мультипрограммной ЭВМ

Страница	Такты															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	Динамическое распределение оперативной памяти															
ОЗУ 0	A0	A0	A0	D0	D0	D0	C0	C0	C0	C1	C1	C1	C1	C1	C1	C1
1		B0	B0	B0	A0	A0	A0	D0	D0	D0	D1	D1	D1	D1	D1	D1
2			C0	C0	C0	B0	B0	B0	A1	A1	A1	A1	A1	A1	C2	C2
	Таблица страничного преобразования для программы A															
A 0	0	0	0	-	1	1	1	-	-	-	-	-	-	-	-	-
1	-	-	-	-	-	-	-	-	2	2	2	2	2	2	-	-
	Таблица страничного преобразования для программы B															
B 0	-	1	1	1	-	2	2	2	-	-	-	-	-	-	-	-
	Таблица страничного преобразования для программы C															
C 0	-	-	2	2	2	-	0	0	0	-	-	-	-	-	-	-
1	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	2
	Таблица страничного преобразования для программы D															
D 0	-	-	-	0	0	0	-	1	1	1	-	-	-	-	-	-
1	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1	1

Страничное преобразование линейного адреса в физический



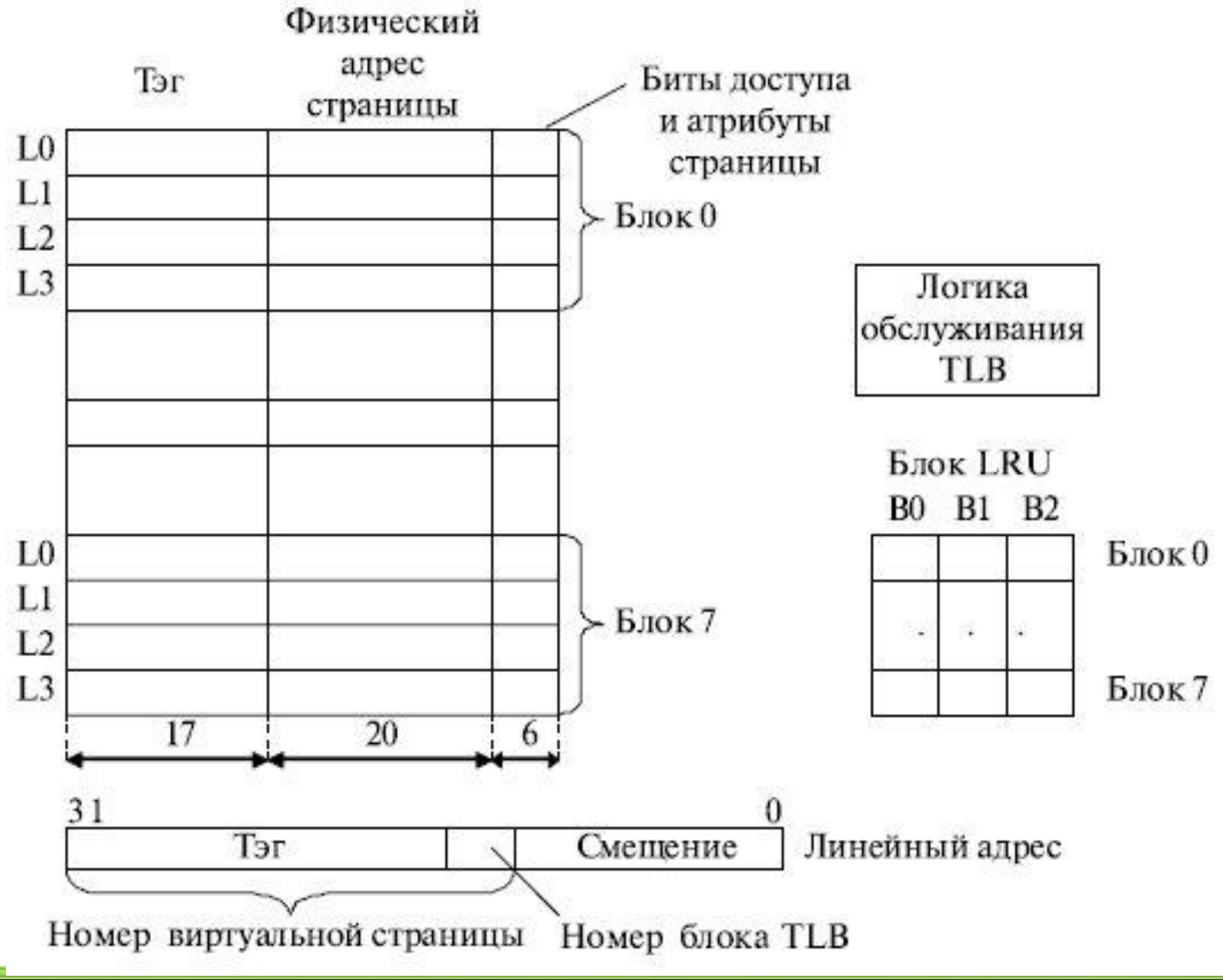
Структура элементов каталога таблиц страниц и таблицы страниц



