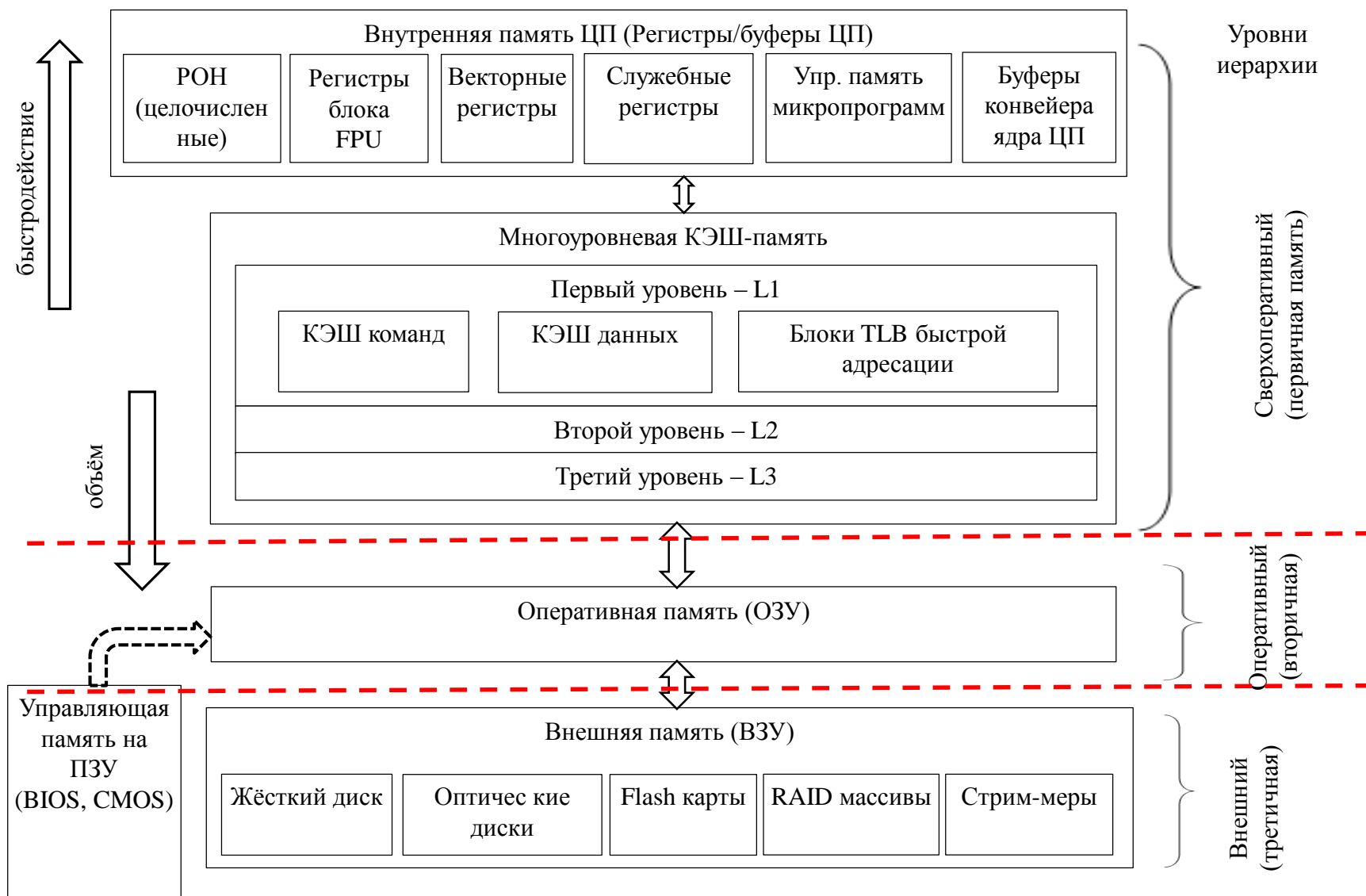


# Подсистема памяти ВС. Иерархия ЗУ



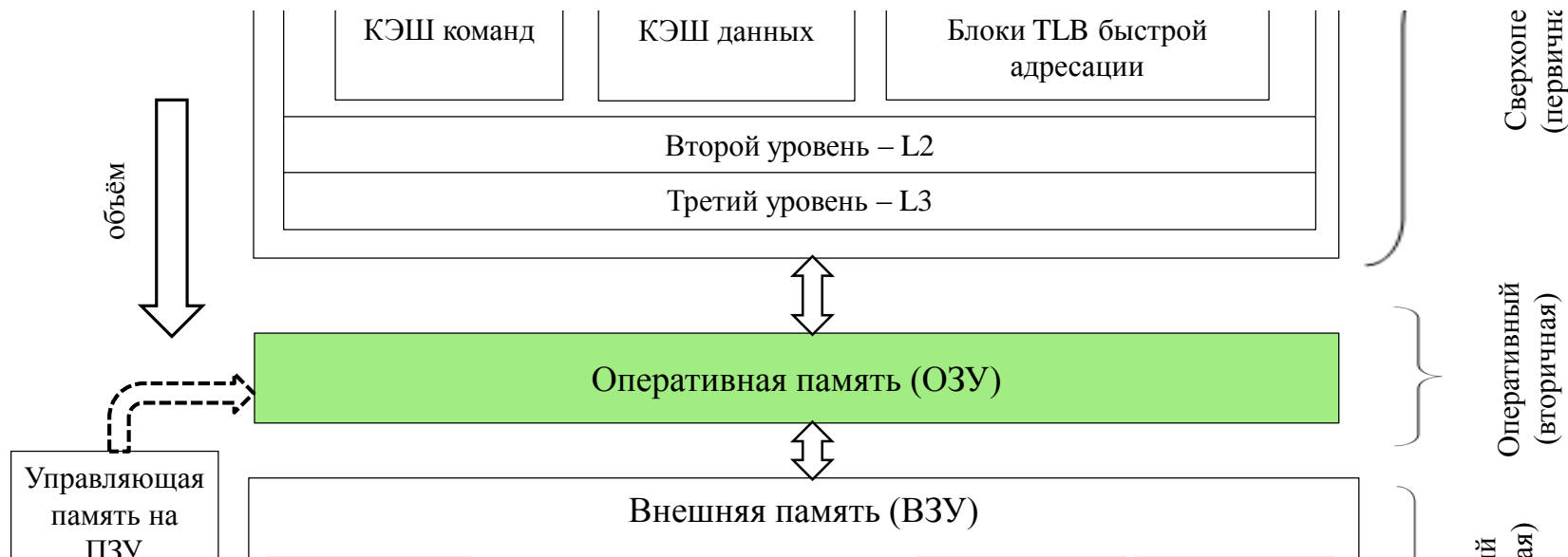
Иерархия памяти ВС

# Подсистема памяти ВС. Типы ЗУ



Иерархия памяти ВС

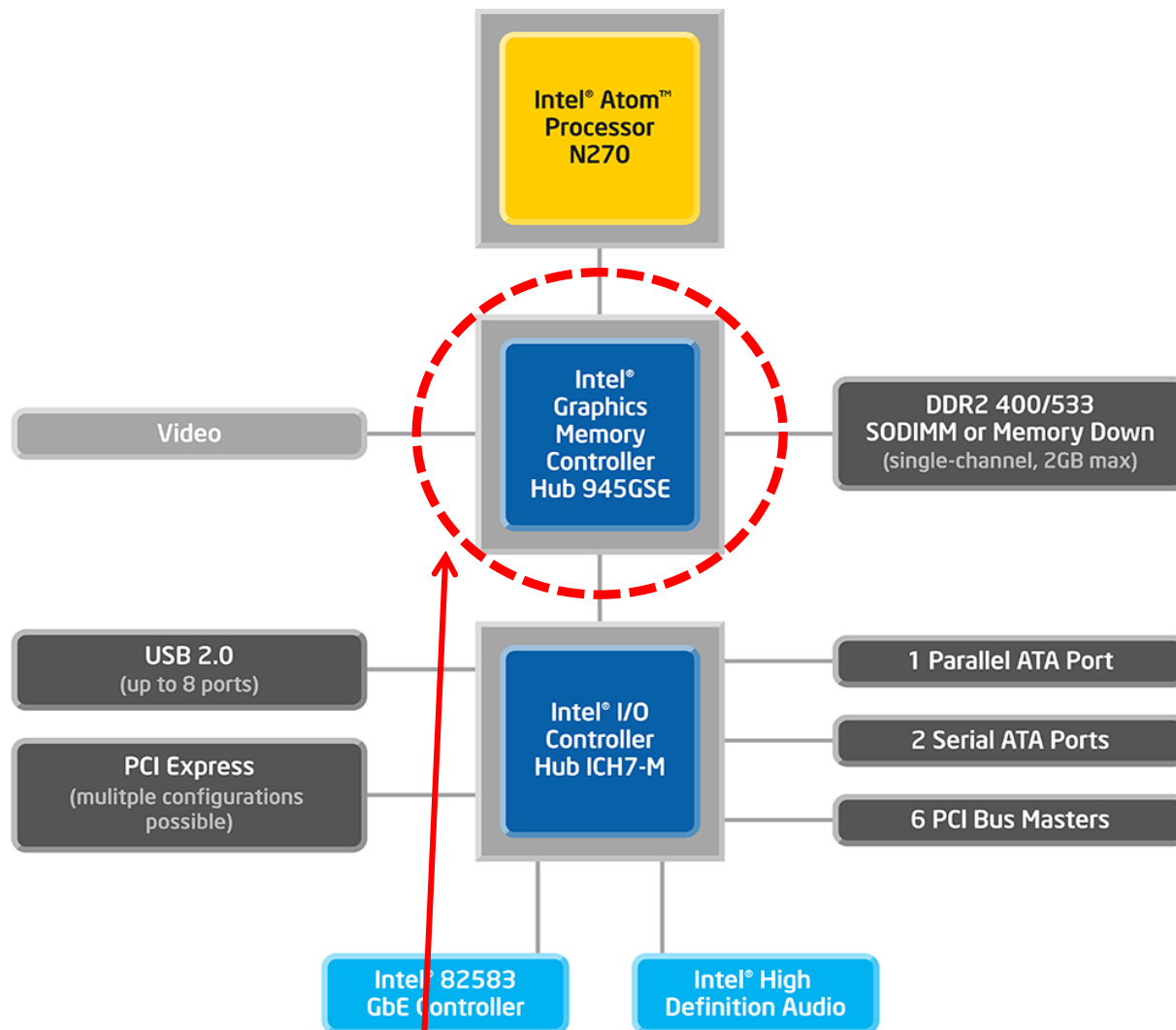
# Подсистема памяти ВС. ОП



## Основная память ВС (вторичная)

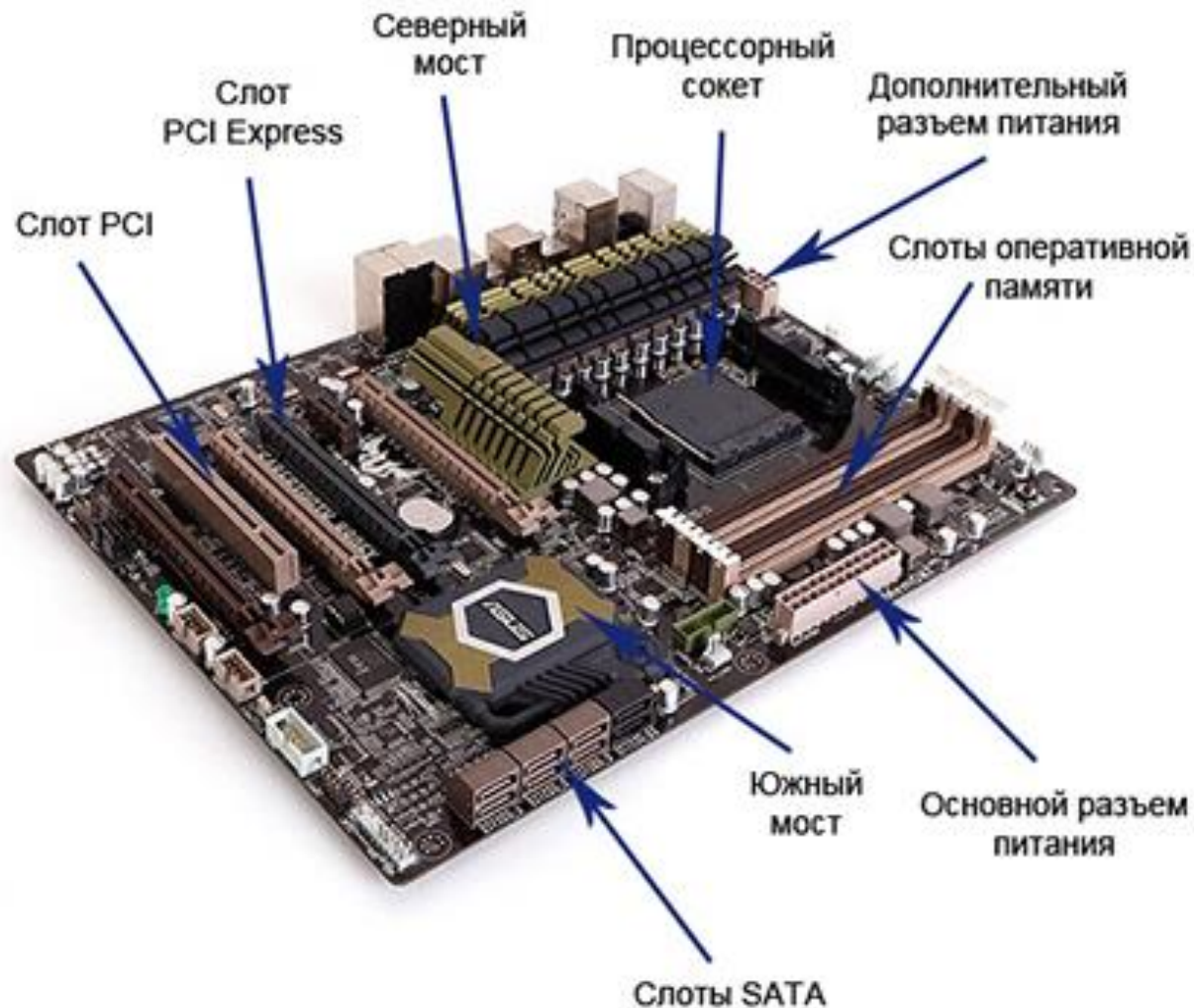
*хранения активных программ и данных, то есть тех программ и данных, с которыми в настоящее время работает ВС.*

# Схема расположения ОП в персональной ВС



В однопроцессорной архитектуре персональных ВС ОП подключалась к ЦП, КЭШ через системную шину (шину памяти) и специализированный контроллер – системный контроллер/**северный мост**.

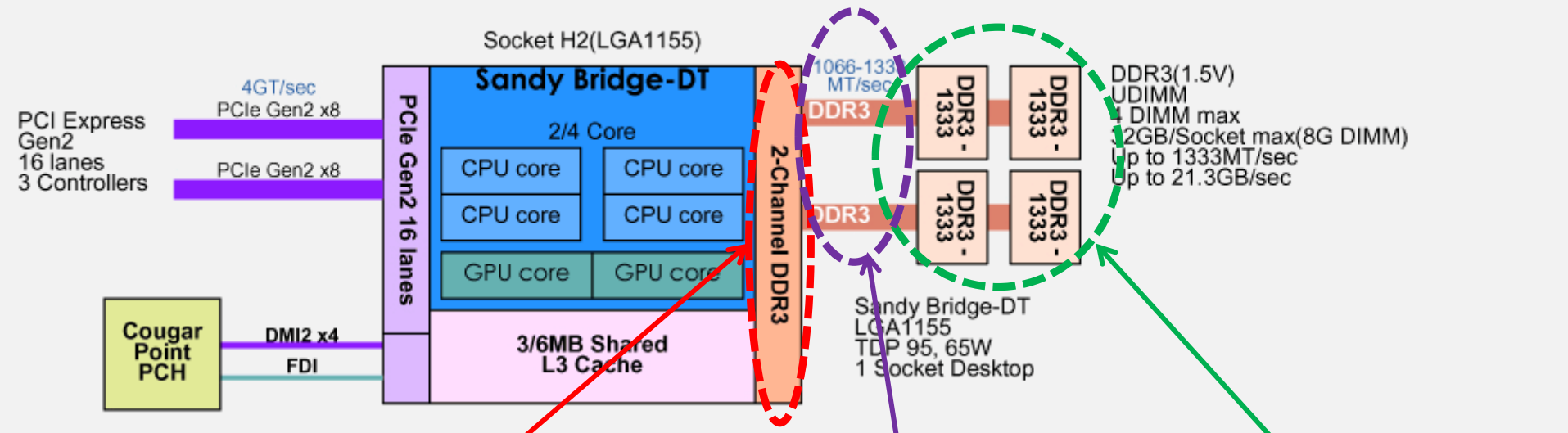
# Схема расположения ОП в персональной ВС



В однопроцессорной архитектуре персональных ВС ОП подключалась к ЦП, КЭШ через системную шину (шину памяти) и специализированный контроллер – системный контроллер/северный мост.

# Схема расположения ОП в современных ВС

## Sandy Bridge Socket H2 (LGA1155) Client PC

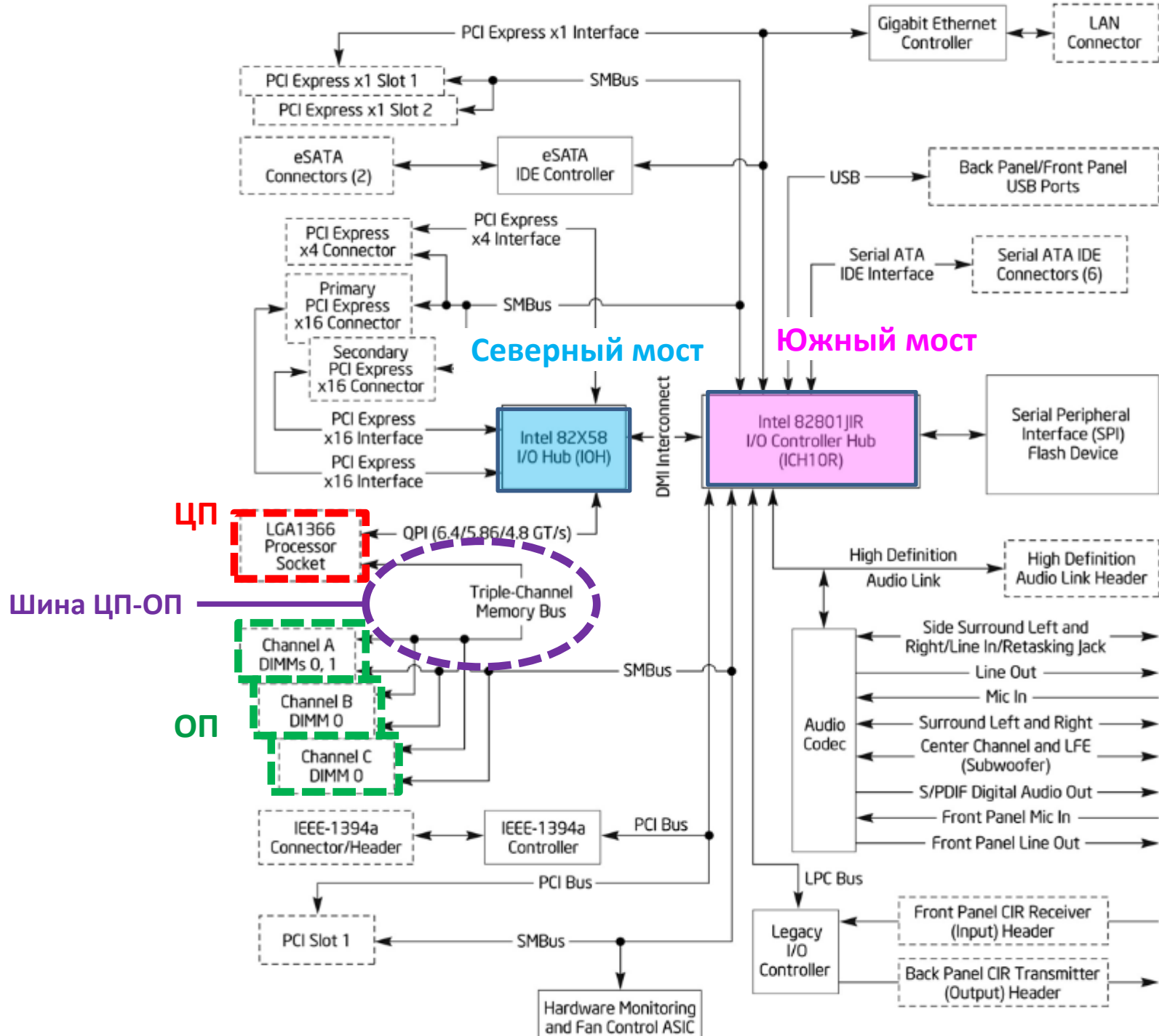


В современных ВС с микросхемами высокой степени интеграции контроллер памяти **встроен** в процессор и дает возможность напрямую подключаться к **ОП** через шину памяти, реализованную в виде **проводников** на материнской плате.

