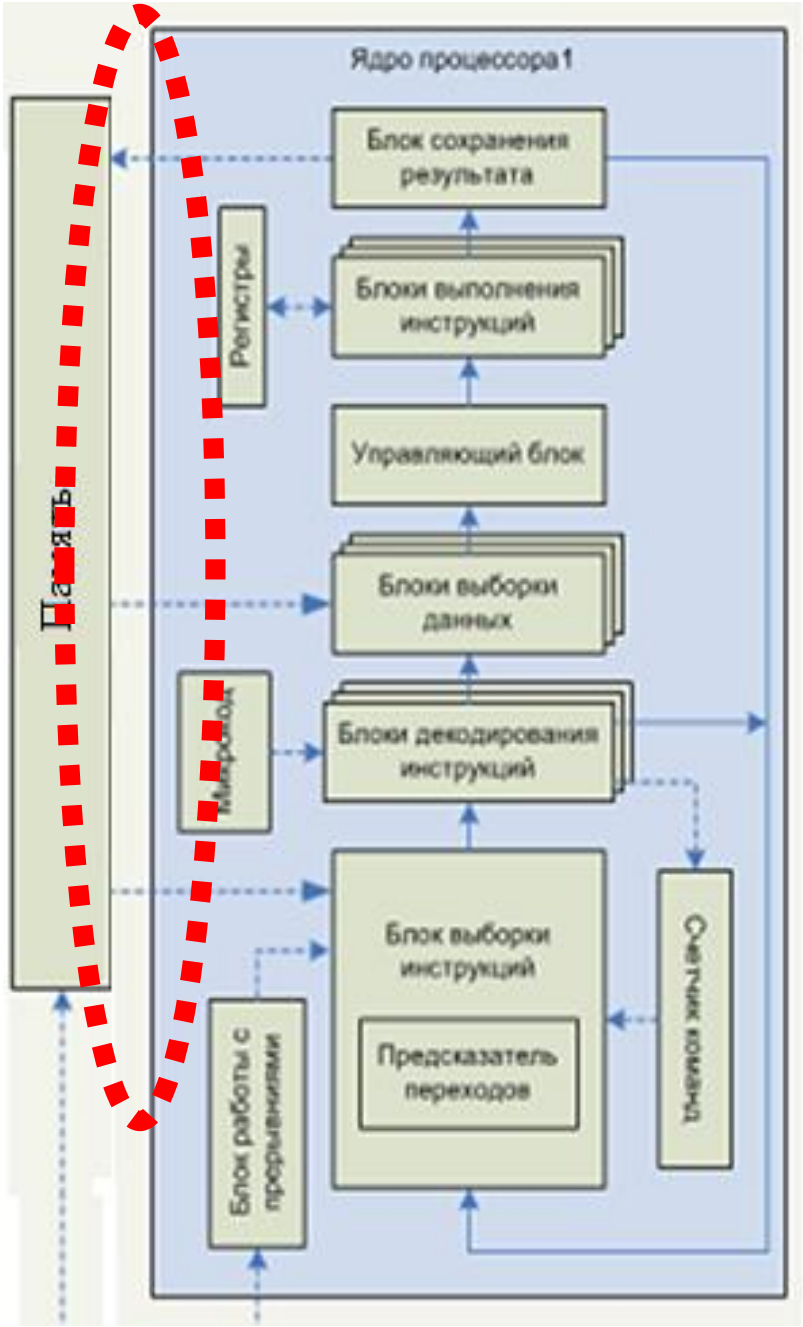
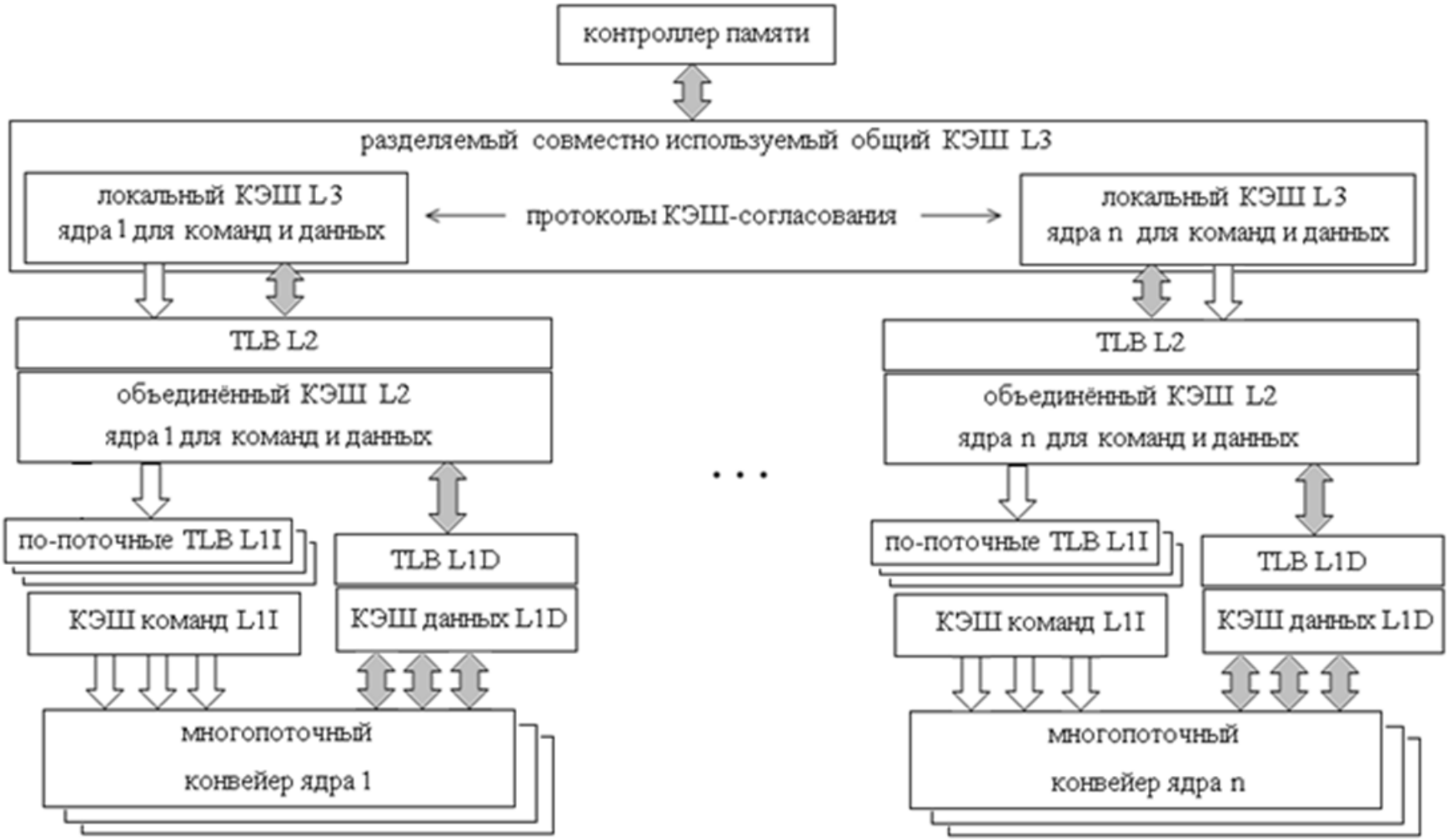
ПУТИ СОВЕРШЕНСТВОВАНИЯ МИКРОАРХИТЕКТУРЫ ЦП