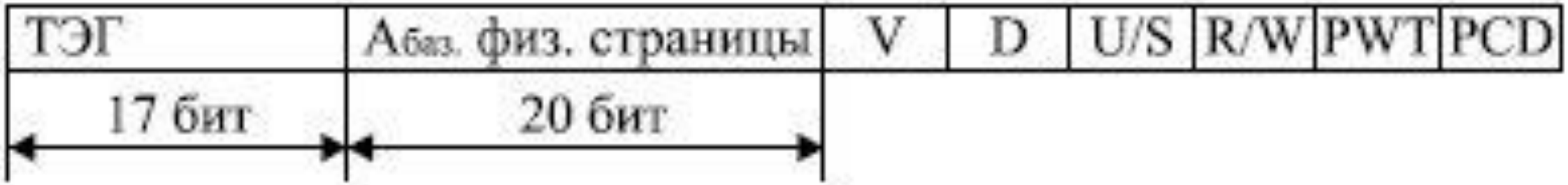
Допустимые действия со страницами на различных уровнях привилегий

U/S	R/W	Допустимо для уровня 3	Допустимо для уровней 0, 1, 2
0	X	Ничего	Чтение/запись
1	0	Чтение	Чтение/запись
1	1	Чтение/запись	Чтение/запись

Структура буфера TLB ассоциативной трансляции страничного адреса