# Схема подключения ОП в современных ВС

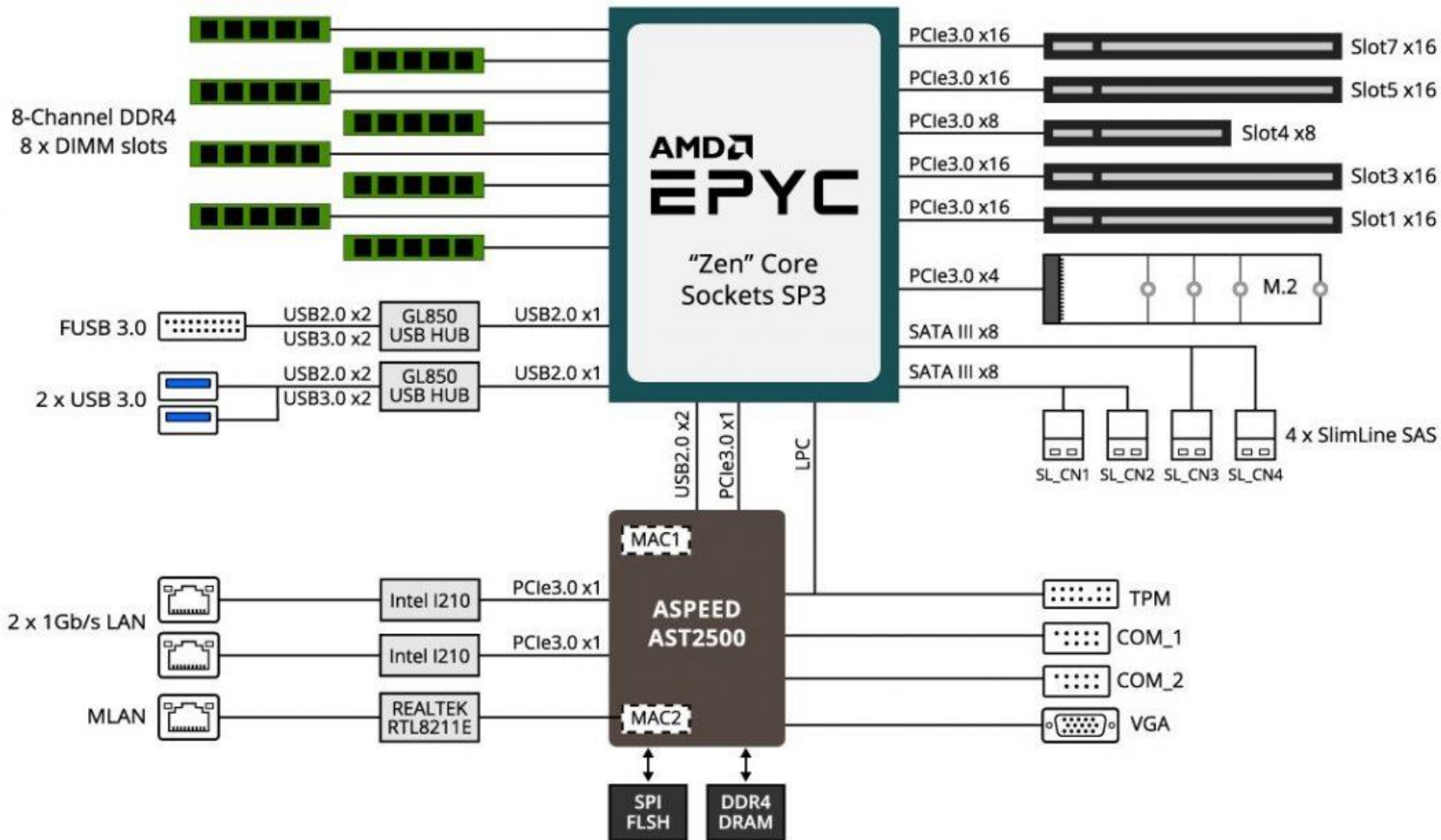
block diagram of the major functional areas of the board

([http://downloadmirror.intel.com/18128/eng/DX58SO\\_TechProdSpec.pdf](http://downloadmirror.intel.com/18128/eng/DX58SO_TechProdSpec.pdf))





# Схема расположения ОП в современных ВС



4 двухканальных слота (8 модулей DDR4) и 16 для DDR5



# Характеристики схем ОП

- **тип и принцип действия**
- **производительность**
  - частота,
  - тайминги,
  - пропускная способность
- **ёмкость памяти,**
- **напряжение питания**
- **эксплуатационные характеристики**
  - стоимость (общая и относительная)
  - производитель/качество
  - срок эксплуатации

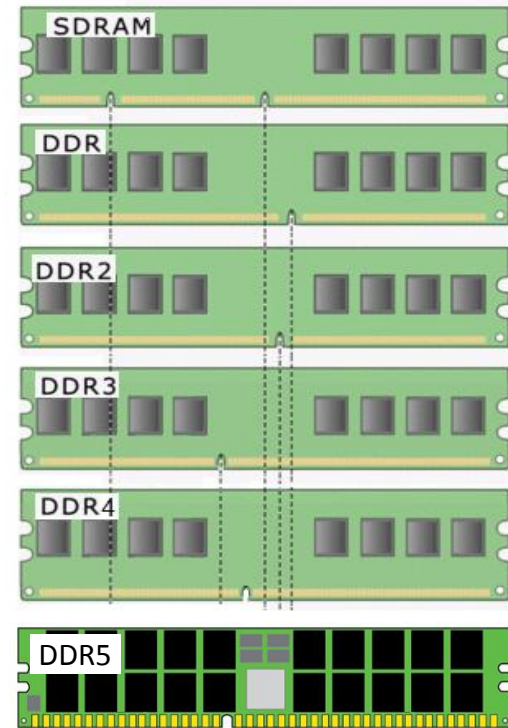
# Характеристики схем ОП

## Тип/конфигурация

*DDR2 (устарела), DDR3, DDR4,  
DDR5 (в разработке)*

## Принцип действия

- **динамическая** (на конденсаторах)
- **синхронная** (все временные интервалы отправки команд и данных из/в память привязаны к частоте синхронизации)
- **энергозависимая** (данные теряются при отключении питания)
- **память произвольного доступа** (в любой момент доступна любая ячейка)
- **прямо-адресуемая**
- **физическая или виртуальная** (при малой емкости)
- **вторичная память** (доступна процессору путём прямой адресации через шину адреса)



# Характеристики схем ОП

## производительность

- эффективная частота

*DDR2 400÷1066 МГц; DDR3 800÷2400 МГц; DDR4 2133÷4266 МГц*

*DDR5 ???*

- тайминги

- пропускная способность

# Характеристики схем ОП

## Производительность/тайминги

### DRAM Comparison: Overview

Item	DDR	DDR2	DDR3	DDR4
Data Rate (Mbps)	200, 266, 333, 400	400, 533, 667, 800	800, 1067, 1333, 1600, 1866, 2133	1600, 1866, 2133, 2400, 2666, 2933, 3200
CL	2 / 2.5 / 3	2 / 3 / 4 / 5 / 6	5~14	9~24
BL	2 / 4 / 8	4 / 8	8, BC4 (fixed, on the fly)	8, BC4 (fixed, on the fly)
Termination (DQ)	VTT	VTT	VTT	VDDQ
VDD/VDDQ	2.5V	1.8V	1.5V/1.35V/1.25V	1.2V
VREF.DQ	0.5 VDDQ	0.5 VDDQ	0.5 VDDQ	internally generated (training is a must)
Row Buffer	n/a	n/a	n/a	4 (x4/8), 2 (x16), 1 (x32)
Bank	4	4, 8 (1Gb ~)	8	16 (x4/8), 8 (x16/32)
Page Size	-	1KB (x4/8), 2KB (x16)	1KB (4/x8), 2KB (x16), 4KB (x32)	1/2KB (x4), 1KB (x8), 2KB (x16), 4KB (x32)
PKG Type	66Pin TSOP-II	FBGA (60/84 Ball)	FBGA (78/96)	FBGA (78/96)
Prefetch	2 Bit	4 Bit	8 Bit	8 Bit

CL (CAS Latency)

<http://testlabs.kz/ram/2046-ddr4-review.html>

# Характеристики схем ОП

## Производительность/тайминги

### •DDR2



Параметр/Режим	DDR2-667		DDR2-800	
Тайминги	5-5-5-12	4-4-4-12	5-5-5-12	4-4-4-12
Минимальная латентность псевдослучайного доступа, нс	30.7	28.6	27.4	24.7
Максимальная латентность псевдослучайного доступа, нс	34.0	31.9	30.4	27.6
Минимальная латентность случайного доступа **, нс	91.6	88.0	76.2	74.2
Максимальная латентность случайного доступа **, нс	97.3	91.7	78.8	76.4

<http://www.ixbt.com/mainboard/ddr2-800-am2-fx62.shtml>

$$\text{Частота} = 667 \text{ МГц} = 6,67 \times 10^8 \left[ \frac{\text{такт}}{\text{сек}} \right]$$

$$\text{Длительность такта} = \frac{1}{\text{частота}} = \frac{1}{6,67 \times 10^8 \left[ \frac{\text{такт}}{\text{сек}} \right]} = 1,5 \times 10^{-9} \left[ \frac{\text{сек}}{\text{такт}} \right]$$

$$\text{Латентность} = 5 [\text{тактов}] \times 1,5 \times 10^{-9} \left[ \frac{\text{сек}}{\text{такт}} \right] = 7,5 \times 10^{-9} [\text{сек}] = 7,5 \text{ нс}$$

# Характеристики схем ОП

## Производительность/тайминги

### •DDR3

Название	Номинальная тактовая частота, МГц	Номинальные тайминги
Goodram Pro DDR3-2133 2 x 2 Гбайт (GP2133D364L10/4GDC)	1066	10-10-10-30
Apacer Armor Series DDR3-2133 2 x 4 Гбайт (78.BAGGL.AFK0C)	1066	11-11-11-30
Corsair Vengeance 8GB DDR3-2133 2 x 4 Гбайт (CMZ8GX3M2X2133C9R)	1066	9-11-10-30
ADATA XPG Xtreme Series DDR3 2133 2 x 4 Гбайт (AX3U2133XC4G10-2X)	1066	10-11-11-30

$$\text{Частота} = 1,066 \times 10^9 \left[ \frac{\text{такт}}{\text{сек}} \right] \quad \text{Длительность такта} = \frac{1}{10^9 \left[ \frac{\text{такт}}{\text{сек}} \right]} = 10^{-10} \left[ \frac{\text{сек}}{\text{такт}} \right]$$

$$\text{Латентность} = 10 [\text{тактов}] \times 10^{-10} \left[ \frac{\text{сек}}{\text{такт}} \right] = 10^{-9} [\text{сек}]$$

# Характеристики схем ОП

## Производительность/тайминги

### •DDR4

Тайминги	DDR4-2133 (стандарт Jedec)	DDR4-2666 (XMP профиль)	DDR4-3000 (XMP профиль)	DDR4-3000 (строгие тайминги)
CL	15	14	15	12
tRCD	15	14	16	13
tRP	15	14	16	15
tRAS	36	36	39	15
			1	1
			382	270
			14	9



Латентность = 60 нс =  $60 \times 10^{-9}$  [сек]



# Характеристики схем ОП

## Производительность/пропускная способность



Пропускная способность = ширина выборки (разрядность ЦП) × частоту

• **DDR2** – 64бит (8Байт) × 667 МГц = 5,336 ГБ/с

8Б × 800 МГц = 6,4 ГБ/с

Параметр	Двухканальный режим	2 × DDR2-667	2 × DDR2-800
Теоретическая ПСП, МБ/с		10667	12800
Средняя ПСП на чтение, МБ/с		3368	3590
Средняя ПСП на запись, МБ/с		2759	2909
Макс. ПСП на чтение, МБ/с		6590 (61.8 %)	6819 (53.3 %)

<http://www.ixbt.com/mainboard/ddr2-800-am2-fx62.shtml>

# Характеристики схем ОП

## Производительность/пропускная способность

- $DDR2 \sim 3,2 \div 8,5 \text{ ГБ/с}$
- $DDR3 \sim 6,4 \div 19,2 \text{ ГБ/с}$
- $DDR4 \ 2133 \div 4266 \text{ МГц} \sim 17 \div 34 \text{ ГБ/с}$
- $DDR5 \text{ ок } 32 \div 64 \text{ ГБ/с}$

4 ч  
ПО

AIDA64 Cache & Memory Benchmark				
	Read	Write	Copy	Latency
Memory	58101 MB/s	46857 MB/s	57860 MB/s	65.6 ns
CPU Type	OctalCore Intel Core i7-5960X Extreme Edition (Haswell-E, LGA2011-v3)			
CPU Stepping	R2			
CPU Clock	3500.0 MHz (original: 3000 MHz, overclock: 17%)			
Memory Bus	1200.0 MHz	DRAM:FSB Ratio		36:3
Memory Type	Quad Channel DDR4-2400 SDRAM (14-14-14-36 CR2)			
Chipset	Intel Wellsburg X99, Intel Haswell-E			
Motherboard	Gigabyte GA-X99-Gaming G1 WiFi			
AIDA64 v4.70.3200 / BenchDLL 4.1.622-x64 (c) 1995-2014 FinalWire Ltd.				

4 четырех-канальных модуля  
ПСП=1200×16×8=153600МБ/с

# Характеристики схем ОП

## Ёмкость памяти

- *DDR2 ~ 1÷8 ГБ*
- *DDR3 ~ 1÷32 ГБ*
- *DDR4 ~ 4÷64 ГБ*
- *DDR5 экспериментальный образец 64 ГБ*

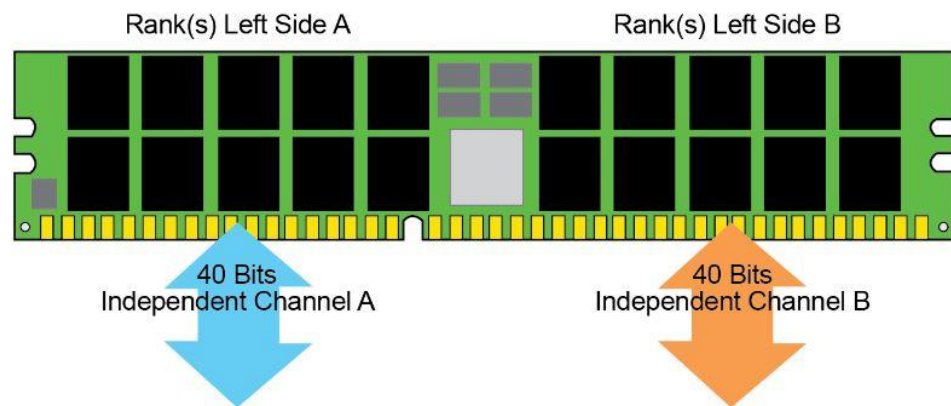
# Характеристики схем ОП

## Напряжение питания

DDR2	DDR3	DDR3L	DDR4	DDR5
1.8V	1.5V	1.35V	1.2V	1.1V

Технические характеристики оперативной памяти по стандартам JEDEC — *Joint Electron Device Engineering Council* (Объединенный инженерный совет по электронным устройствам)

# DDR5



- техпроцесс 1z-nm
- ECC – с коррекцией ошибок
- RDIMM – регистровая память с буферизацией сигналов управления и адресов
- 2 независимых 40-битных (32 бита + ECC) канала данных в каждом модуле
- увеличение группы банков для наращивания производительности

Пока не поддерживается ни одной из платформ Intel или AMD

- <https://club.dns-shop.ru/digest/24761-sk-hynix-pokazala-operativnuu-pamyat-ddr5-na-ces-2020/>
- <https://news.samsung.com/global/samsung-develops-industrys-first-3rd-generation-10nm-class-dram-for-premium-memory-applications>

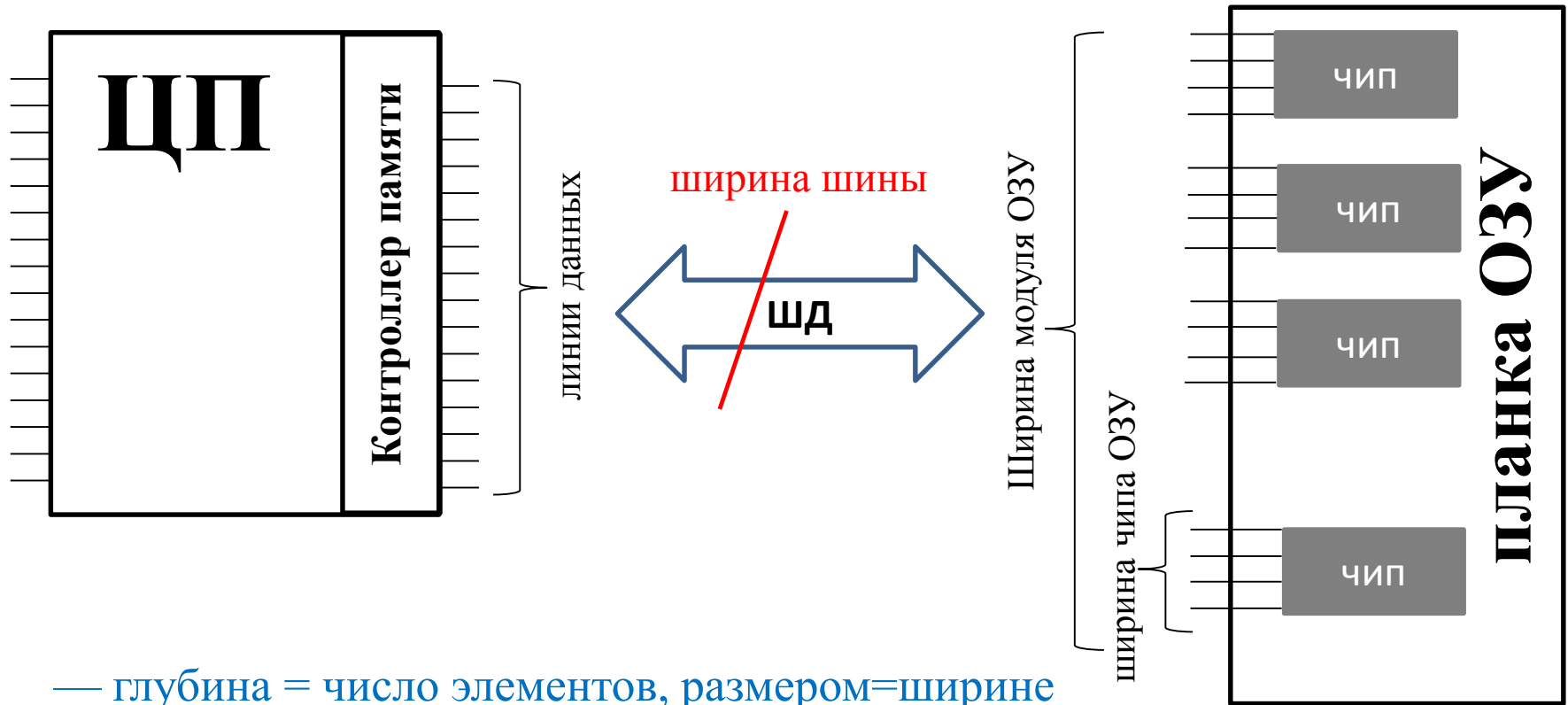
# Физическая организация модулей ОЗУ

— емкость модуля (планки) ОЗУ = ширина ОЗУ × глубина ОЗУ

*максимальный объем информации, который данный модуль способен в себя вместить.*

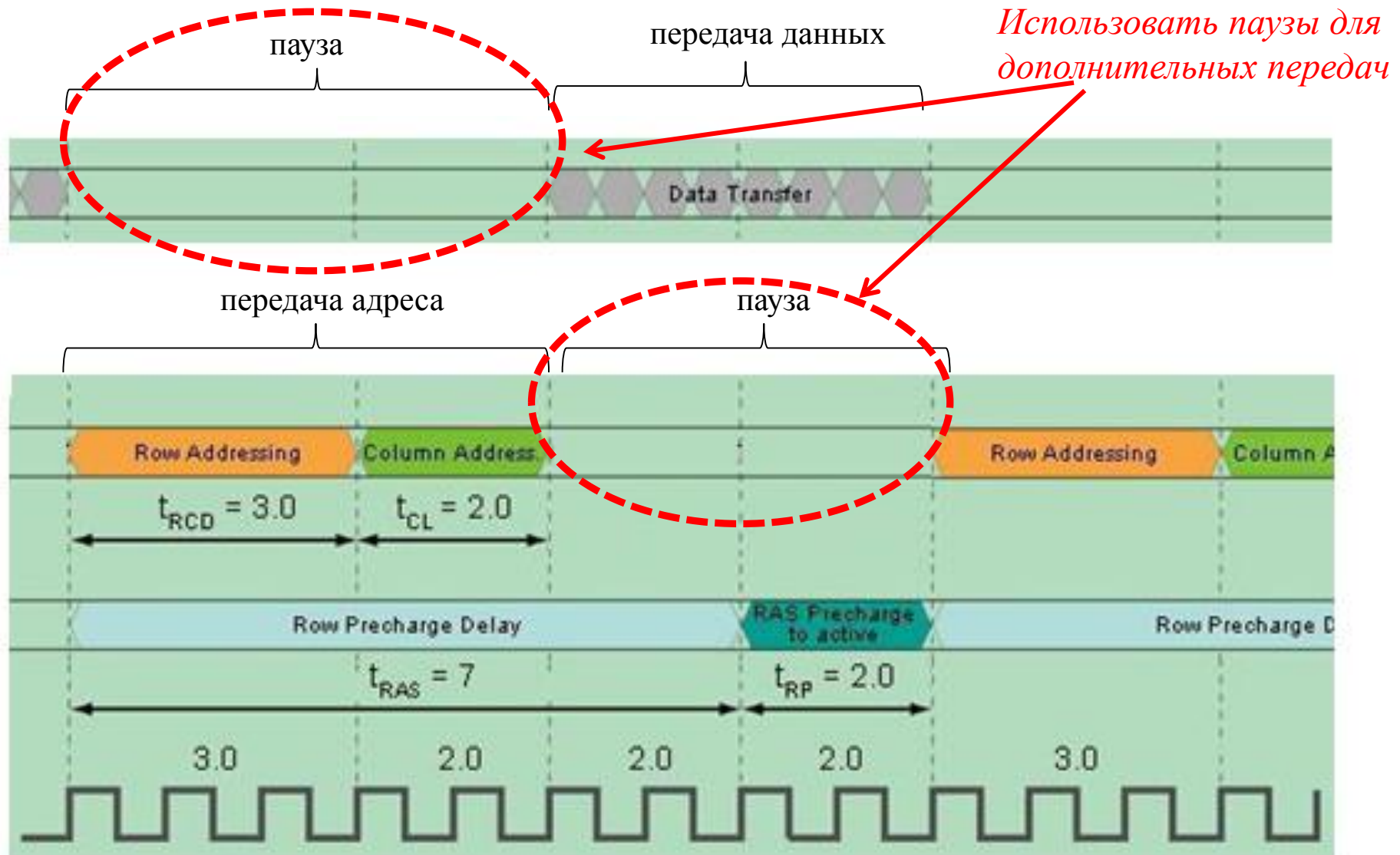
— ширина модуля (планки ОЗУ) = разрядность интерфейса шины данных = разрядность шины данных контроллера памяти = 64 бита.