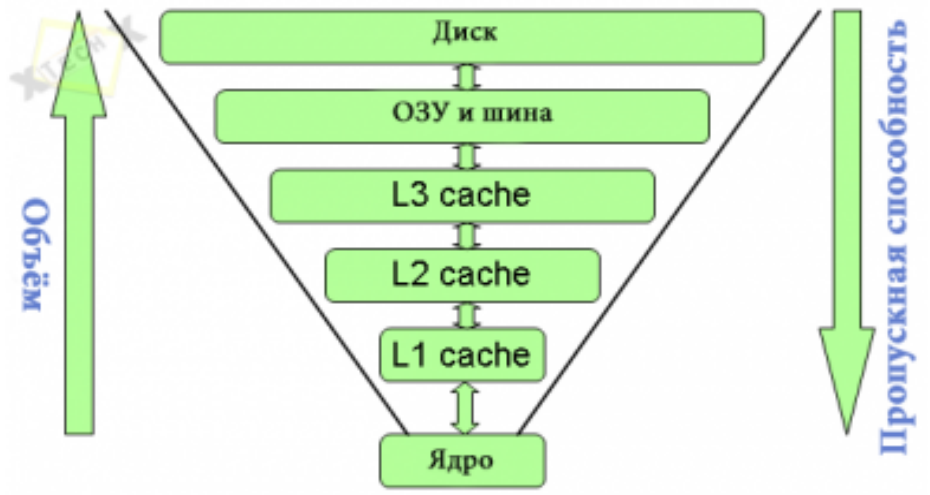


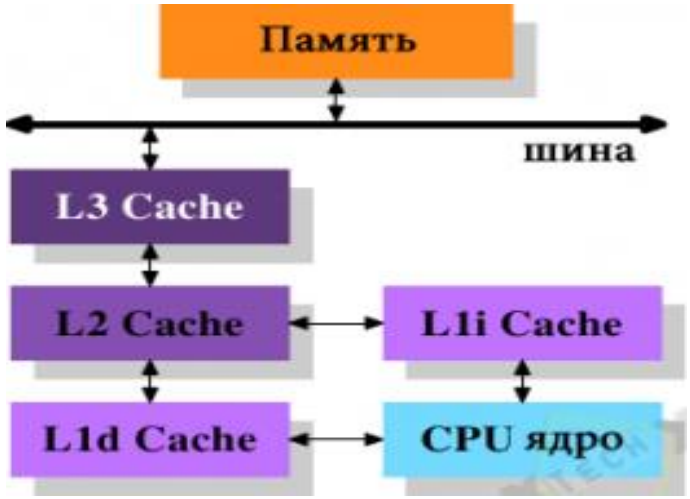
1. На этапе выборки инструкций из памяти
2. На этапе декодирования инструкций
3. **На этапе выборки операндов (данных) из памяти и сохранения результатов в память (на уровне взаимодействия ЦП и ОП)**
   - *Несколько уровней КЭШ,*
   - *КЭШ верхнего уровня L1 делится на КЭШ команд (L1i) и КЭШ данных (L1d),*
   - *использование выделенных и общих КЭШей,*
   - *повышение процента ассоциативности КЭШ (способы отображения строк в КЭШ-памяти),*
   - *использование буферов ассоциативной трансляции (TLB - Translation Lookaside Buffer),*
   - *использование механизмов КЭШ-согласования,*
   - *использование стратегии обратной записи (Write-Back) = отложенной записи*.
4. На уровне блоков управления исполнением инструкций
5. На этапе исполнения инструкций
6. Общий принцип совместного исполнения инструкций (конвейер команд).
7. Общий принцип совмещения операций (многофункциональная (суперскалярная) обработка, конвейер операций)
8. Использование интеллектуальных блоков и средств избежать простоев конвейера команд и конвейера операций
9. Общие принципы распараллеливания вычислений (многозадачность, многопоточность, многопроцессорность / многоядерность)

СПОСОБ ВЗАИМОДЕЙСТВИЯ ЦП И ОП (ЧЕРЕЗ МНОГОУРОВНЕВУЮ КЭШ)



СПОСОБ ВЗАИМОДЕЙСТВИЯ ЦП И ОП (ЧЕРЕЗ МНОГОУРОВНЕВУЮ КЭШ)





КЭШ 1-го уровня (L1I, L1D)

- наименьшая латентность (время доступа),
- наименьший размер, кроме того,
- часто делаются многопортовыми для одновременного выполнения операций записи и чтения, либо двух операций чтения за такт.
- разделяется на КЭШ-команд (L1I) и КЭШ-данных (L1D) для параллельного выбора из памяти команд и операндов.

КЭШ 2-го уровня (L2) = выделенный тыльный КЭШ

- обычно имеет значительно большую латентность доступа,
- значительно больше по размеру.
- отличается большей автономностью по сравнению с КЭШем LL
- обеспечивает настройку политик ядер в соответствии с различными рабочими наборами для своего ядра; служит подходящим блоком для управления ресурсами (например, отключением энергии для её экономии);
- значительно улучшает быстродействие, т.к. каждое ядро имеет собственный тыльный КЭШ L2 и совокупная полоса пропускания (из расчёта для всех ядер), значительно больше общего LL
- общее снижение времени задержки по отношению к системной шине.

КЭШ 3-го уровня (L3) = LL (last level) – КЭШ последнего уровня.