Формат строки модуля основной памяти TLB

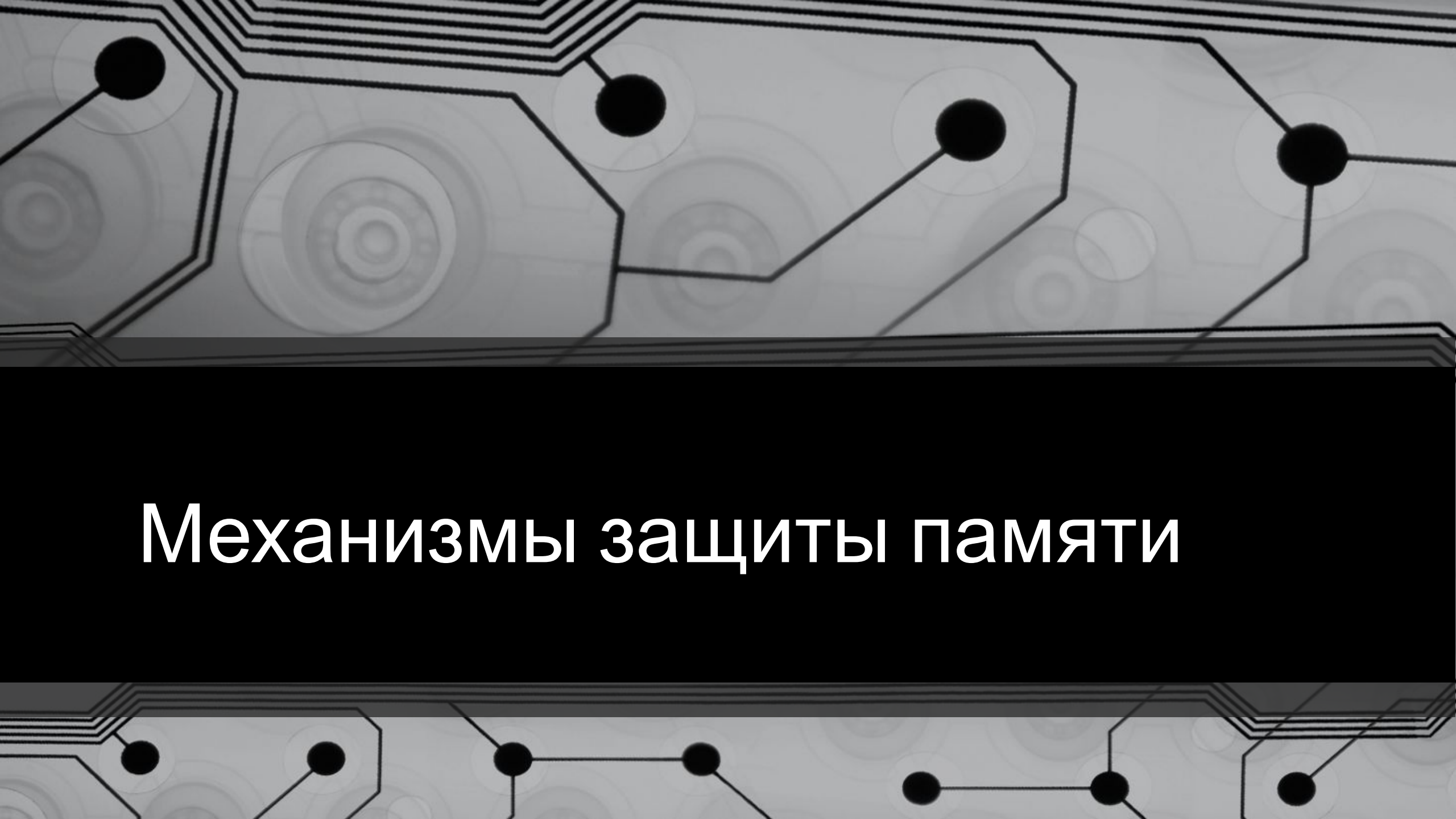


Порядок изменения бит в строке *LRU*

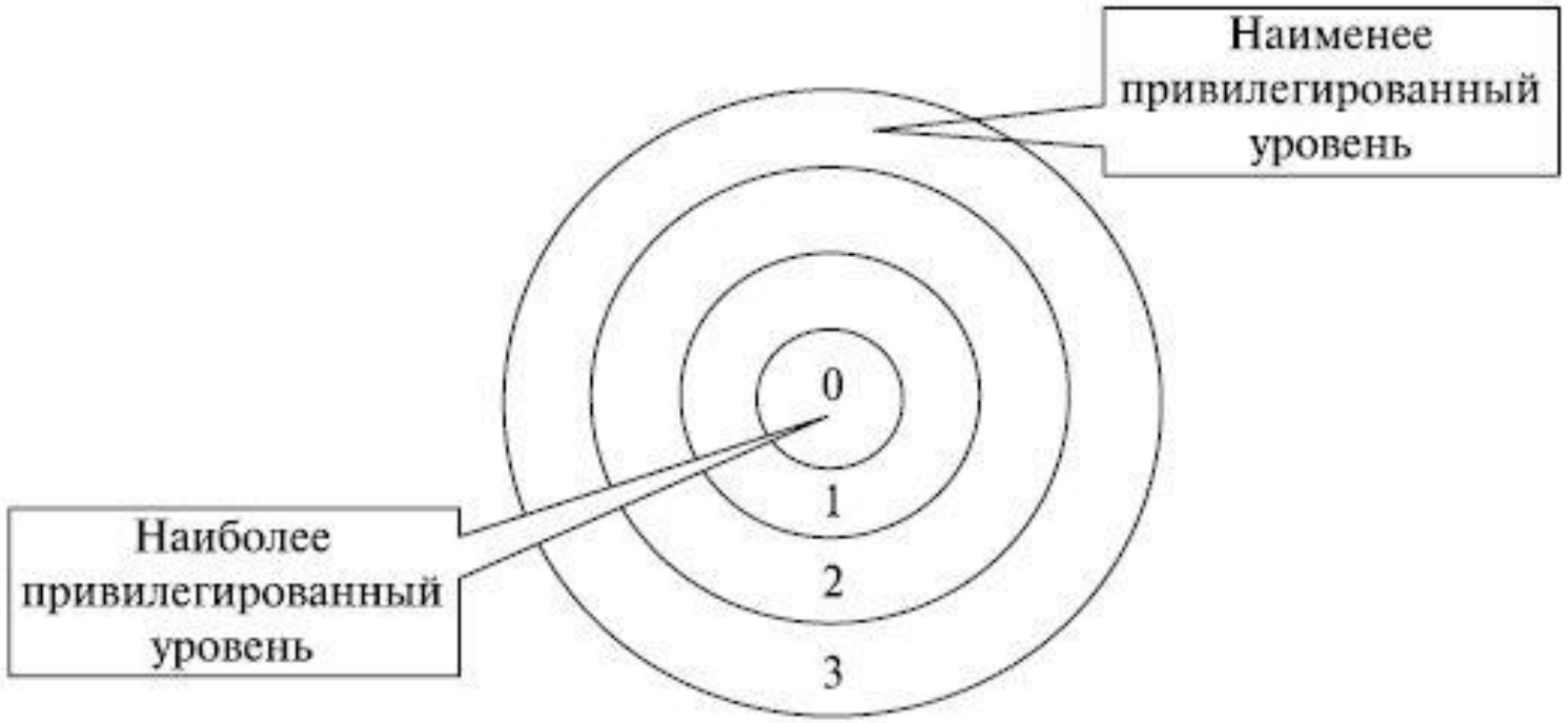
Бит LRU	Последнее обращение
B0 = 1	L0 или L1
B0 = 0	L2 или L3
B1 = 1	L0
B1 = 0	L1
B2 = 1	L2
B2 = 0	L3