ширина ОЗУ = ширина чипа × количество чипов



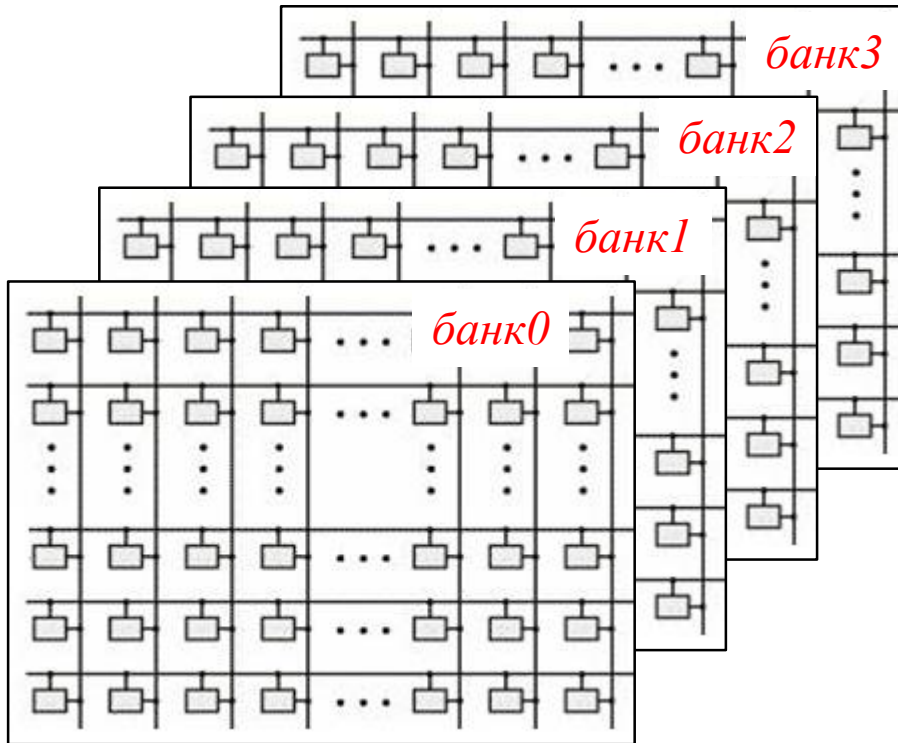
— глубина = число элементов, размером=ширине

# Логическая организация доступа к данным ОЗУ



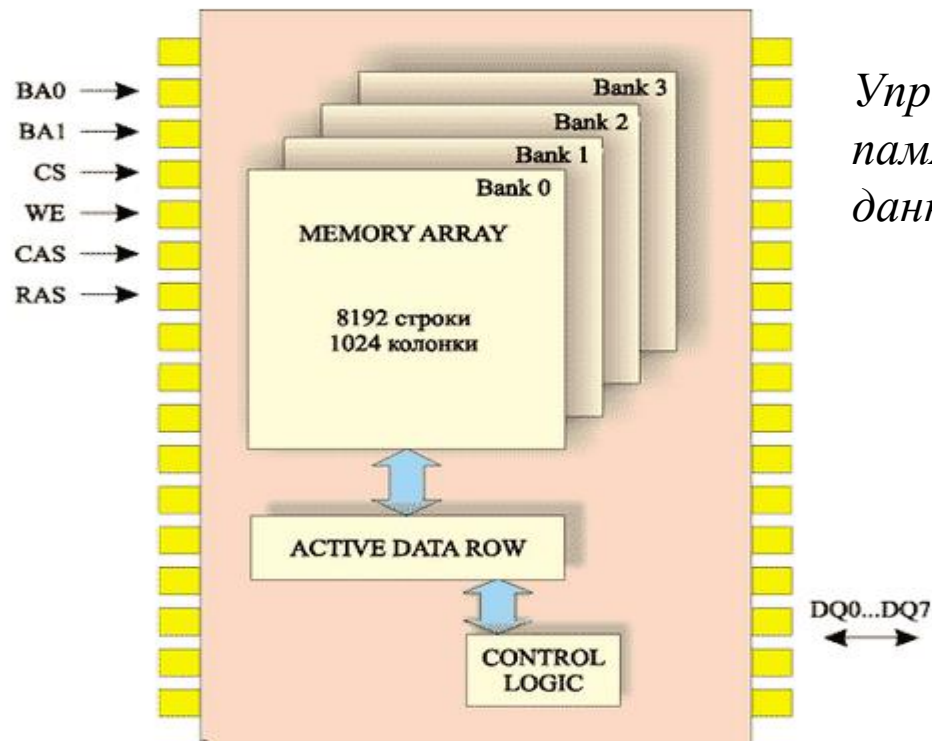


# Логическая организация доступа к данным ОЗУ



Вместо одного набора ячеек (*банка памяти*) использовать несколько наборов (*несколько банков*) и чередовать обращение к ним, заполняя паузы (*Bank Interleave*)

# Логическая организация ОЗУ = расслоение памяти + расслоение адресов

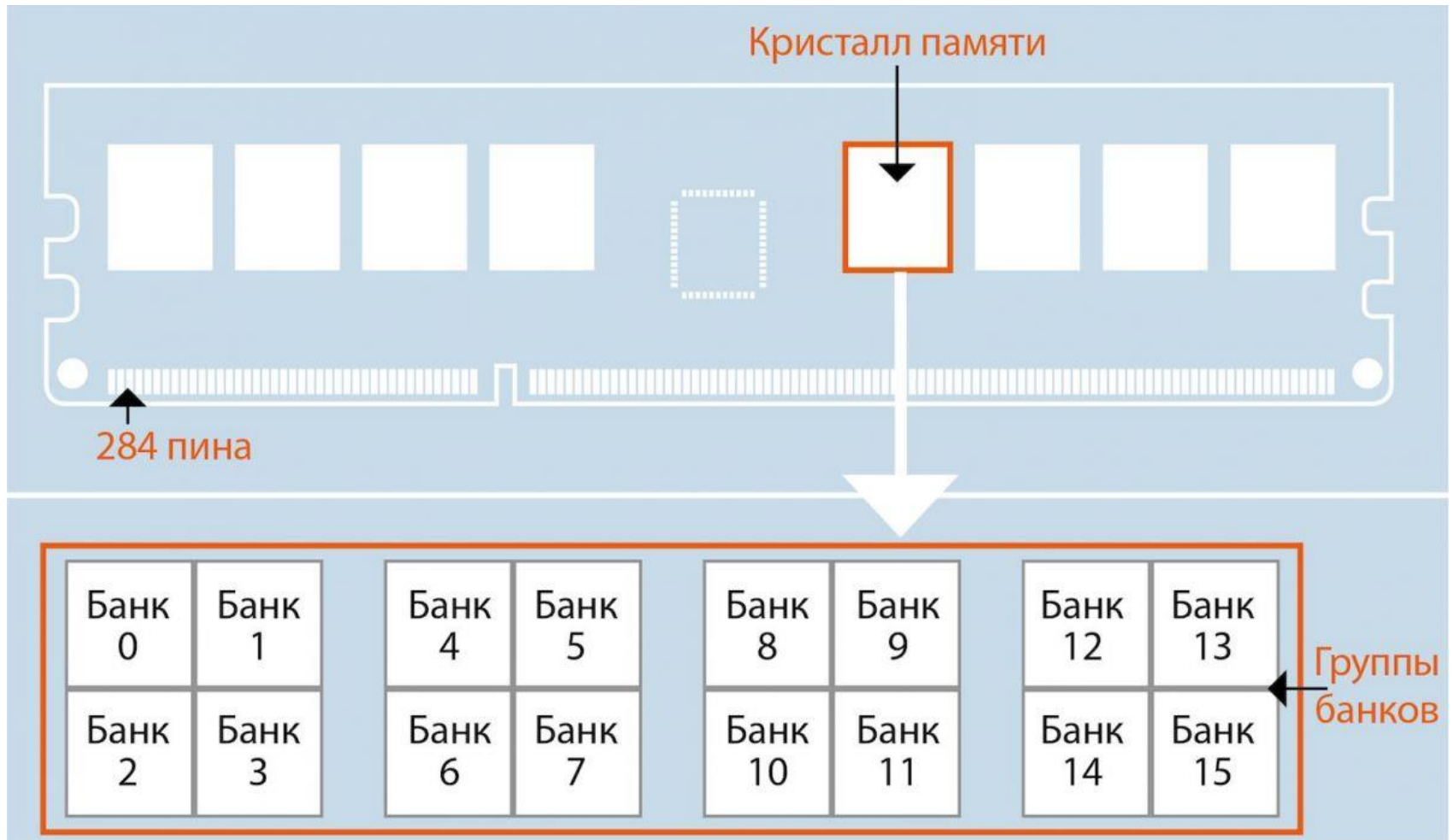


*Упрощенная схема чипа  
памяти с шириной шины  
данных 8 бит*



Упрощенная схема чипа памяти с шириной шины данных 8 бит

# Группы банков



4 группы по 4 банка

# Логическая организация ОЗУ = расслоение памяти + расслоение адресов

На **физическом уровне** каждый чип ОЗУ строиться из нескольких наборов ячеек (банков) с автономными схемами адресации, записи и чтения – *расслоение памяти*.

На **логическом уровне** управления памятью организуются последовательные обращения к различным физическим модулям – *расслоение адресов*.

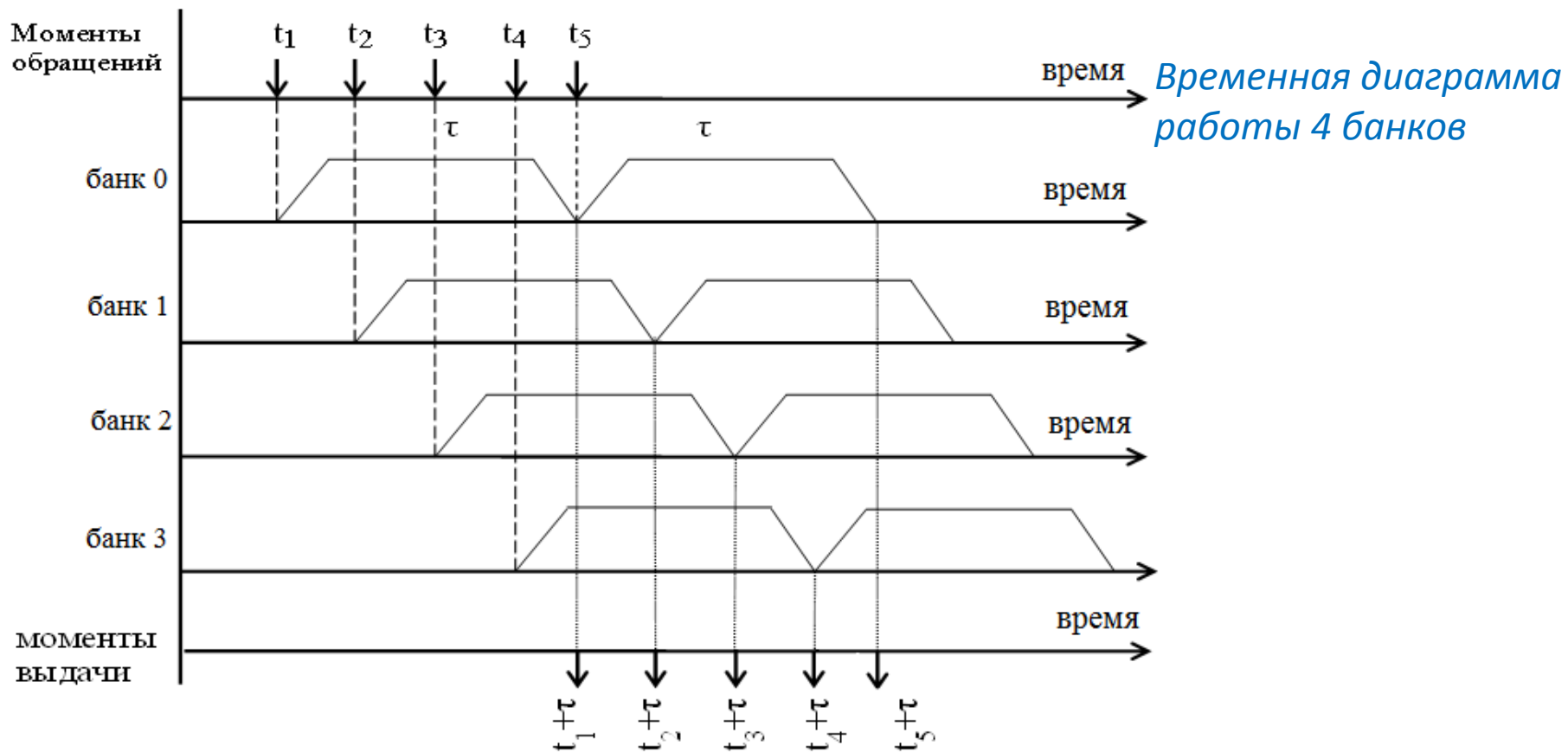
Типичный случай распределения адресов – последовательность вида  $a, a+1, a+2, a+3$

	Адрес ячейки в банке BA <sub>n</sub> ... BA <sub>4</sub>	Адрес группы BA <sub>3</sub> BA <sub>2</sub>	Адрес банка BA <sub>1</sub> BA <sub>0</sub>
$a$	0.....0000	0 0	0 0
$a+1$	0.....0000	0 0	0 1
$a+2$	0.....0000	0 0	1 0
$a+3$	0.....0000	0 0	1 1
$a+4$	0.....0001	0 1	0 0
$a+5$	0.....0001	0 1	0 1

При этом типичное распределение ячеек по модулям (банкам) обеспечивается за счет использования адресов вида:

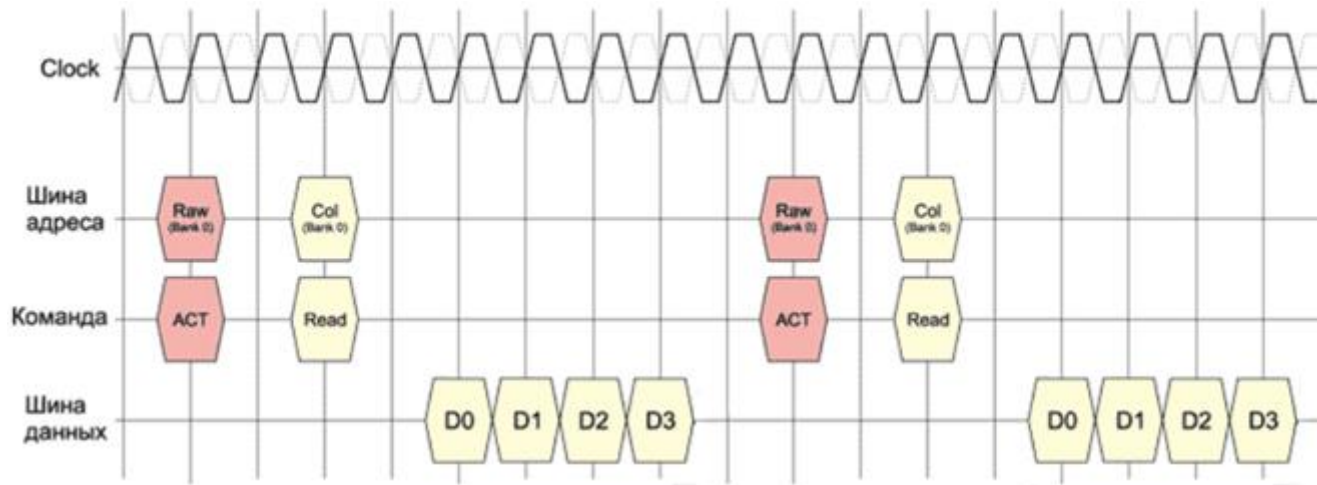
(Адрес ячейки в банке памяти) - (Адрес группы) - (Адрес банка)

# РАССЛОЕНИЕ ПАМЯТИ

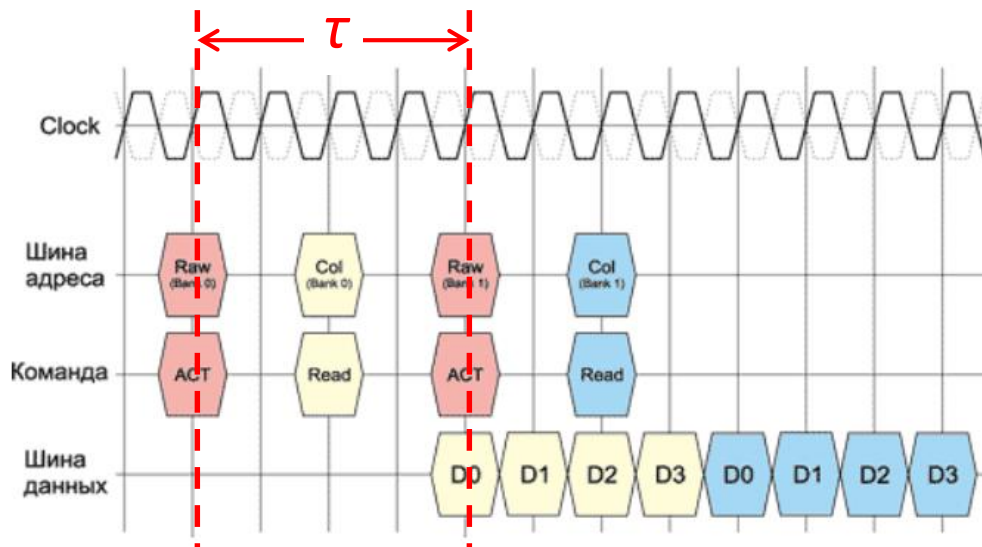


Обращение к каждому следующему банку происходит со сдвигом по фазе ( $\tau$ ) по отношению к предыдущему обращению