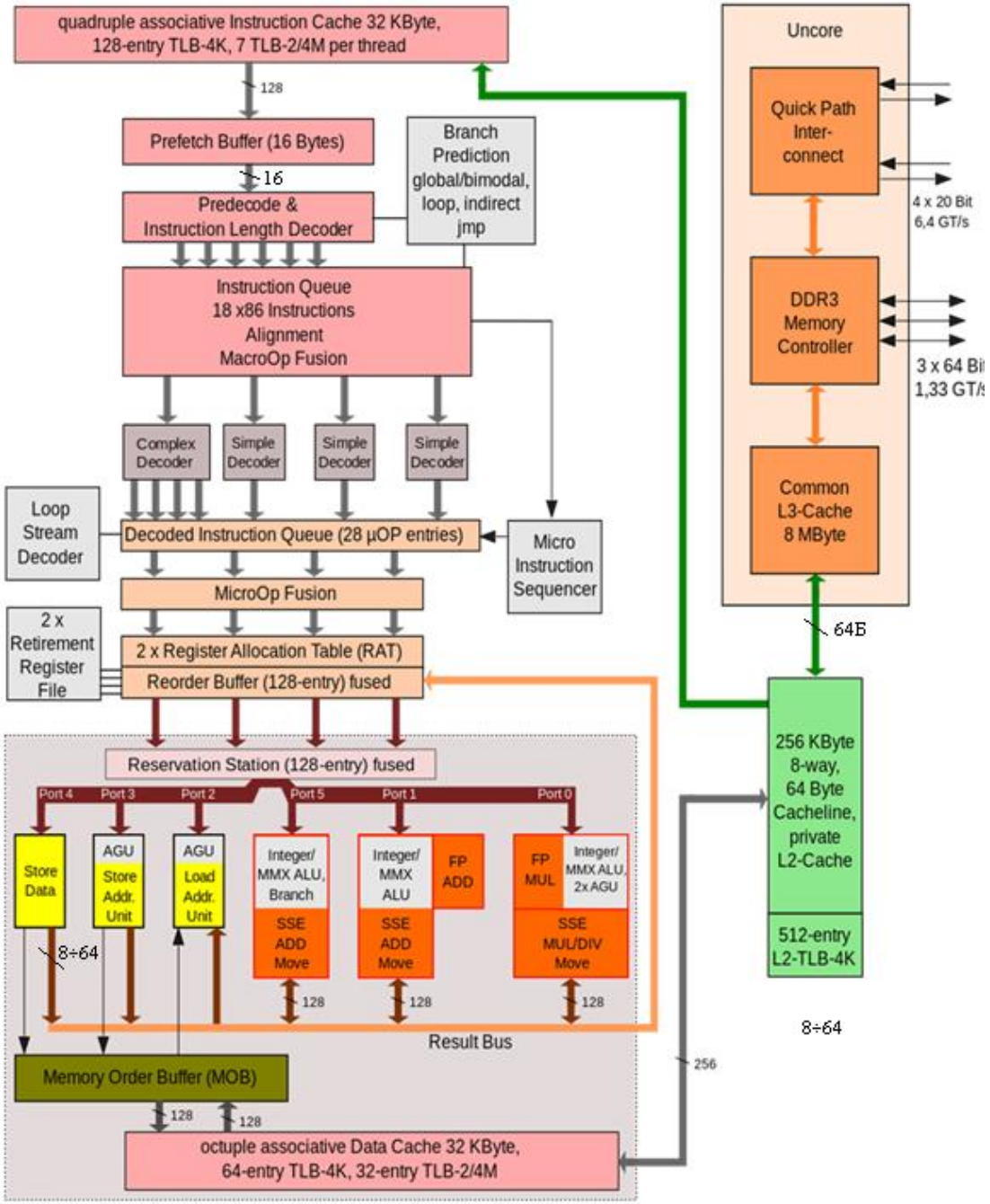
- самый большой по объёму (мультимегабайтный) увеличивает коэффициент попаданий до максимального значения,
- довольно медленный, но всё же он гораздо быстрее, чем оперативная память.
- используется для связи промежуточных процессов и работы на общих структурах данных.
- обеспечивает быстрый доступ к памяти при вводе-выводе и для блоков ускорителей.