Порядок замены строк в блоке *TLB*

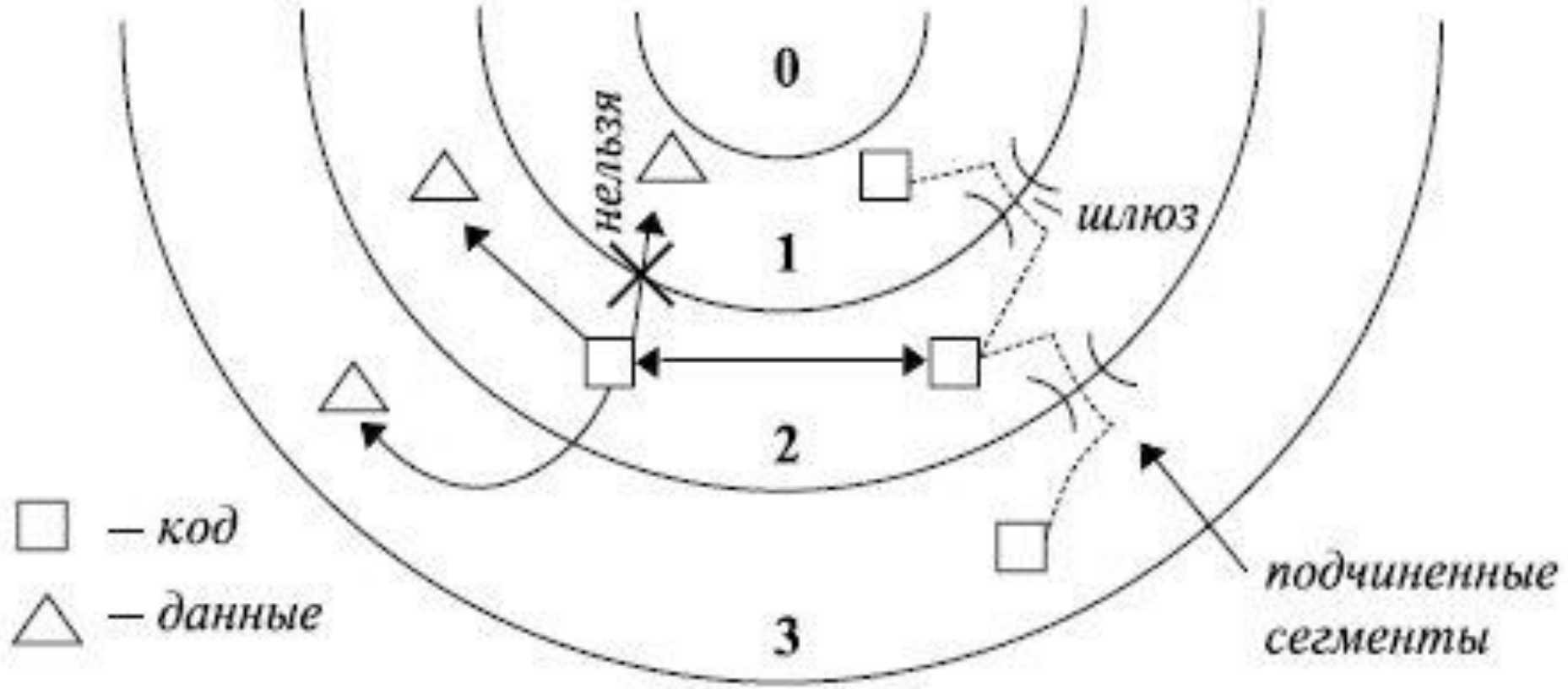
B0	B1	B2	Заменяемая строка
0	0	X	L0
0	1	X	L1
1	X	0	L2
1	X	1	L3