# Логическая организация ОЗУ = расслоение памяти + расслоение адресов

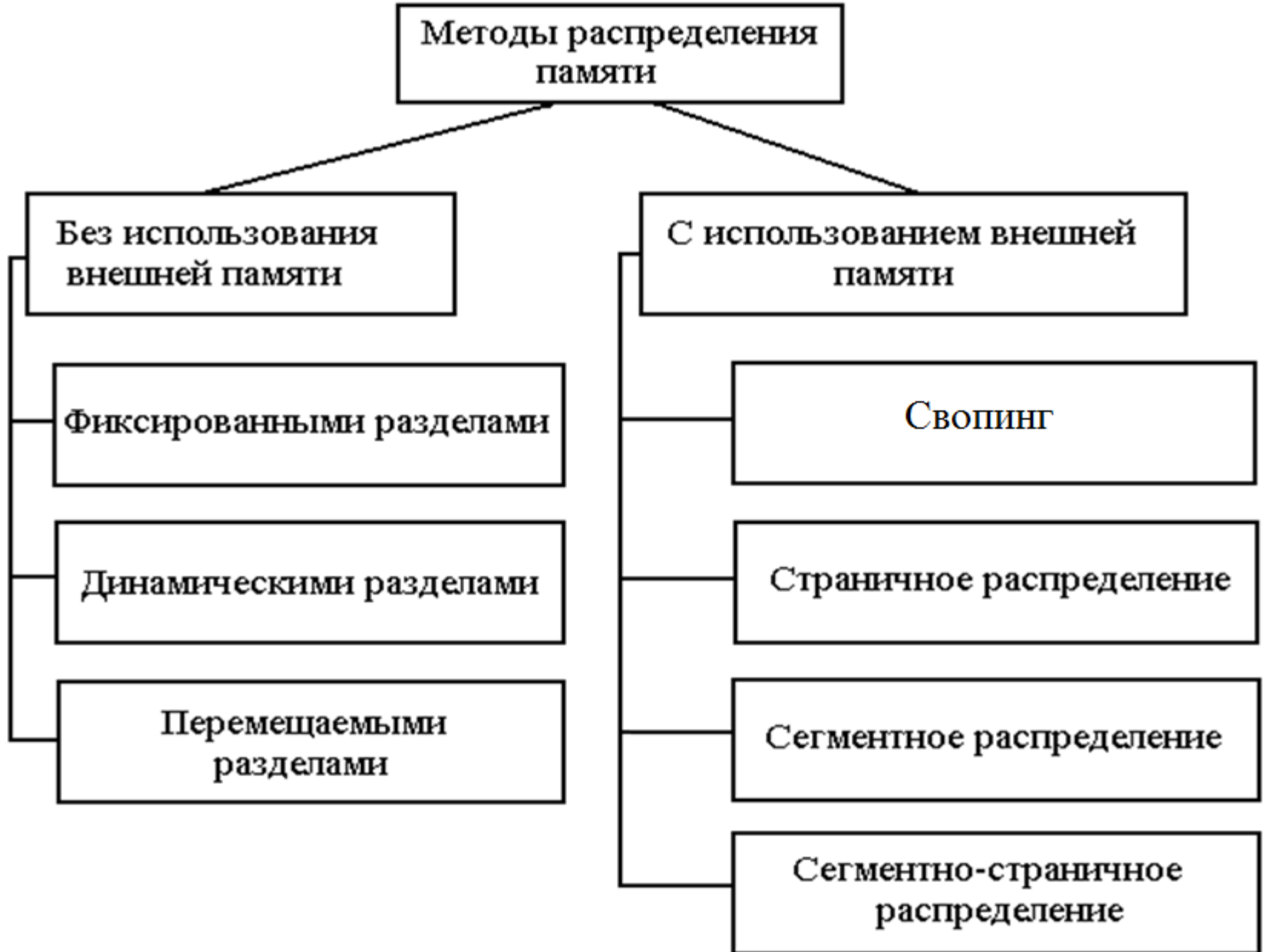


Пример обработки запроса на чтение в случае одного логического банка



Пример обработки запроса на чтение в случае двух логических банков

# МЕТОДЫ УПРАВЛЕНИЯ ОПЕРАТИВНОЙ ПАМЯТЬЮ ВС

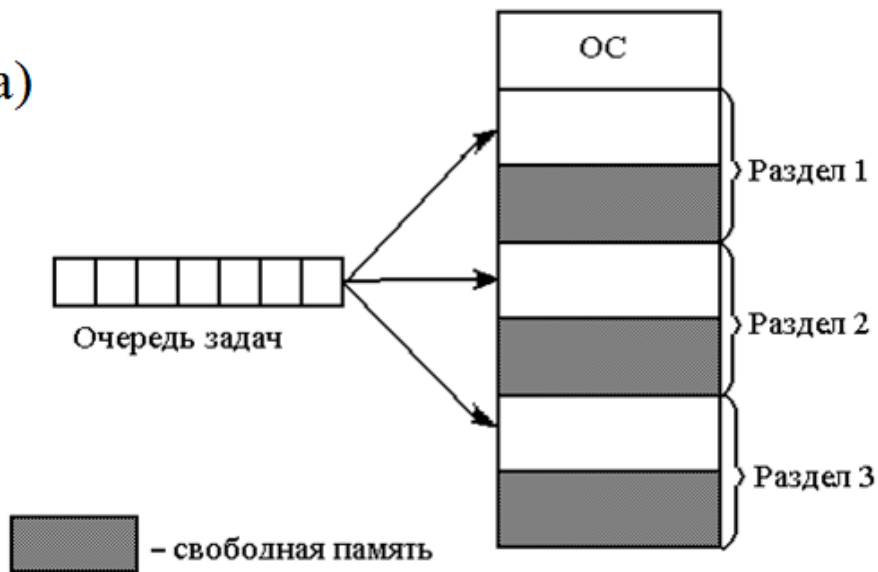




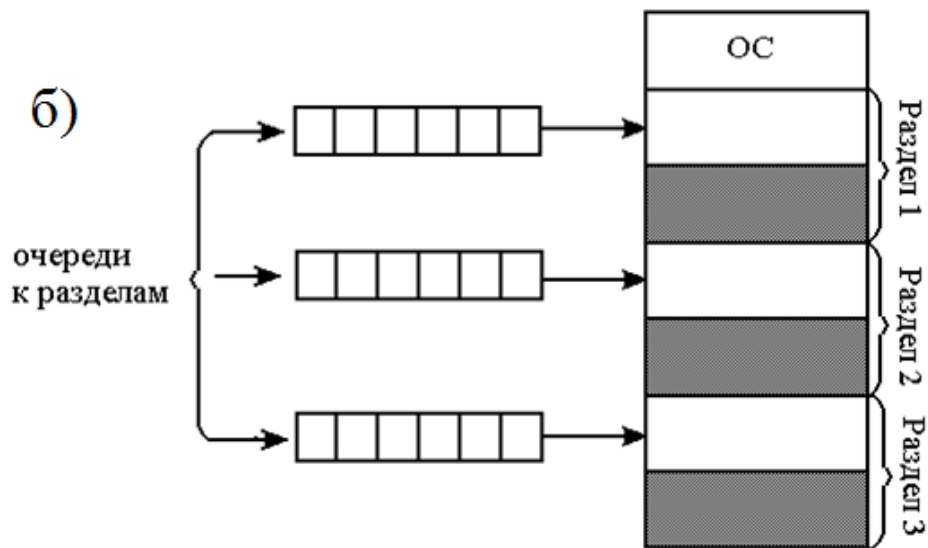
# МЕТОДЫ УПРАВЛЕНИЯ ОПЕРАТИВНОЙ ПАМЯТЬЮ ВС

## *Распределение памяти фиксированными разделами*

а)



б)



### Задачи п/с управления памятью:

- сравнения размера поступившей на выполнение программы с размерами свободных разделов памяти;
- выбора подходящего раздела;
- загрузка программы и настройка адресов.

### Достоинства:

- простота реализации

### Недостатки:

- неэкономно расходуется память (задача занимает весь раздел целиком) и всегда в конце раздела есть незанятая область. Даже если суммарный объем свободной ОП машины позволяет выполнять некоторую программу, разбиение памяти на разделы не позволяет сделать этого.
- коэффициент мультипрограммирования (одновременно исполняемых задач) ограничен числом разделов.

# МЕТОДЫ УПРАВЛЕНИЯ ОПЕРАТИВНОЙ ПАМЯТЬЮ ВС

## *Распределение памяти разделами переменной величины*

### Задачи п/с управления памятью:

- ведение таблиц свободных и занятых областей (адреса и размеры участков памяти);
- анализ запроса (при поступлении новой задачи);
- просмотр таблицы свободных областей с целью выбора раздела для размещения вновь поступившей задачи по правилам:
  - «первый попавшийся раздел достаточного размера»,
  - «раздел, имеющий min достаточный размер»,
  - «раздел, имеющий max достаточный размер»;
- загрузка задачи в выделенный ей раздел.

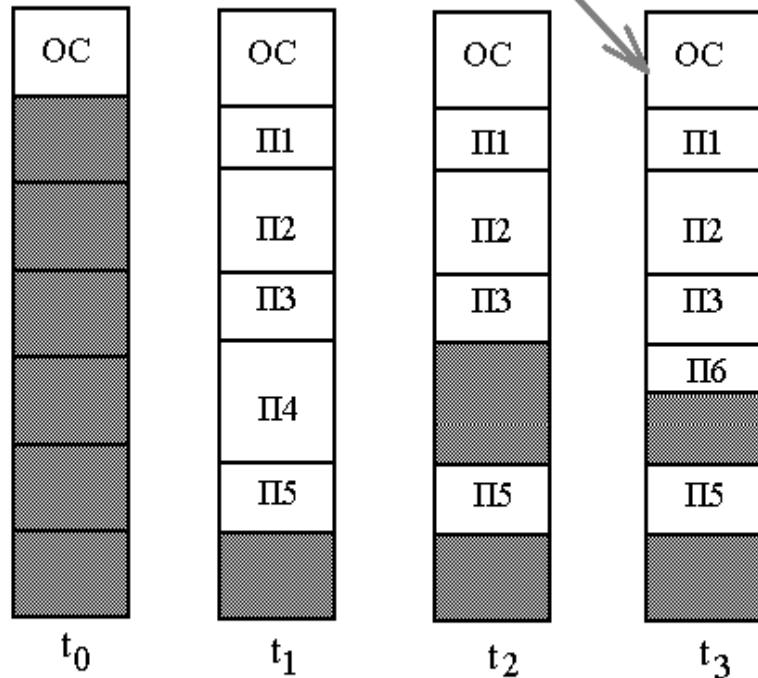
### Достоинства:

- относительная простота
- позволяет использовать два **смежных** свободных раздела как один большой или разбивать один большой на несколько малых
- выше коэффициент мультипрограммирования


### Недостатки:

- фрагментация памяти – это наличие многих несмежных областей памяти, настолько маленьких по размеру, что ни в одну из них нельзя поместить ни одну из пришедших на выполнение программ, хотя суммарный объем таких фрагментов может составить значительную величину.

новая задача П6

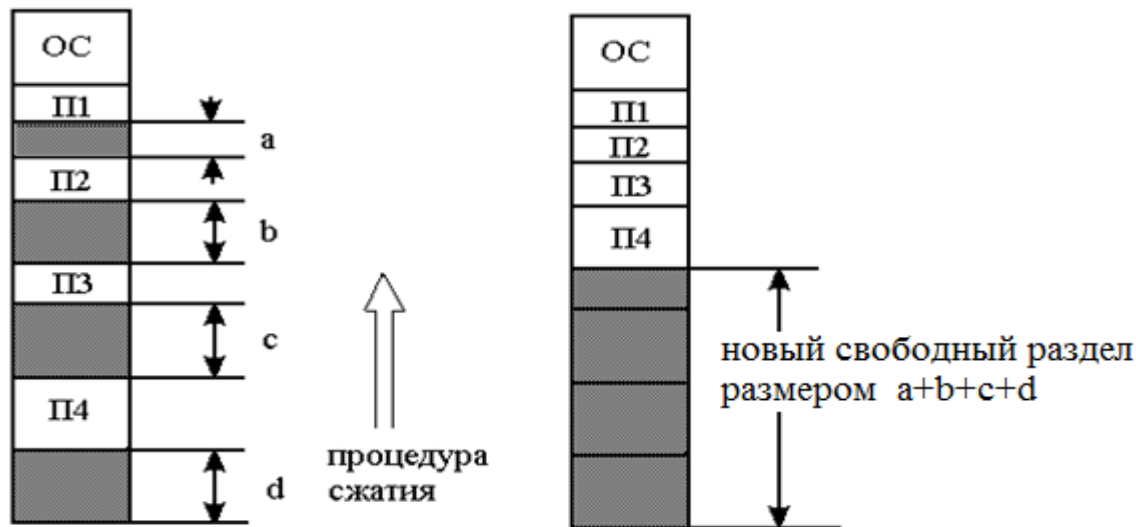


 – занятая область

 – свободная область

# МЕТОДЫ УПРАВЛЕНИЯ ОПЕРАТИВНОЙ ПАМЯТЬЮ ВС

## *Распределение памяти перемещаемыми разделами*



### Задачи п/с управления памятью:

те же, что и при распределении ОП переменными разделами +

- выполнение процедуры сжатия/дефрагментации – перемещение всех занятых участков в сторону старших либо в сторону младших адресов так, чтобы все свободные участки памяти составляли единую область.
- коррекция таблицы свободных и занятых областей.

### Достоинства:

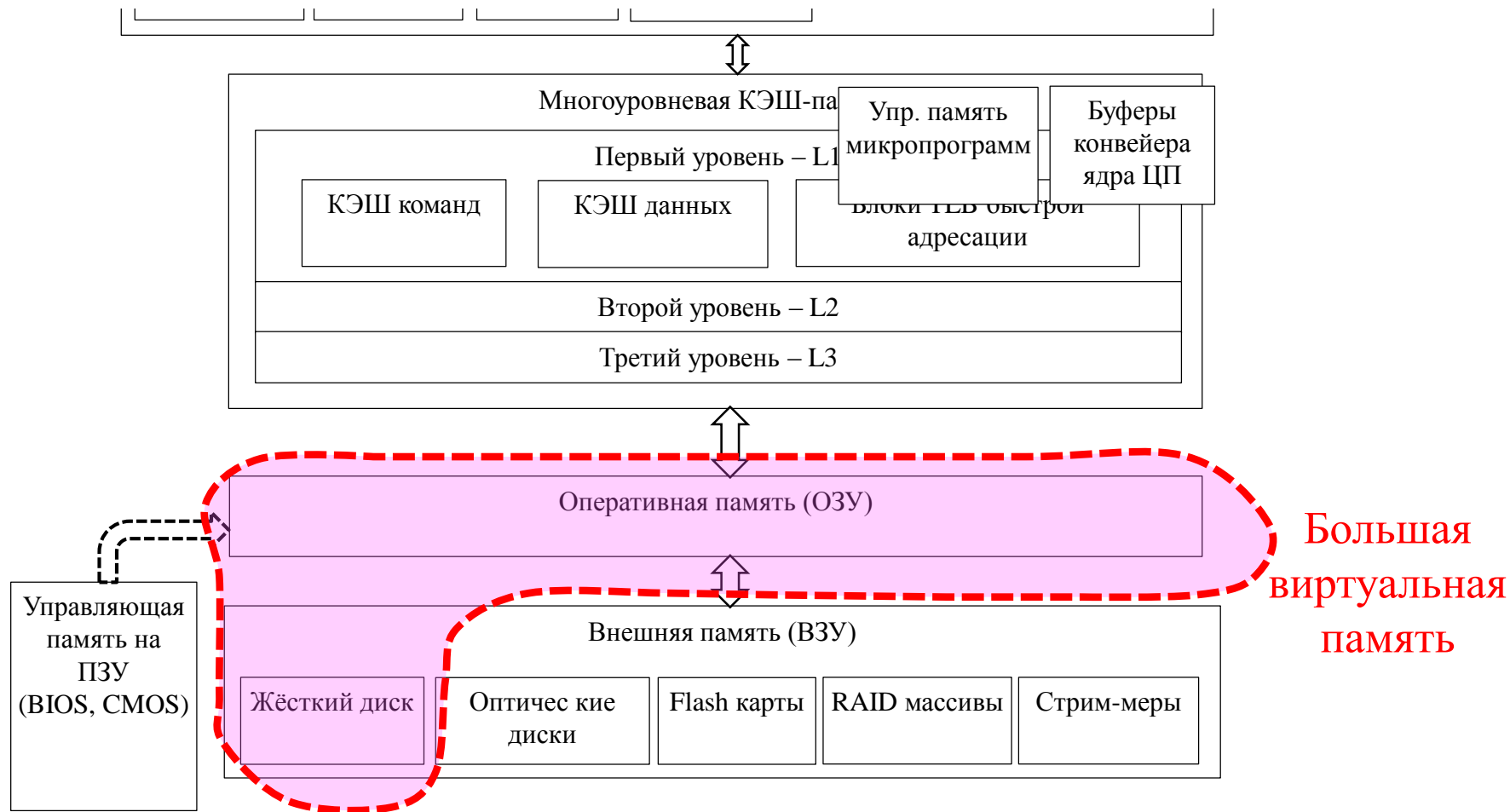
- эффективное использование памяти

### Недостатки:

- процедура сжатия/дефрагментации требует значительного времени, что может свести на нет преимущества данного метода

# МЕТОДЫ УПРАВЛЕНИЯ ОПЕРАТИВНОЙ ПАМЯТЬЮ ВС

## *С использованием внешней памяти*



**Достоинства:** увеличение объёма ОП.

**Недостатки:** сложность управления.

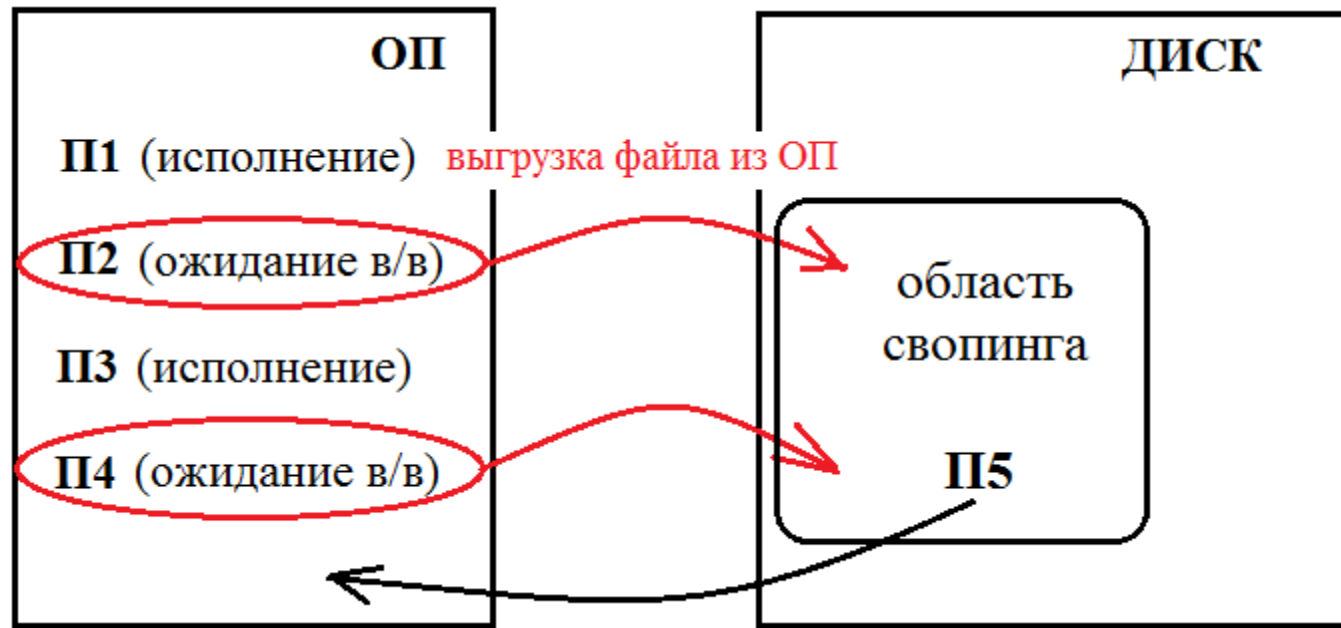
# МЕТОДЫ УПРАВЛЕНИЯ ОПЕРАТИВНОЙ ПАМЯТЬЮ ВС С ИСПОЛЬЗОВАНИЕМ ВНЕШНЕЙ ПАМЯТИ



*Все исполняемые программы и их данные должны находиться в ОП. Если часть задачи находится на диске, то в момент обращения к этой отсутствующей части она перемещается в ОП, а что-то взамен выгружается из ОП на диск.*

# МЕТОДЫ УПРАВЛЕНИЯ ОПЕРАТИВНОЙ ПАМЯТЬЮ ВС

## СВОПИНГ



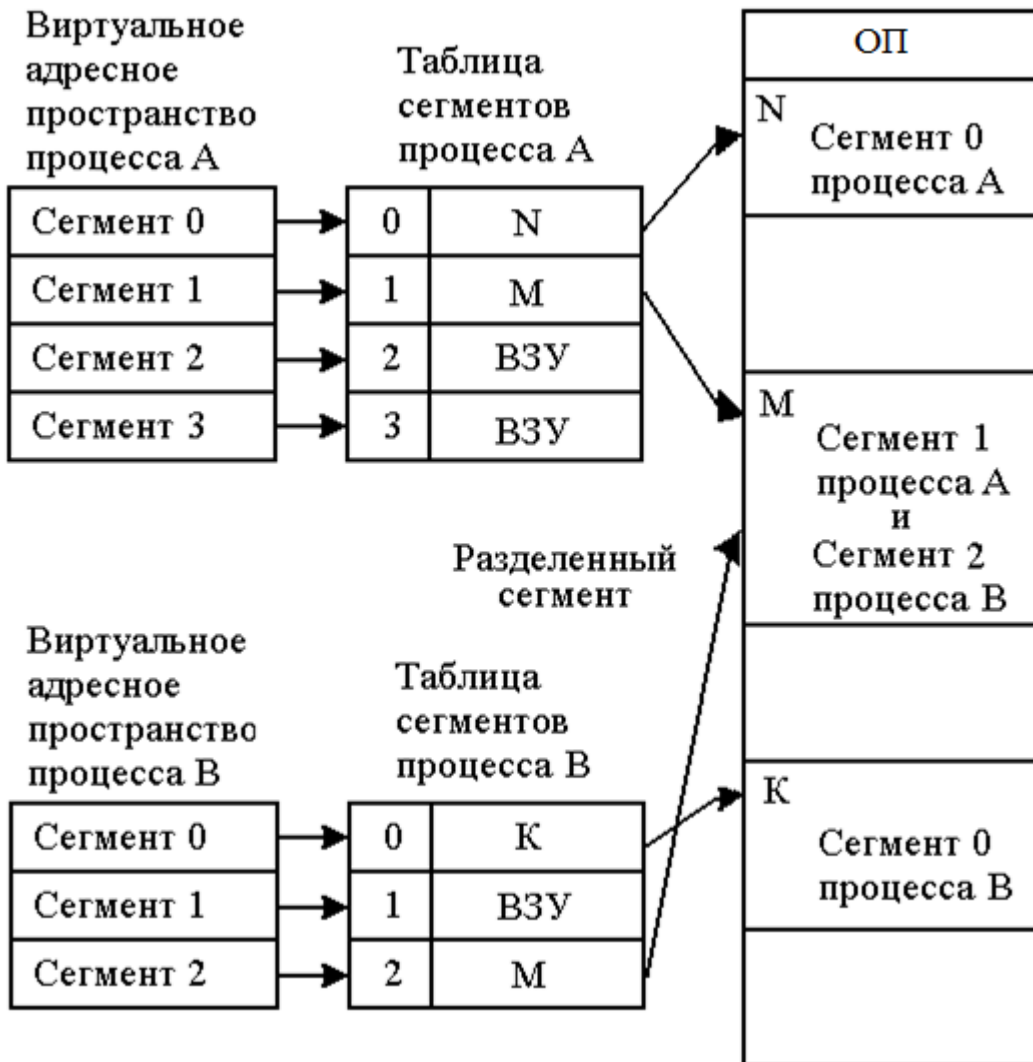
*Некоторые процессы (задачи), обычно находящиеся в состоянии ожидания, целиком могут отгружаться (откачиваться) на диск, а на их место подгружаться другие.* При этом программа-планировщик ОС не исключает их из своего рассмотрения и при наступлении условий, в которых возможно выполнять некоторую задачу, находящуюся в области свопинга на диске, эта задача перемещается в ОП.

Существуют различные алгоритмы отгрузки процессов на диск и подкачивания других процессов, а также различные способы выделения оперативной и дисковой памяти загружаемому процессу.

# МЕТОДЫ УПРАВЛЕНИЯ ОПЕРАТИВНОЙ ПАМЯТЬЮ ВС

## СЕГМЕНТНОЕ РАСПРЕДЕЛЕНИЕ

*Задача делится на части (сегменты) по их смысловому назначению: сегмент кода, сегмент стека, сегменты данных.*



### Задачи п/с управления памятью:

- ведение таблиц сегментов задач (адрес, размер, атрибуты)
- Ведение таблиц свободных разделов
- Анализ запросов и выделение раздела задаче.
- При необходимости выполнение процедуры откачки/подкачки сегмента из/в ОП
- Проверка при подкачке имеется ли раздел достаточного размера

### Достоинства:

- дифференциация способов доступа к разным частям программы (сегментам): например, запрет записи в сегмент кода.
- возможно разделение одного сегмента несколькими задачами.
- Отсутствие неиспользуемой памяти.

### Недостатки:

- Сложность организации (сопоставление размера сегмента и свободного раздела).



# МЕТОДЫ УПРАВЛЕНИЯ ОПЕРАТИВНОЙ ПАМЯТЬЮ ВС

## СТРАНИЧНОЕ РАСПРЕДЕЛЕНИЕ

Задача делится на равные части (страницы) согласно настройкам ЦП и ОС (4 КБ, 2-4МБ).

вирт. адресное пространство задачи1

вирт.стр.0
вирт.стр.1
вирт.стр.2
вирт.стр.3
вирт.стр.4

Таблица страниц пр.1

N <sub>в.с.</sub>	N <sub>ф.с.</sub>	Упр.ин
0	5	
1	ВЗУ	
2	ВЗУ	
3	10	
4	2	

Физическая память	N физ. стр.
	0
	1
4 пр.1	2
	3
	4
0 пр.1	5
	6
	7
0 пр.2	8
	9
	10
5 пр.2	11
	12
	13
	14

вирт. адресное пространство задачи 2

вирт.стр.0
вирт.стр.1
вирт.стр.2
вирт.стр.3
вирт.стр.4
вирт.стр.5

страниц пр.2

N <sub>в.с.</sub>	N <sub>ф.с.</sub>	Упр.ин
0	8	
1	ВЗУ	
2	ВЗУ	
3	ВЗУ	
4	ВЗУ	
5	11	



### Задачи п/с управления памятью:

- Ведение таблиц страниц задач (адрес, размер, атрибуты)
- Ведение таблиц свободных страниц
- Анализ запросов и выделение страниц задаче.
- При необходимости выполнение процедуры откачки/подкачки страниц из/в ОП

### Достоинства:

- Относительная простота реализации – не нужно проверять размер свободной области – все страницы одинаковые.

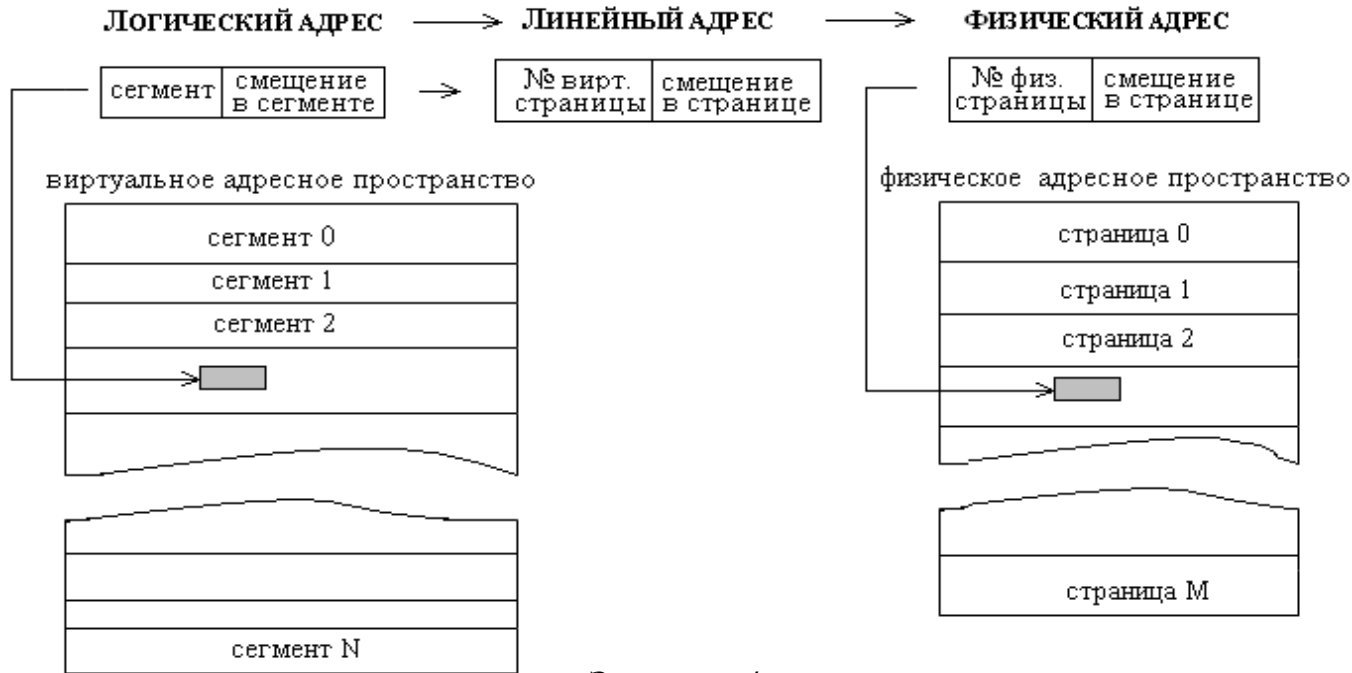
### Недостатки:

- Отсутствие дифференциация способов доступа к разным частям программы.
- Наличие в конце последней страницы неиспользуемой памяти.

$$V_{\text{вирт.стр.}} = V_{\text{физ.стр.}} = 2^k$$

# СТРАНИЧНО-СЕГМЕНТНОЕ РАСПРЕДЕЛЕНИЕ

## Преобразования адресных пространств



### Задачи п/с управления памятью:

- Ведение таблиц сегментов, таблиц страниц, таблиц свободных страниц
- Анализ запросов и выделение страниц задаче,
- Анализ адресов обращений в пределах сегмента.
- Преобразование логических адресов (в пределах сегментов) в виртуальные адреса (в пределах страниц).
- Проверка присутствия страницы в ОП и при необходимости выполнение процедуры откочки/подкачки

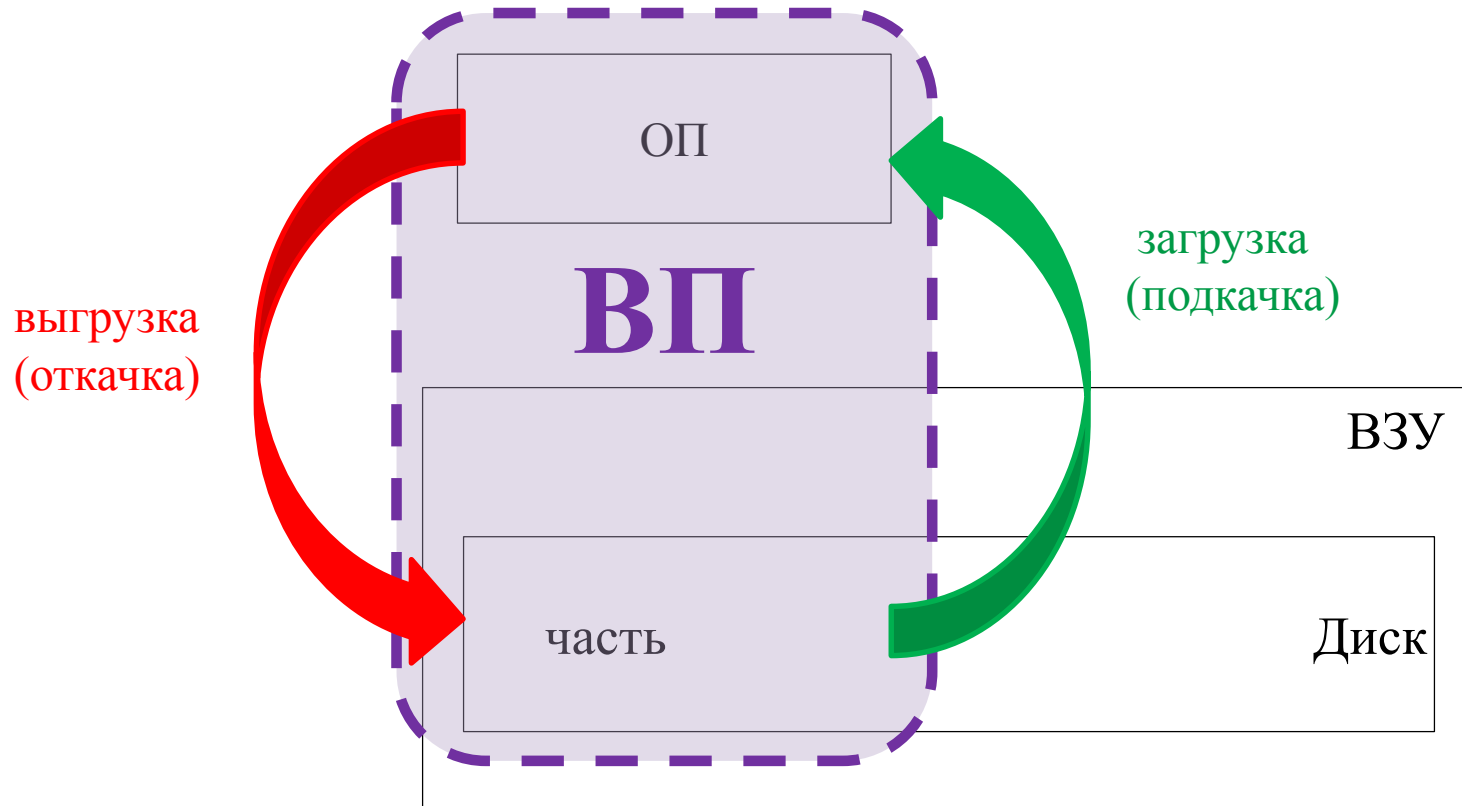
### Достоинства:

- Дифференциация способов доступа к разным сегментам.
- Простота подкачки/откочки страниц из/в ОП – все страницы одинаковые.

### Недостатки:

- Сложность организации.
- Наличие в конце последней страницы неиспользуемой памяти.

# ОБМЕН БЛОКАМИ МЕЖДУ ОП И ДИСКОМ

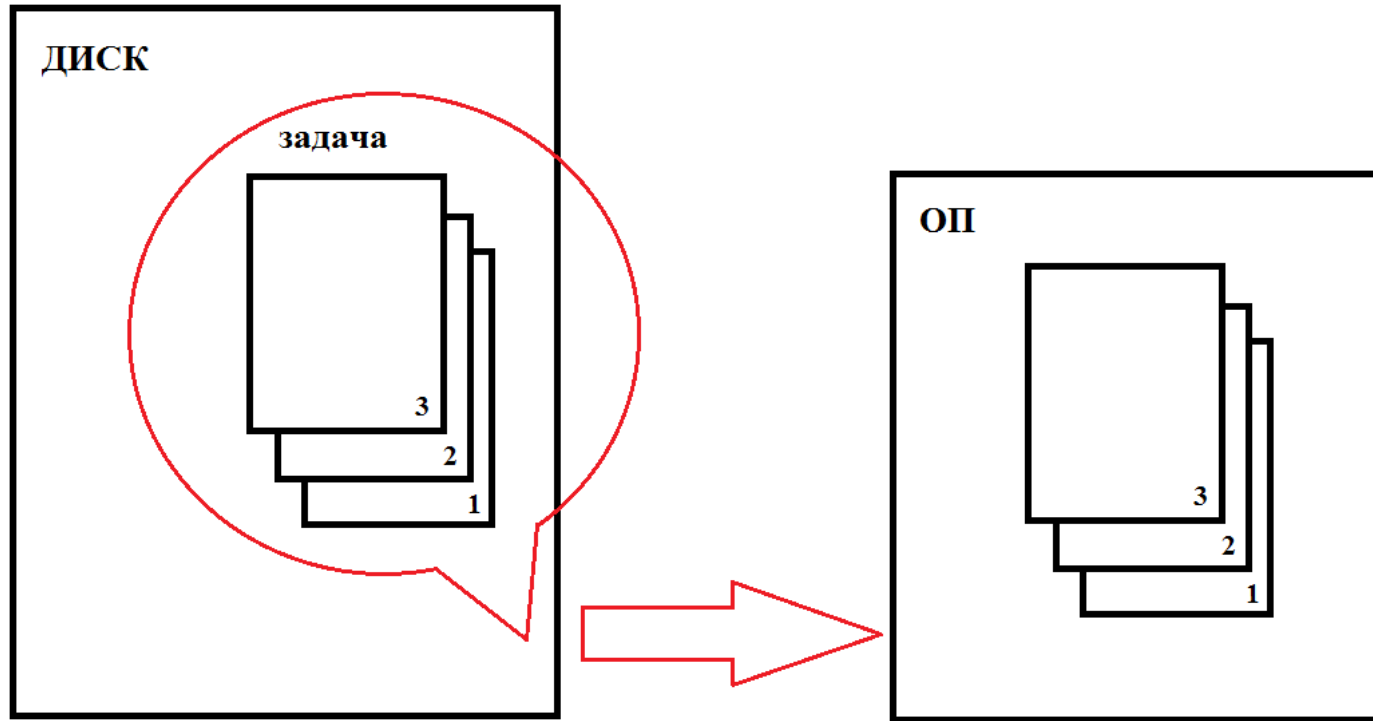


# АЛГОРИТМЫ ПОДКАЧКИ СТРАНИЦ/СЕКМЕНТОВ В ОП

- *Предварительное размещение.*
- *Опережающая подкачка*
- *Подкачка по требованию.*

# АЛГОРИТМЫ ПОДКАЧКИ / ЗАГРУЗКИ.

## Предварительное размещение



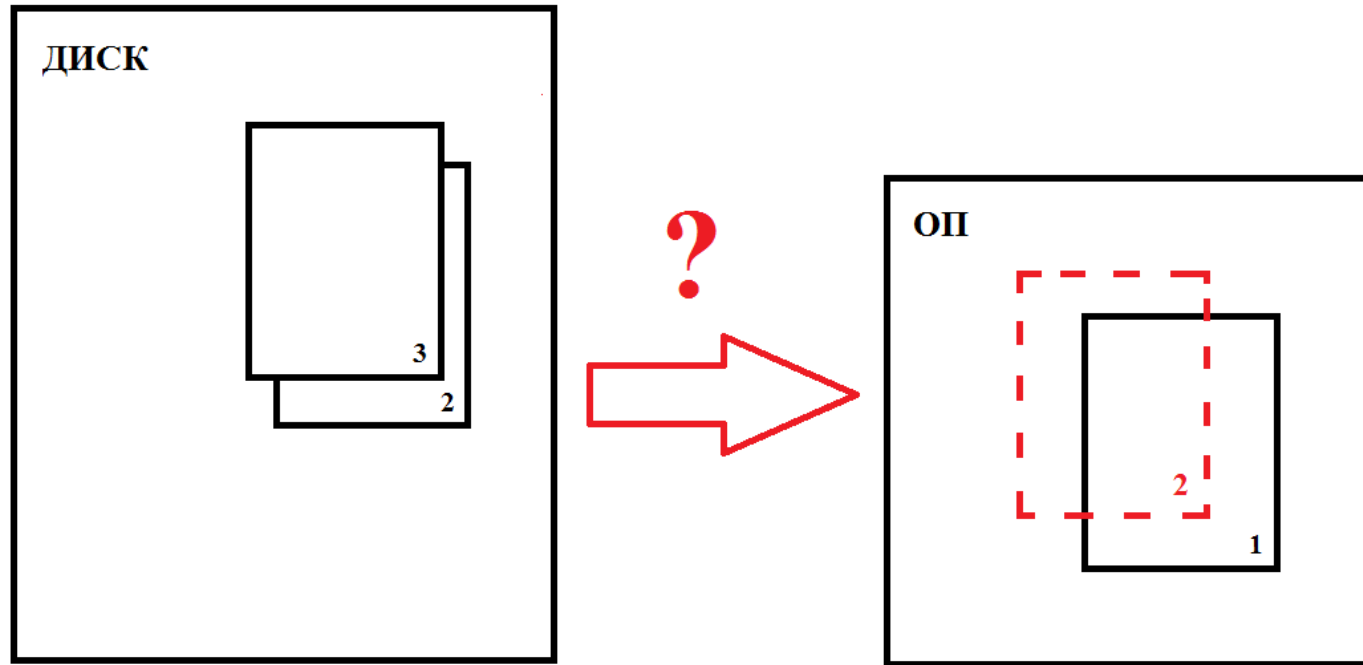
*Все страницы/сегменты задачи заранее размещаются в ОП.*

**Достоинства:** Все страницы/сегменты в наличии, нет прерываний для подгрузки.

**Недостатки:** Снижается коэффициент мультипрограммирования (количество одновременно выполняемых процессов), т.к. каждая задача занимает много места, и в ОП может быть помещено меньшее количество задач. Стратегия не подходит для больших по объему задач.