Механизмы защиты памяти

"Кольца защиты"

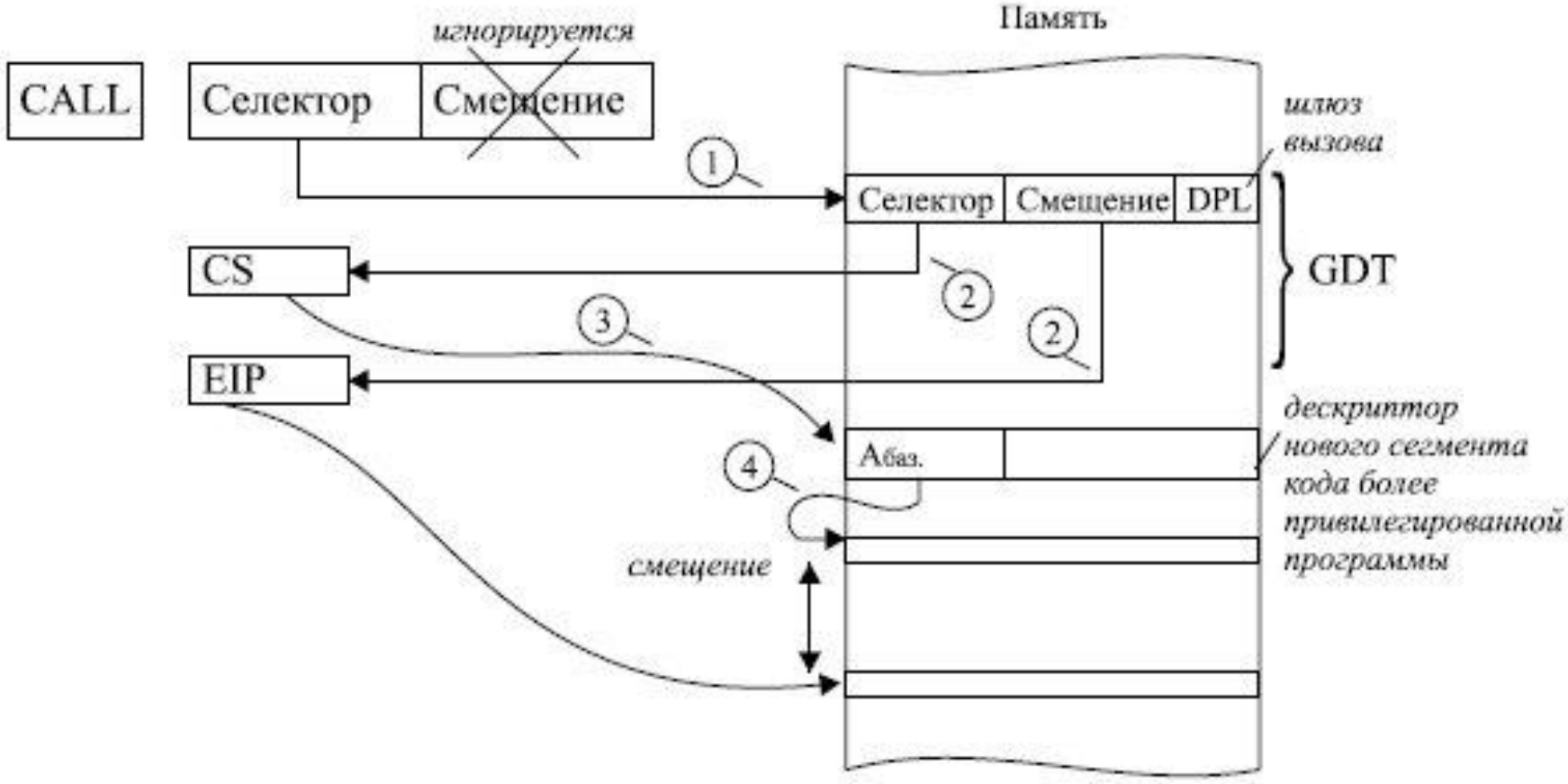


Порядок взаимодействия программ и данных на разных уровнях

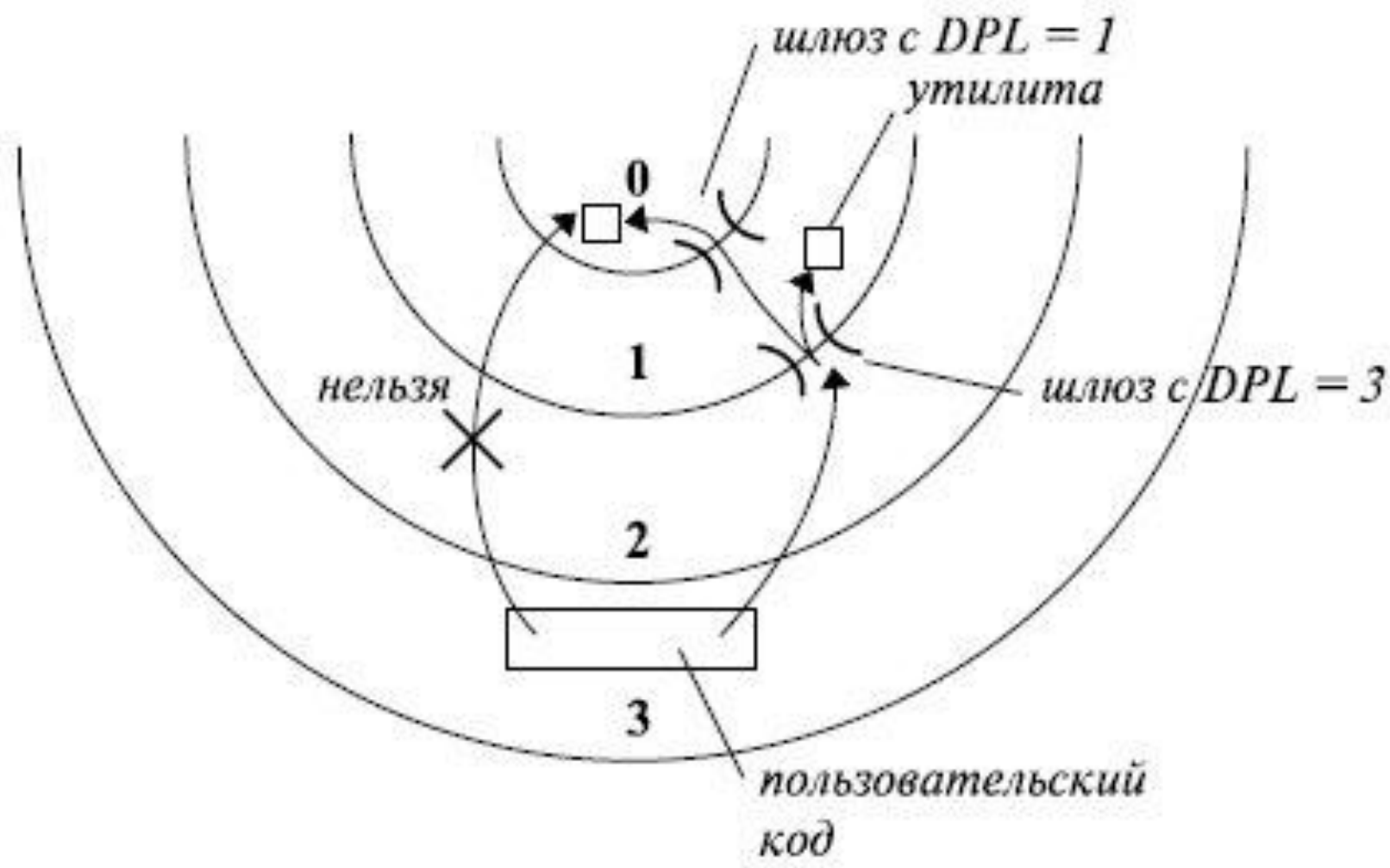


Формат шлюза вызова

Использование шлюза вызова для обращения к программам на более высоком уровне



Последовательное обращение к более привилегированным программам



Литература:

1. https://www.dropbox.com/s/gmebqjp8sc3jbfj/Arkhitektura_kompyutera_6-e_izdanie.djvu?dl=0
2. https://www.dropbox.com/s/7sescv0sx9nggqj/Lit.Lec.6.docx?dl=0

ТСИС

(Технические средства информационных систем)
Программное обеспечение информационных систем (1-40 01 73)

?

- Лекция 6

  Адресация. Режимы работы процессора.
  Управление памятью.

Ковалевский Вячеслав Викторович
4096tb@gmail.com

Тема письма:
БГУИР. … .

https://www.dropbox.com/s/7dd2s2qbfdmhz6x/TCIC.Lec6.pps?dl=0