# АЛГОРИТМЫ ПОДКАЧКИ / ЗАГРУЗКИ.

## Опережающая подкачка



*ОС пытается предсказать, какие страницы/сегменты потребуются и их заранее загрузить*

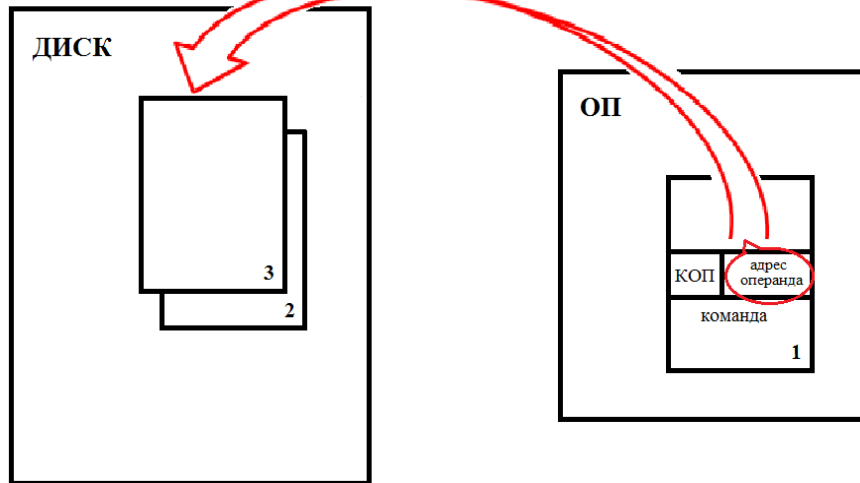
**Достоинства:** Если в большинстве случаев удастся принимать правильное решение о выборе страниц для подкачки, то время выполнения процесса значительно уменьшается.

**Недостатки:** Этот метод эффективен для программ с небольшим количеством переходов, поскольку в этом случае с большой вероятностью следующей понадобится страница с порядковым виртуальным номером на 1 больше, чем у текущей. Такое предсказание сделать легко.

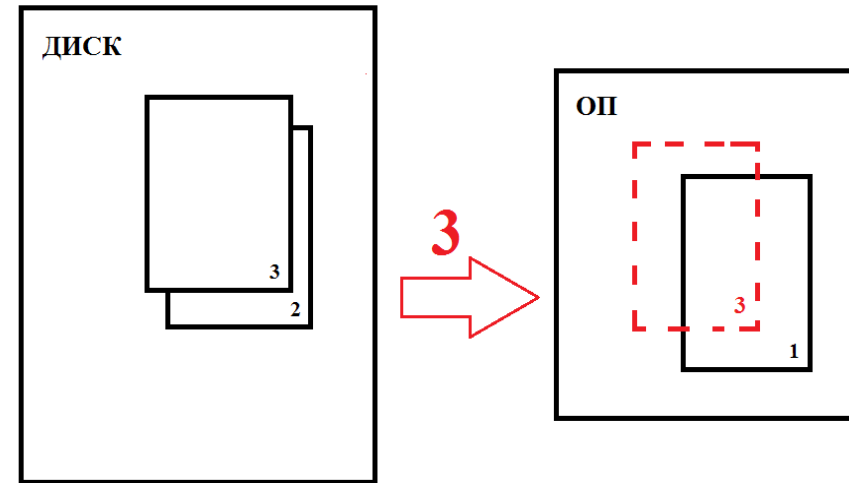
# АЛГОРИТМЫ ПОДКАЧКИ / ЗАГРУЗКИ.

## Подкачка по требованию

*обращение по адресу на стр.3*



а)



б)

*Страница/сегмент загружается в ОП после обращения к ней/нему (по адресу операнда или передачи управления из адресного поля команды) – по прерыванию*

**Достоинства:** Экономия памяти, т.к. метод гарантирует, что в ОП будут переписываться только те страницы, которые фактически необходимы для работы процессов.

**Недостатки:** Процесс должен накапливать в памяти требуемые ему страницы по одной. При появлении ссылки на каждую новую страницу процессу приходится ждать, когда эта страница будет передана в основную память.



# АЛГОРИТМЫ **ОТКАЧКИ** СТРАНИЦ/СЕГМЕНТОВ

- **RND** (RaNDom)  
– *Выталкивание случайной страницы.*
- **FIFO** (First In First Out)  
– *Выталкивание первой пришедшей страницы*
- **LRU** (Least Recently Used)  
– *Выталкивание дольше всего не использовавшейся страницы*
- **LFU** (Least Frequently Used)  
– *Выталкивание реже всего используемой страницы*
- **MRU** (Most Recently Used)  
– *Выталкивание чаще всего используемой страницы*
- **NRU** (Non Recently Used)  
– *учитывающий число и тип обращения*
- ... ..

# Аналогичны алгоритмам замещения строк КЭШ-памяти

*Основные стратегии замещения строк (поиск наименее нужных строк):*

1. алгоритм **RND** (randomize-алгоритм) – выталкивается случайно выбранная строка;
2. алгоритм **LFU** (Least Frequently Used) – выталкивается строка с наименьшим количеством обращений;
3. алгоритм **LRU** (Least Recently Used) – выталкивается давно не используемая строка;
4. алгоритм **MRU** (Most Recently Used) – выталкивается последняя использованная строка; аналог **LRR** = Least Recently Replaced
5. алгоритм **FIFO** (First Input First Output) – выталкивается строка, которая была загружена из ОП раньше всех,
6. комбинации основных алгоритмов:
  - ARC (Adaptive Replacement Cache = LRU+LFU);
  - Clock (Second-Chance)=FIFO+LRU;
  - LRD (Least Reference Density) FIFO+LRU+LFU;
  - LRFU, LIRS, FBR, ALRFU, DBL, OPT, CAR ...

**Самые часто  
используемые**

# АЛГОРИТМЫ ОТКАЧКИ

## Выталкивание случайной строки (RND)



**Достоинство:** простота, малые издержки, равноправие задач.

**Недостатки:** рассчитана на "слепое" везение, в реальных сложных системах она применяется редко, т.к. может отгрузиться именно та строка, которая скоро понадобится.

# АЛГОРИТМЫ ОТКАЧКИ

## Выталкивание первой пришедшей строки (FIFO)



**Достоинство:** простота, малые издержки, равноправие задач.

**Недостатки:** рассчитана на последовательную обработку, не подходит для реальных сложных программ, где множество переходов и циклов.

# АЛГОРИТМЫ ОТКАЧКИ

## Выталкивание дольше всего не использовавшейся строки (LRU)



**Достоинство:** анализ реального порядка использования строк в ВС.

**Недостатки:** сложность хранения, сравнение, обновления временных меток строк. *Возможно, что давно загруженная строка часто используется.*

# АЛГОРИТМЫ ОТКАЧКИ

## Выталкивание строки, к которой мало обращались (LFU)



**Достоинство:** анализ реального порядка использования строк.

**Недостатки:** сложность хранения, сравнение, обновления счетчиков строк. *Возможно, что малое число обращений говорит о том, что со строкой только начали работать и она ещё будет использоваться.*

# АЛГОРИТМЫ ОТКАЧКИ

## Выталкивание строки, к которой много обращались (MRU)



**Достоинство:** анализ реального порядка использования строк.

**Недостатки:** сложность хранения, сравнение, обновления счетчиков строк. *Возможно, что недавно загруженная строка ещё будет использоваться.*



# АЛГОРИТМЫ ОТКАЧКИ

## Адаптивный алгоритм ARC (LRU + LFU)

стр.1
стр.2
стр.3
стр.4
стр.5
...
стр.8

**ARC стек**  
(счетчик+время загрузки)

№ стр.	счетчик обращений	временная метка
1	5	35
2	11	21
3	10	100
4	6	58
5	10	200
6	3	68
7	3	226



*Запрос на размещение  
новой строки в КЭШ*



*определение номера  
выгружаемой строки*

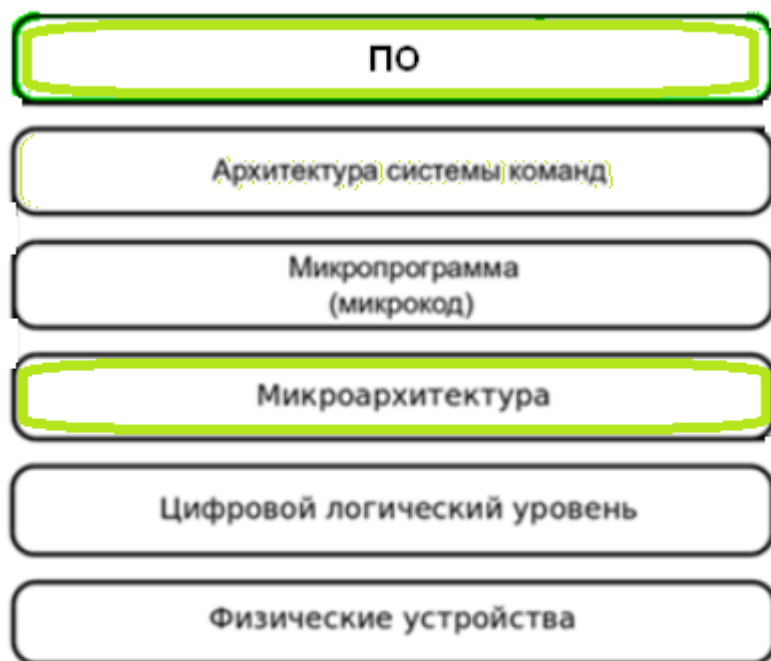
недавно  
давно

**Достоинство:** анализ реального порядка использования строк.

**Недостатки:** сложность хранения, сравнение, обновления счетчиков и временных меток строк.

# ЗАЩИТА ПАМЯТИ

- Защита на верхнем уровне иерархии архитектуры (программно-аппаратный)



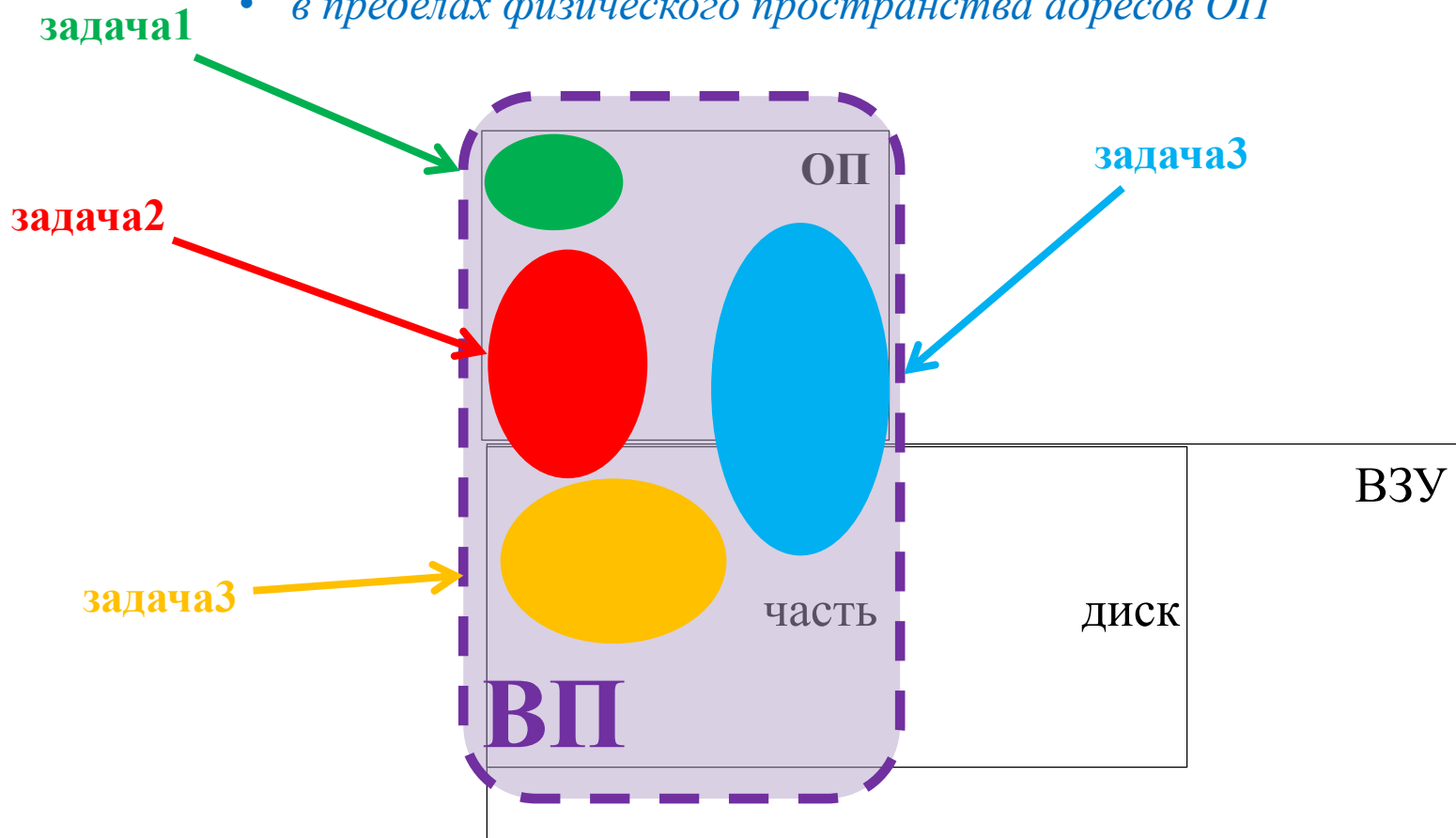
- Защита на нижнем уровне архитектуры (в микросхеме ОЗУ)

# МЕТОДЫ ЗАЩИТЫ ОП:

## сегментно-страничное распределение памяти

*Множество задач (программ) размещаются в произвольных областях ВП (ОП). В процессе вычислений местоположение страниц/сегментов задачи может меняться. ВС должна обеспечивать автоматическую перенастройку и проверку адресов (без участия пользователя):*

- в пределах виртуального пространства адресов*
- в пределах физического пространства адресов ОП*



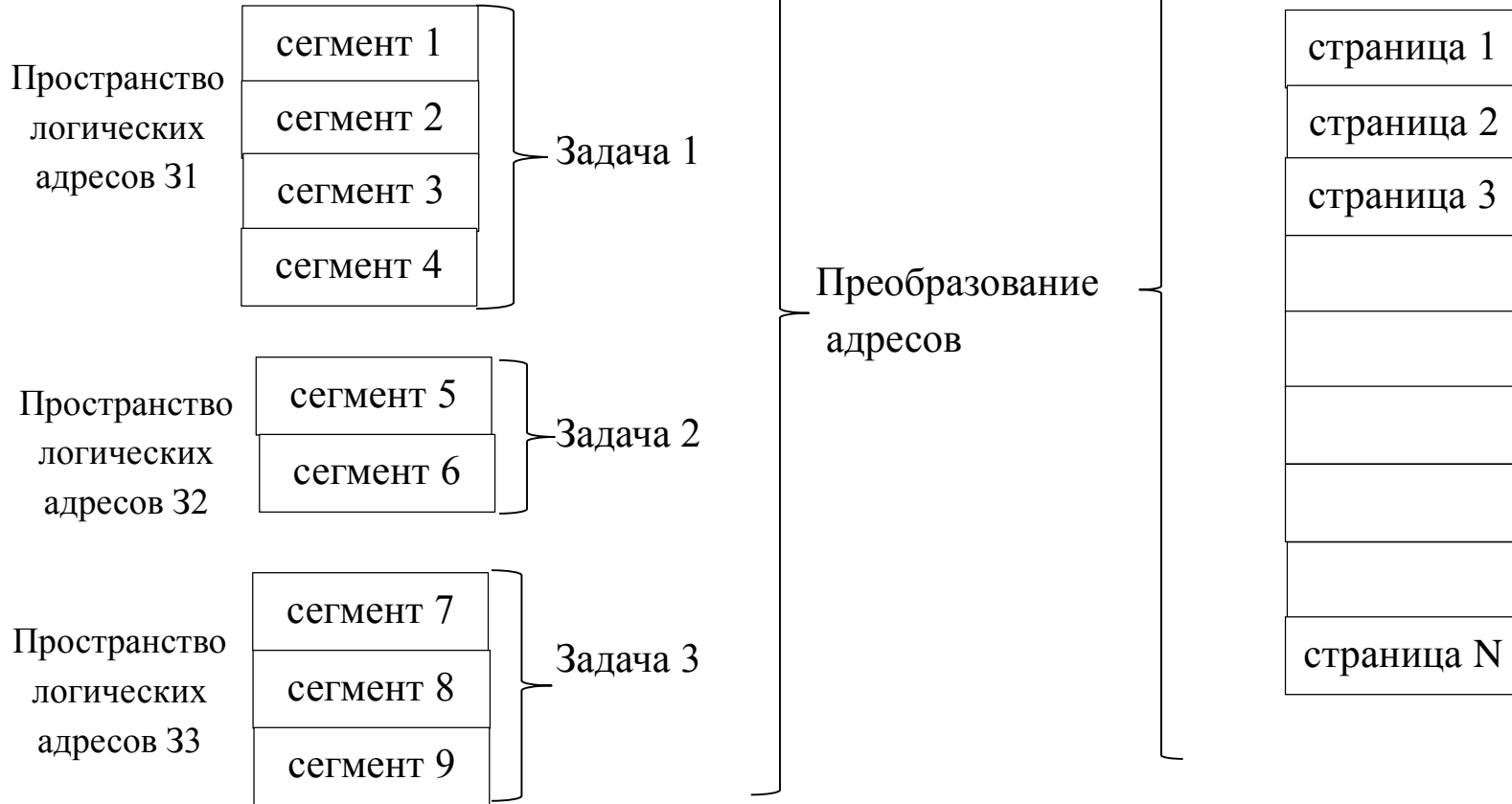
# ПРЕОБРАЗОВАНИЕ АДРЕСОВ

## Логический адрес :

- ссылка на сегмент *и*
- адрес данных в сегменте

## Физический адрес :

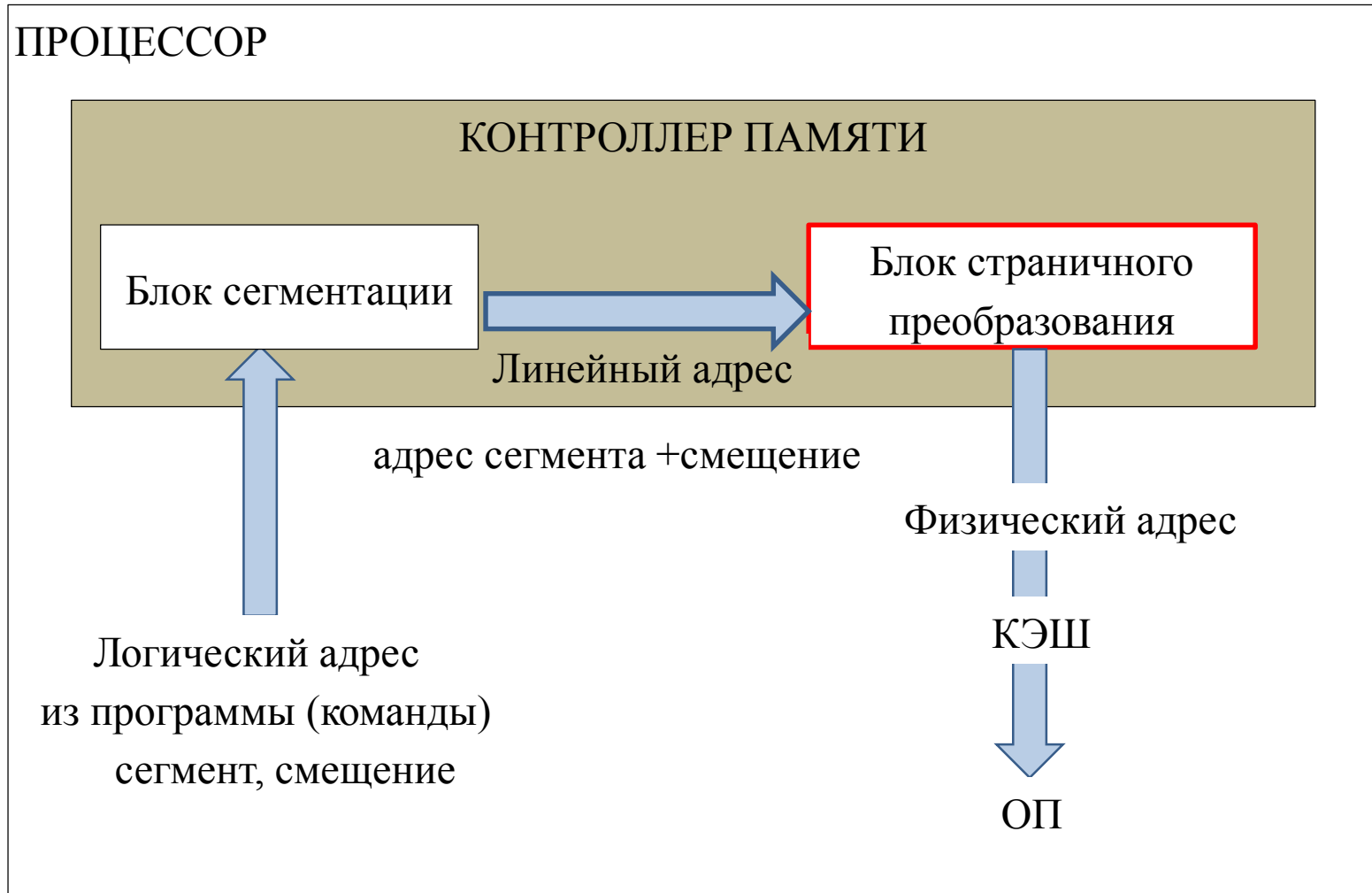
№ ячейки ОП= № страницы *и*  
адрес байта в странице



Виртуальное адресное пространство задач

Пространство физических адресов (ОП)

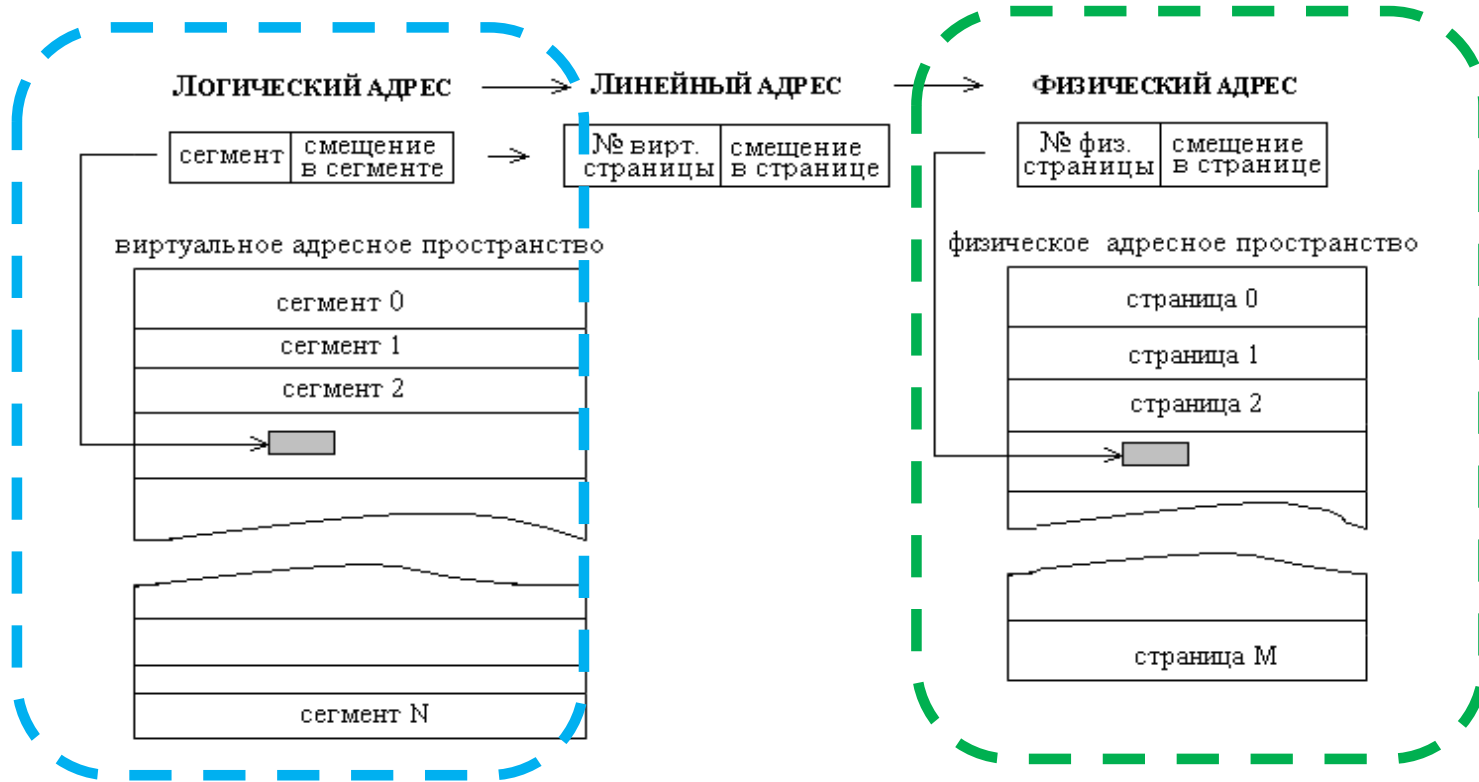
# ПРЕОБРАЗОВАНИЕ АДРЕСОВ



# ЗАЩИТА ПАМЯТИ. ВЕРХНИЙ УРОВЕНЬ

*Задача делится на сегменты, которые размещаются в страницах ОП.*

*В процессе работы ВС происходит **постоянное преобразование адресных пространств***



**Защита на уровне сегментов**  
(в пределах пространства  
логических адресов)

**Защита на уровне страниц**  
(в пределах пространства  
физических адресов)

# ЗАЩИТА ПАМЯТИ. ВЕРХНИЙ УРОВЕНЬ

## Защита на уровне адресов сегментов.

### Логический адрес

Указатель на сегмент

Адрес внутри сегмента

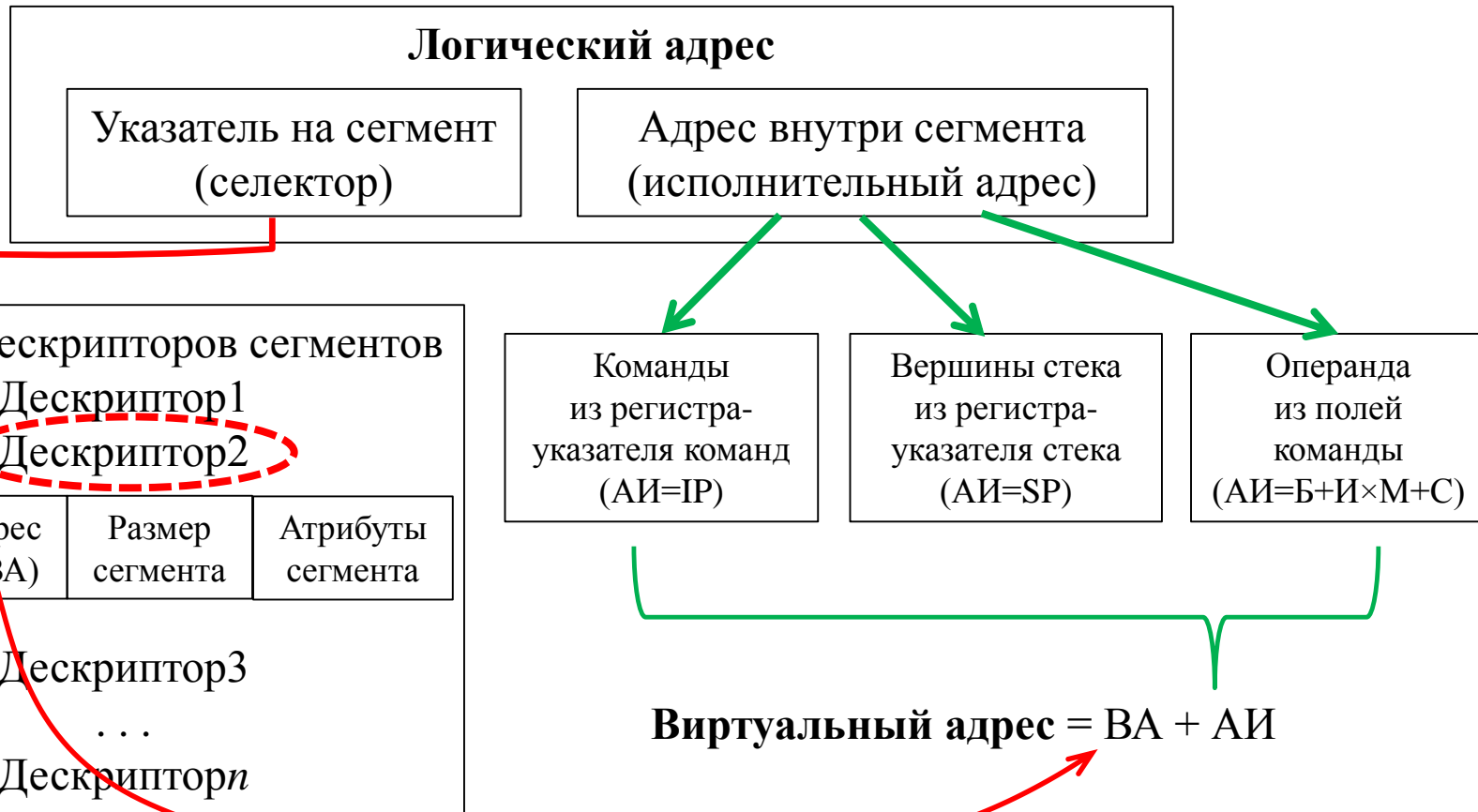
```
04 07 ADD AL,7
66:05 0903 ADD AX,309
05 FFFFFFF0 ADD EAX,0FFFFFFF
80C3 04 ADD BL,4
66:81C2 E703 ADD DX,0E7
81C3 63A70F00 ADD EBX,0FA763
02D0 ADD DL,AL
03D0 ADD EDX,EAX
0308 ADD ECX,DWORD PTR DS:[EAX*4+ESI]
030C86 ADD ECX,DWORD PTR DS:[EAX*4+ESI]
030D 00304000 ADD ECX,DWORD PTR DS:[403000]
038C86 00304000 ADD ECX,DWORD PTR DS:[EAX*4+ESI+403000]
8100 4D010000 ADD DWORD PTR DS:[EAX],14D
810486 4D010000 ADD DWORD PTR DS:[EAX*4+ESI],14D
8105 00304000 4D010000 ADD DWORD PTR DS:[403000],14D
810486 00304000 4D010000 ADD DWORD PTR DS:[EAX*4+ESI+403000],14D
```

Задача не может защититься от несанкционированного доступа других задач, но может **ПРОВЕРИТЬ СОБСТВЕННЫЕ АДРЕСА** обращений на нарушение разрешенных границ (выход за границы сегмента)



# СТРАНИЧНО-СЕГМЕНТНОЕ РАСПРЕДЕЛЕНИЕ.

## Виртуальные адреса

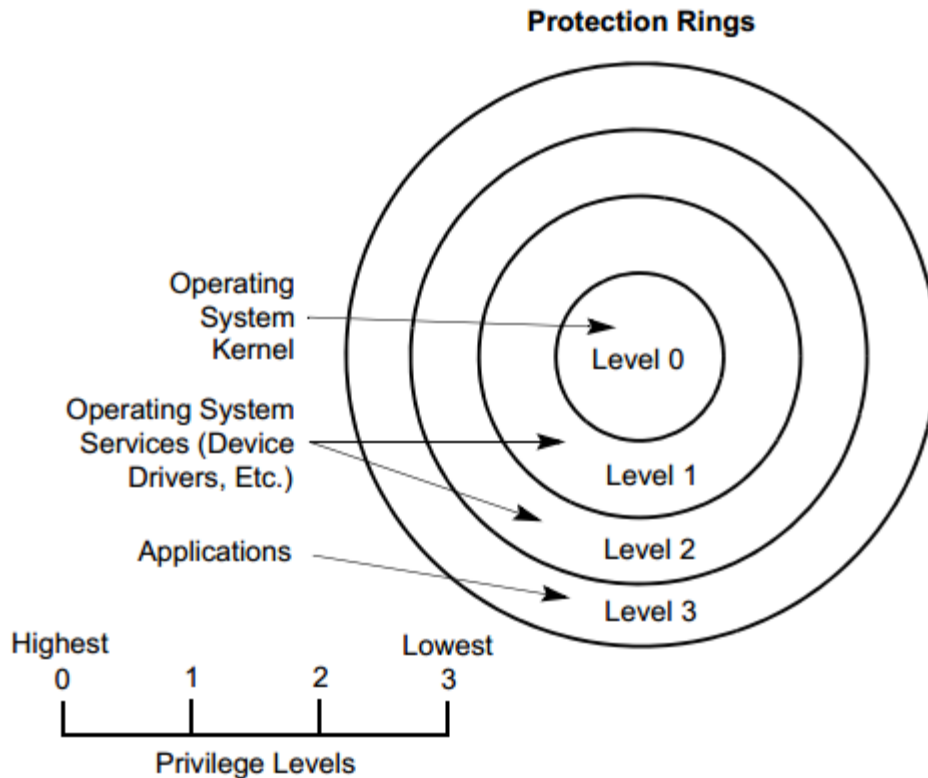


# СТРАНИЧНО-СЕГМЕНТНОЕ РАСПРЕДЕЛЕНИЕ. ЗАЩИТА НА УРОВНЕ СЕГМЕНТОВ

- Проверка прав доступа задачи к сегменту ВП (защита по привилегиям)
- Проверка на обращение в разрешенную область ВП ( $AI \leq Limit$  – выход за границы сегмента)

# СТРАНИЧНО-СЕГМЕНТНОЕ РАСПРЕДЕЛЕНИЕ. ЗАЩИТА НА УРОВНЕ СЕГМЕНТОВ

## *защита по привилегиям*



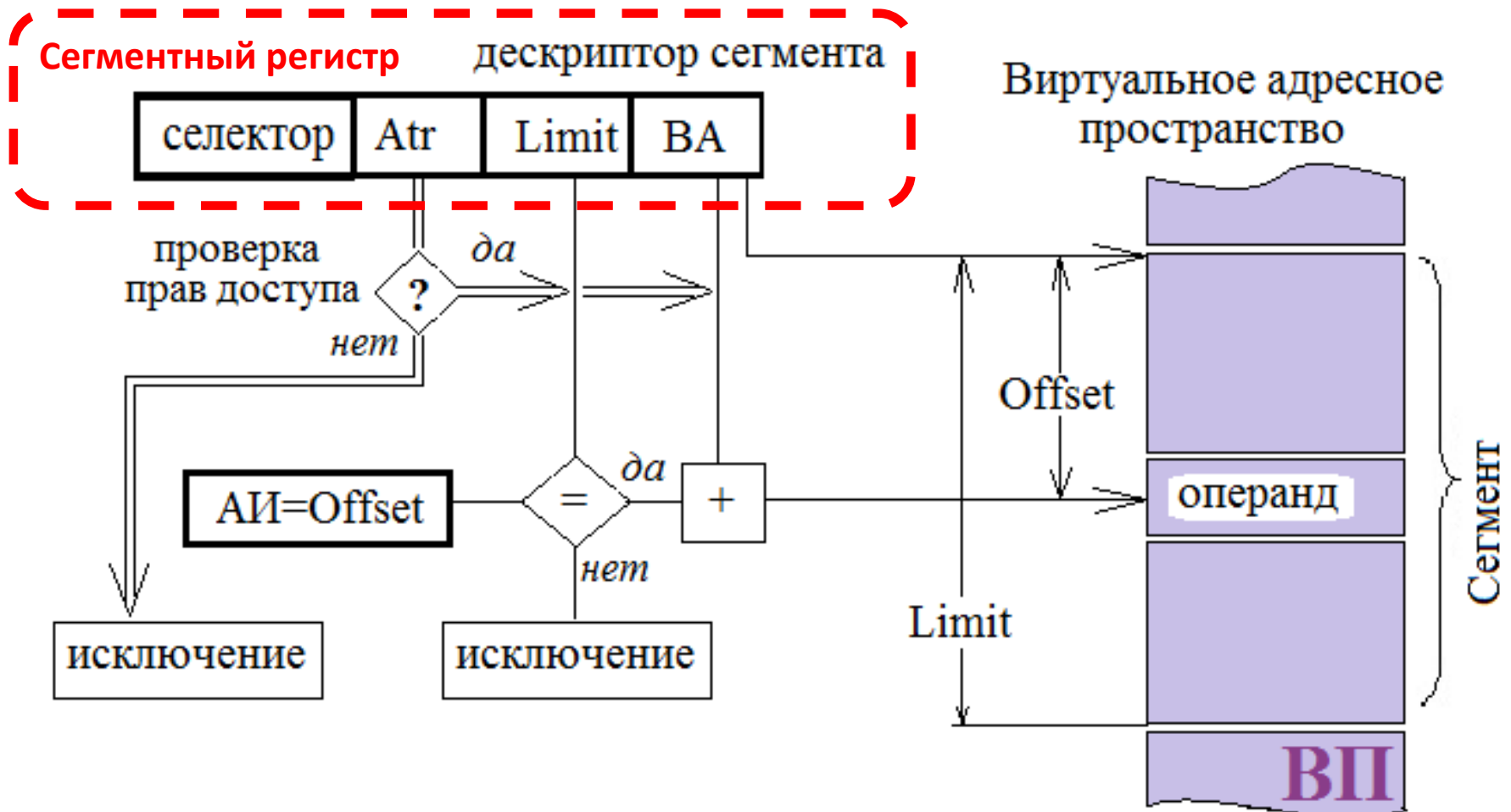
*Любая задача с уровнем привилегии «0» имеет доступ ко всем сегментам с уровнями «0», «1», «2», «3».*

*И наоборот, задача с уровнем привилегий «3» имеет доступ к сегментам только с уровнем не выше «3».*

*Таким образом, сегменты ОС имеют максимальную защиту.*

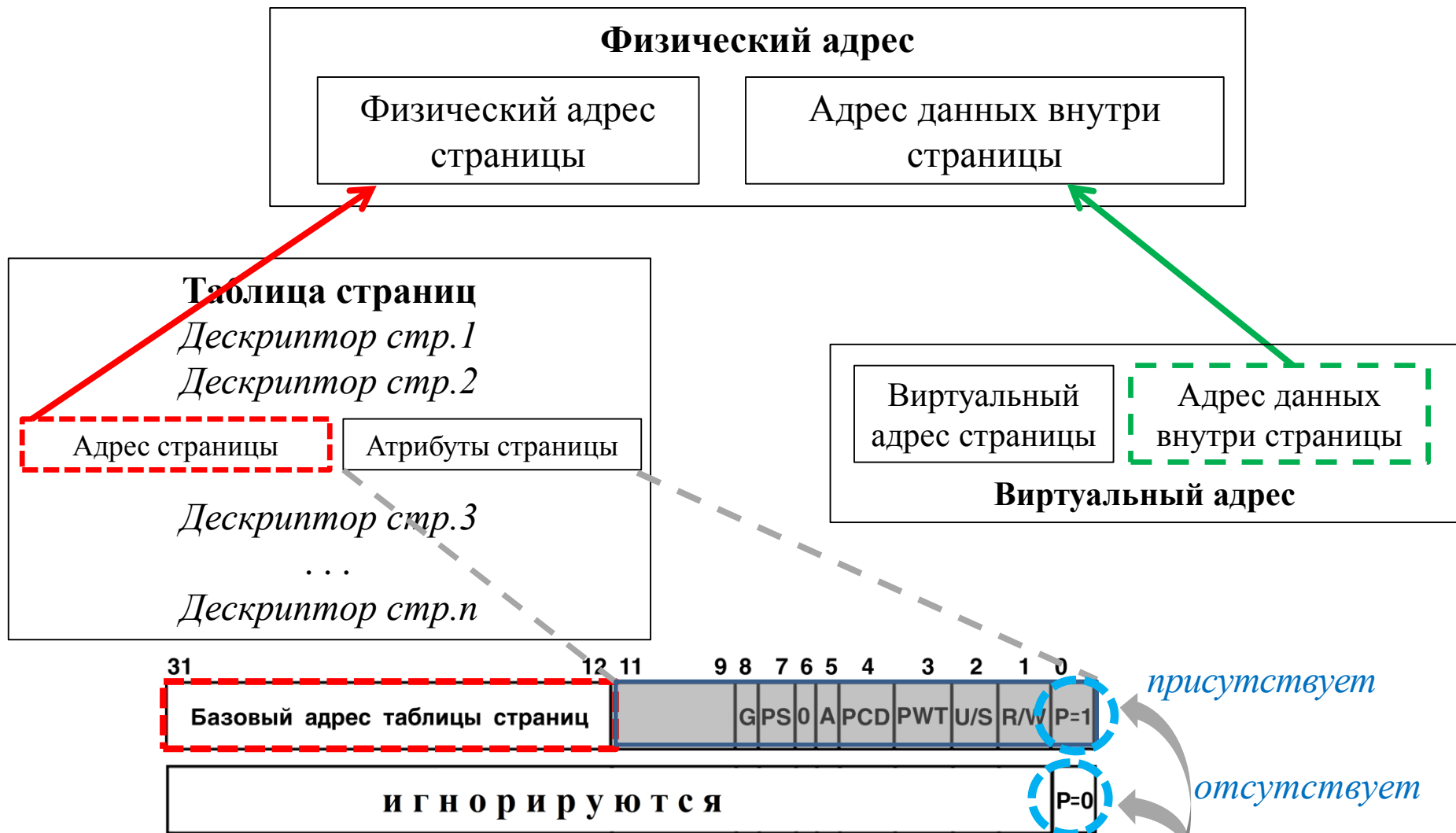
# СТРАНИЧНО-СЕГМЕНТНОЕ РАСПРЕДЕЛЕНИЕ. ЗАЩИТА НА УРОВНЕ СЕГМЕНОВ.

*Проверка на обращение в разрешенную область*



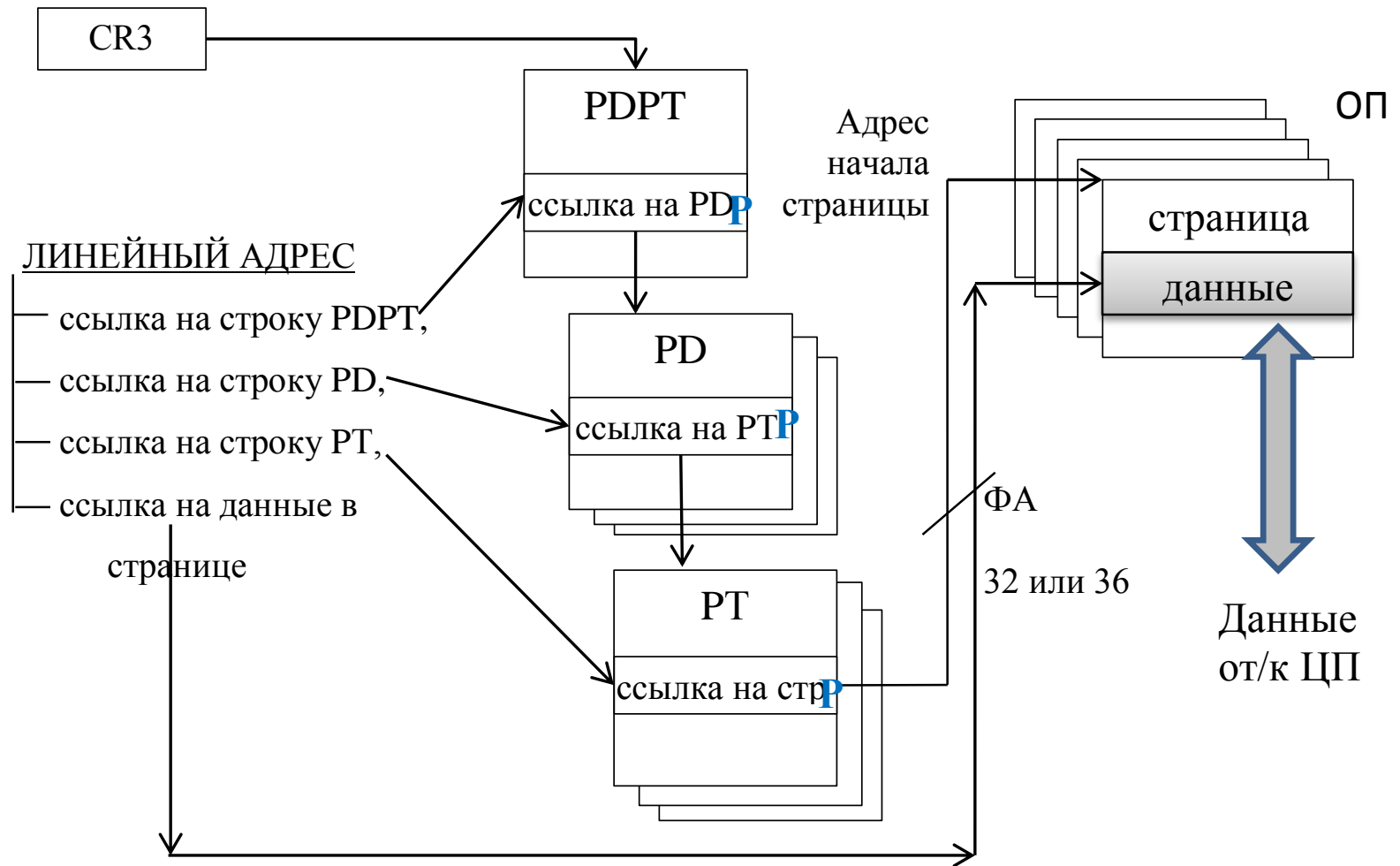
# СТРАНИЧНО-СЕГМЕНТНОЕ РАСПРЕДЕЛЕНИЕ.

## ЗАЩИТА НА УРОВНЕ СТРАНИЦ (бит P)



Проверка присутствия страницы в ОП

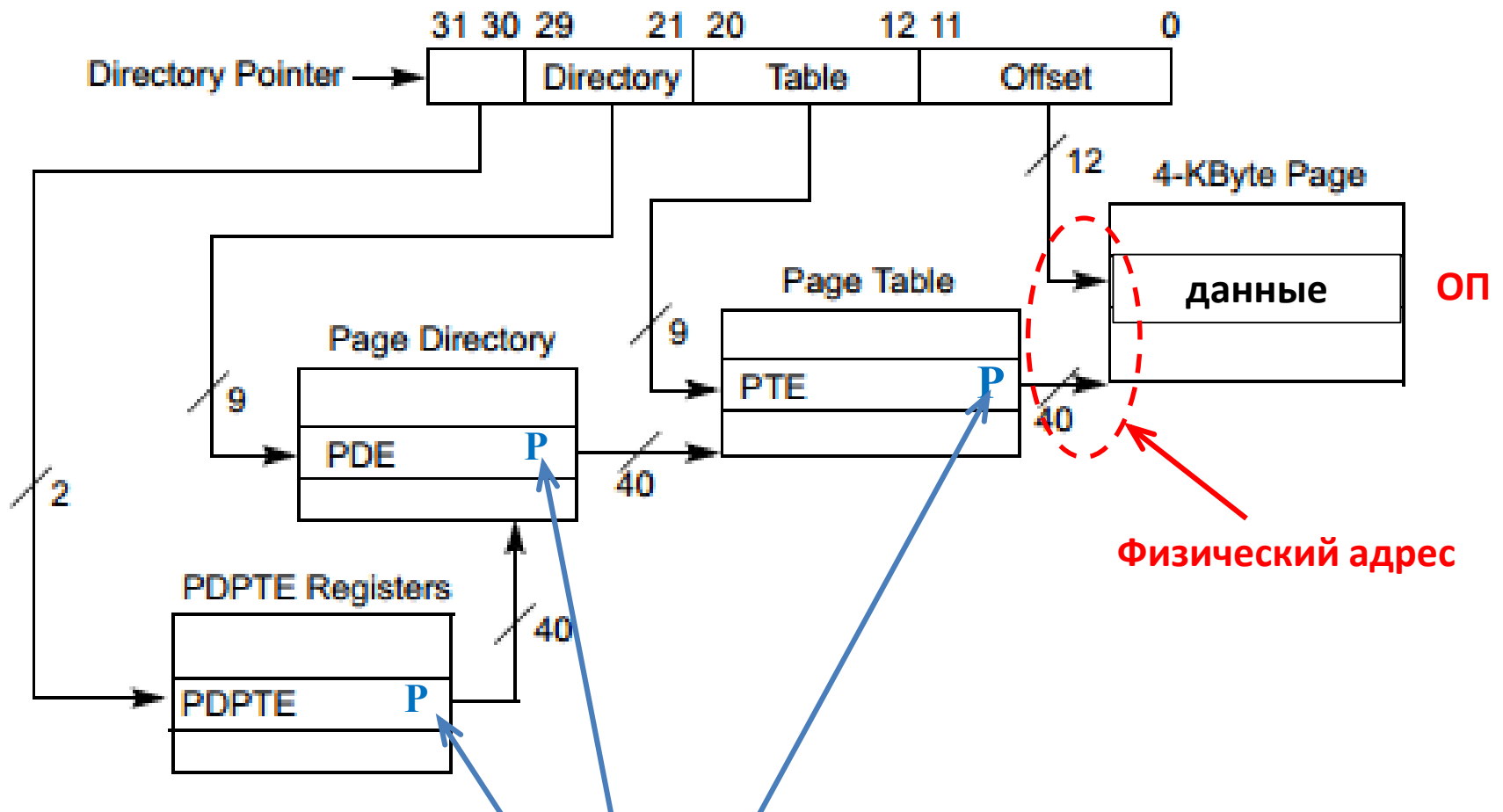
# МЕХАНИЗМЫ СТРАНИЧНОЙ ПЕРЕАДРЕСАЦИИ



Многоэтапная трансляция линейного (виртуального адреса) в физический

Логический адрес

Виртуальный адрес



Проверка присутствия страницы в ОП



ПО

Архитектура системы команд

Микропрограмма  
(микрокод)

Микроархитектура

Цифровой логический уровень

Физические устройства

# ЗАЩИТА ПАМЯТИ НА НИЖНЕМ УРОВНЕ АРХИТЕКТУРЫ (в микросхеме ОЗУ)

*Защита от отказов и сбоев микросхем ОЗУ*

**Отказы:** в отдельных разрядах одной или нескольких ячеек постоянно считывается 0 или 1, вне зависимости от реально записанной туда информации.

## Причины отказов:

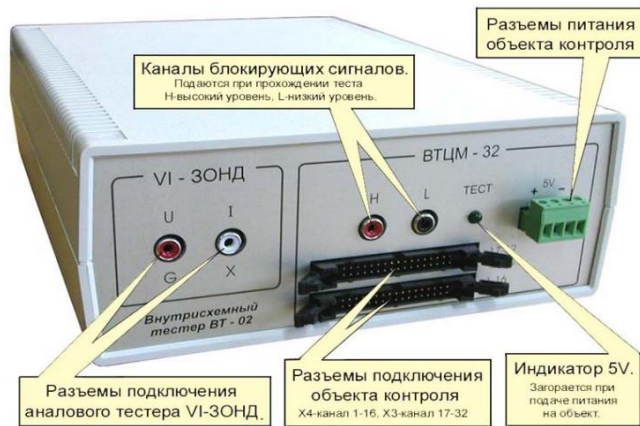
- производственные дефекты,
- повреждение микросхем
- физический износ.

## Способы контроля:

- специализированные функциональные устройства-тестеры
- программы-тестеры аппаратного обеспечения ВС
- диагностика проблем с памятью компьютера средствами ОС

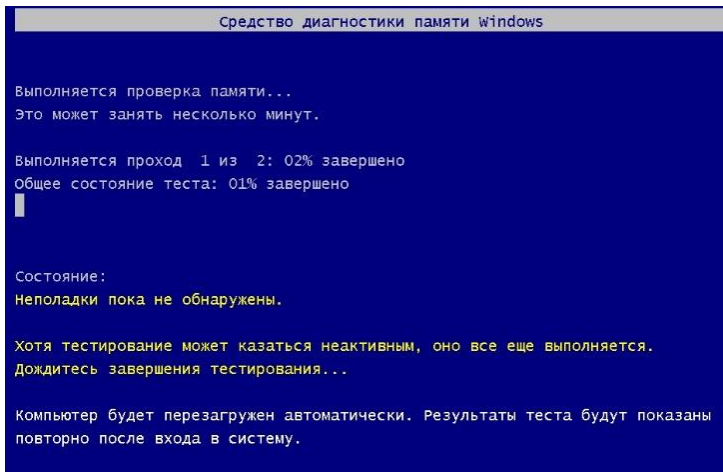
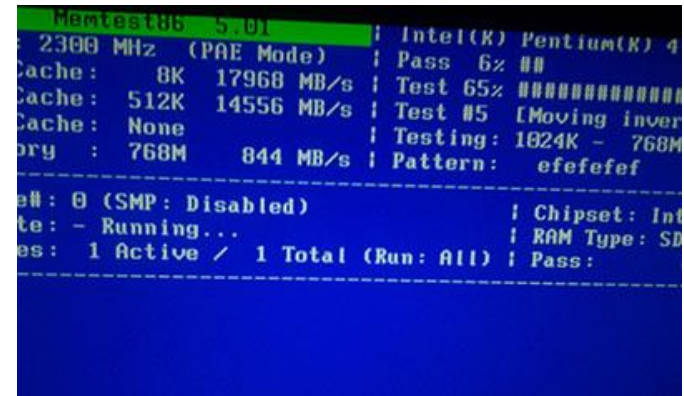
# ДИАГНОСТИКА ОТКАЗОВ МИКРОСХЕМ ОЗУ

## 1. специализированные функциональные устройства - тестеры



## 2. программы-тестеры аппаратного обеспечения ВС

- Memtest86+



## 3. диагностика проблем с памятью компьютера средствами ОС

ПО

Архитектура системы команд

Микропрограмма  
(микрокод)

Микроархитектура

Цифровой логический уровень

Физические устройства

# ЗАЩИТА ПАМЯТИ НА НИЖНЕМ УРОВНЕ АРХИТЕКТУРЫ (в микросхеме ОЗУ)

*Защита от сбоев микросхем ОЗУ*

**Сбой:** случайное событие, выражающееся в неверном считывании или записи информации в отдельных разрядах одной или нескольких ячеек, не связанное с дефектами микросхемы.

## Причины сбоя:

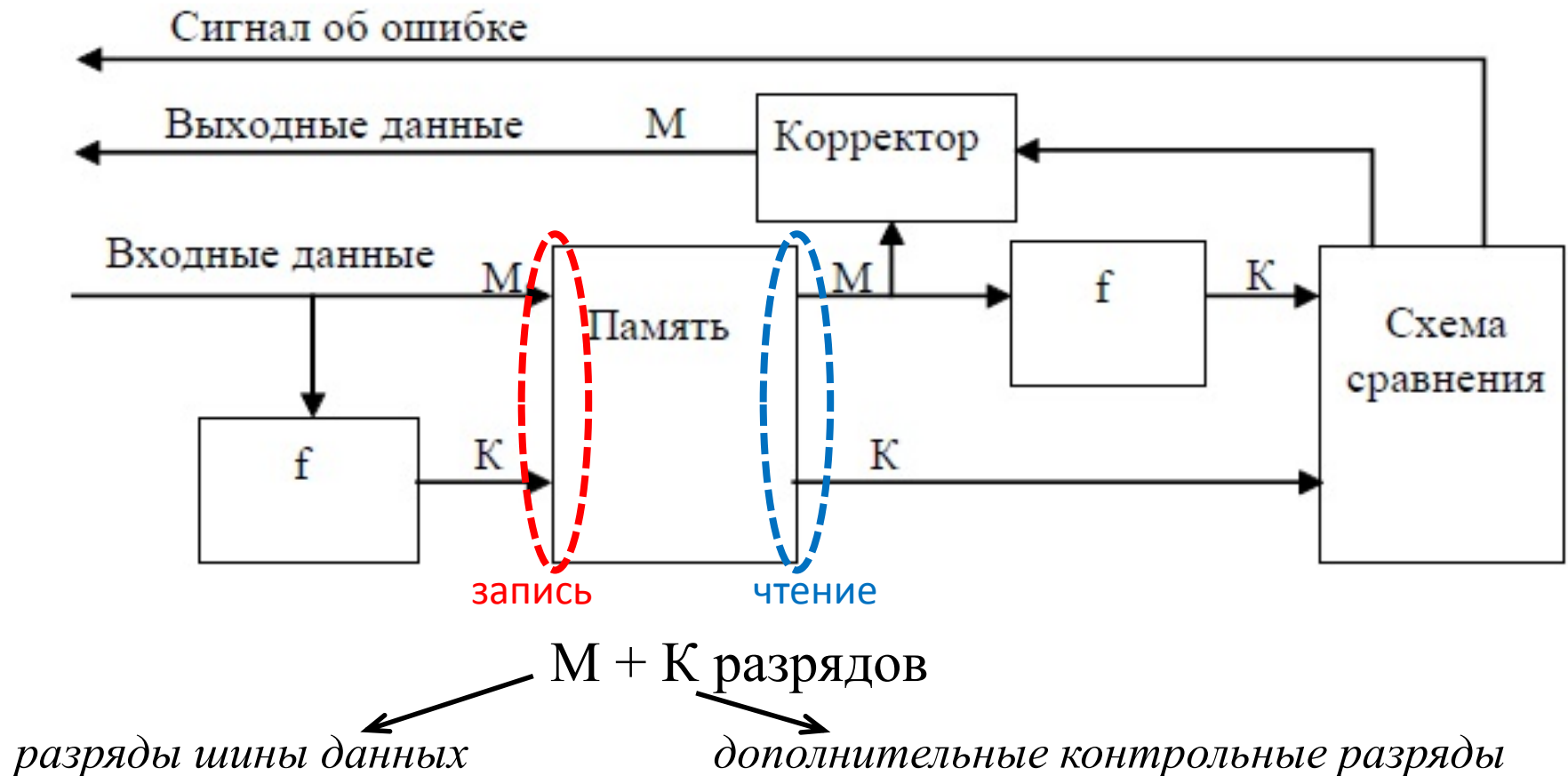
- проблемы с источником питания
- воздействие излучений (альфа-частиц), которые в небольших количествах присутствуют практически везде.

## Способы контроля:

Внутрисхемные модули для обнаружения и исправления ошибок.

# ОБНАРУЖЕНИЯ И ИСПРАВЛЕНИЯ ОШИБОК В МИКРОСХЕМАХ ОЗУ

*Вне зависимости от того, как именно реализуется контроль и исправление ошибок, в основе их всегда лежит введение избыточности. Это означает, что контролируемые разряды дополняются контрольными разрядами, благодаря которым и возможно детектирование ошибок, а в ряде методов - их коррекция.*



# ОБНАРУЖЕНИЯ И ИСПРАВЛЕНИЯ ОШИБОК В МИКРОСХЕМАХ ОЗУ

## Результаты сравнения:

- Не обнаружено ни одной ошибки. Извлеченные из ячейки данные подаются на выход памяти.
- Обнаружена ошибка, и она может быть исправлена. Биты данных и добавочного кода подаются на схему коррекции. После исправления ошибки данные поступают на выход памяти.
- Обнаружена ошибка, и она не может быть исправлена. Выдается сообщение о неисправимой ошибке.

# ОБНАРУЖЕНИЯ И ИСПРАВЛЕНИЯ ОШИБОК В МИКРОСХЕМАХ ОЗУ

## корректирующие коды (Error Correcting Code — ECC)

### Простейший пример :

основан на добавлении к каждому байту информации одного бита паритета.

№ бита	7	6	5	4	3	2	1	0	Бит паритета
Число 23	0	0	0	1	0	1	1	1	1

*Бит паритета - это дополнительный бит, значение которого устанавливается таким, чтобы суммарное число единиц в данных, с учетом этого дополнительного разряда, было четным (или нечетным). В ряде систем за основу берется четность, в иных - нечетность.*

Простейший вариант корректирующего кода также может быть построен на базе битов паритета. Для этого биты данных представляются в виде матрицы, к каждой строке и столбцу которой добавляется бит паритета. Для 64-разрядных данных этот подход иллюстрирует таблица ниже.

# ОБНАРУЖЕНИЯ И ИСПРАВЛЕНИЯ ОШИБОК В МИКРОСХЕМАХ ОЗУ

## корректирующие коды (Error Correcting Code — ECC)

вариант ECC для 8-байтового слова

	0	1	2	3	4	5	6	7	
0	D0	D1	D2	D3	D4	D5	D6	D7	C0
1	D8	D9	D10	D11	D12	D13	D14	D15	C1
2	D16	D17	D18	D19	D20	D21	D22	D23	C2
3	инвертировать бит				D28	D29	D30	D31	C3
4	D32	D33	D34	D35	D36	D37	D38	D39	C4
5	D40	D41	D42	D43	D44	D45	D46	D47	C5
6	D48	D49	D50	D51	D52	D53	D54	D55	C6
7	D56	D57	D58	D59	D60	D61	D62	D63	C7
	K0	K1	K2	K3	K4	K5	K6	K7	P

D - биты данных,  
C - столбец битов паритета строк,  
K - строка битов паритета столбцов,  
P - бит паритета, контролирующий столбец C и строку K.

**M + K разрядов**

M = 64 (разряды данных)

K = 17 (8 бит на строки + 8 бит на столбцы + 1 бит контроля строки и столбца битов паритета).

**Недостаток:**

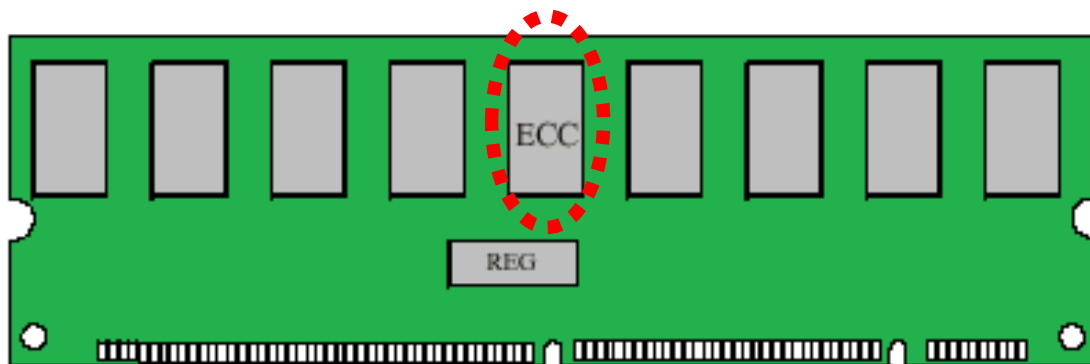
большое число доп. разрядов.

единичное нарушение паритета

**ВЫВОД:** использовать более эффективные коды (например, код Хэмминга).

# ОБНАРУЖЕНИЯ И ИСПРАВЛЕНИЯ ОШИБОК В МИКРОСХЕМАХ ОЗУ

Модули памяти **DDR** с коррекцией ошибок  
(**ECC** - error-correcting code)



Схемы ECC –  
дополнительные  $\Rightarrow$  удорожание ОЗУ  $\Rightarrow$  в серверах  
логические блоки

Samsung DDR4 2666 16 ГБ - 4 340 руб.

Samsung DDR4 2133 ECC DIMM 16Gb - 11 361 руб.