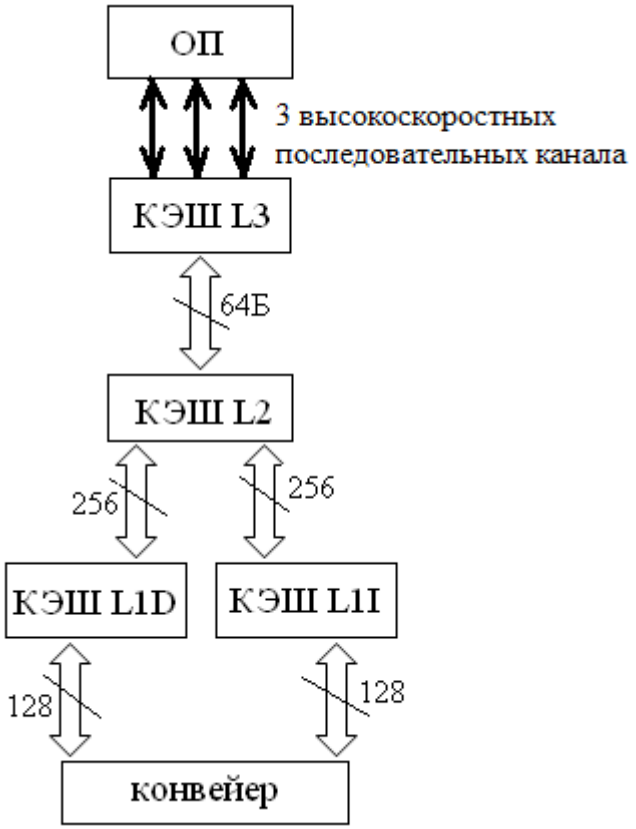
Типовая структура КЭШ-памяти

Основные функции КЭШ-памяти:

- согласование интерфейсов ЦП и ОП,
- обеспечение быстрого доступа к данным,
- упреждающая загрузка данных,
- отложенная запись данных.

Согласование интерфейсов

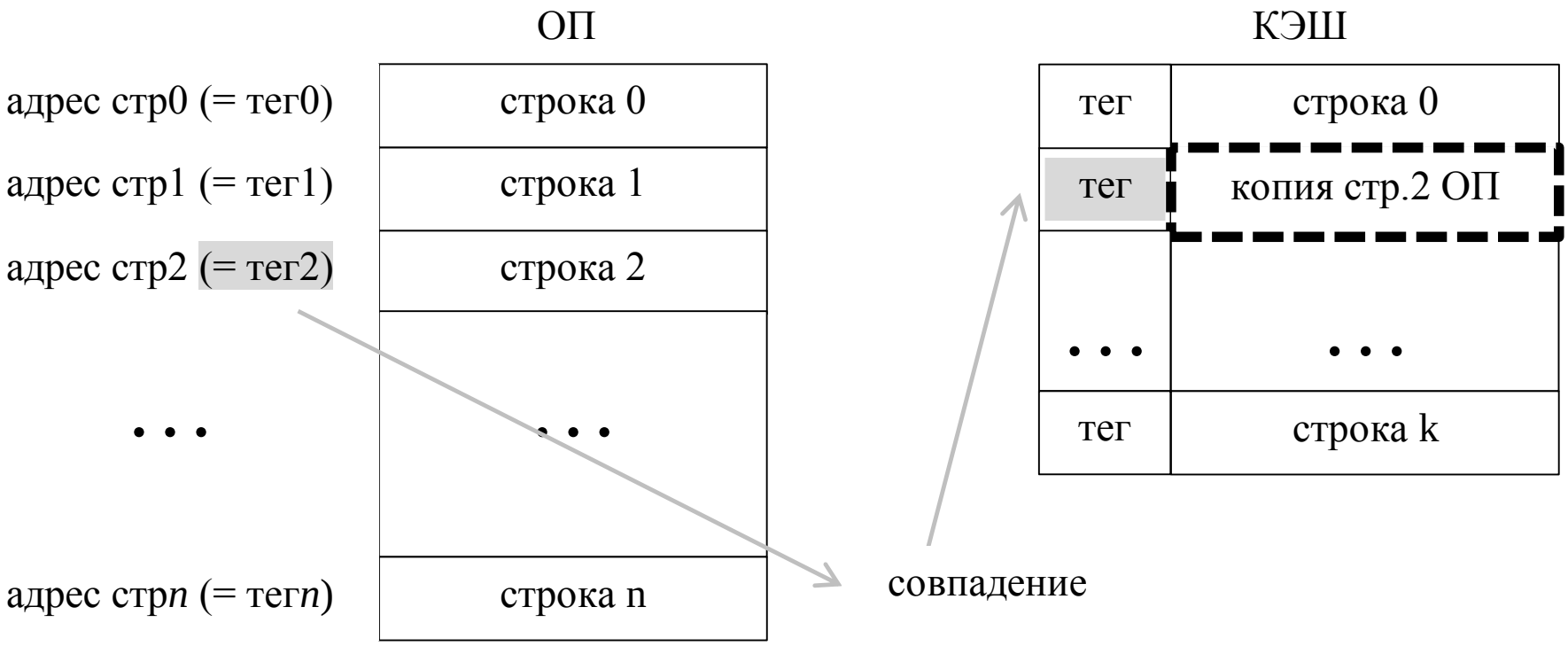




Обеспечение быстрого доступа к данным

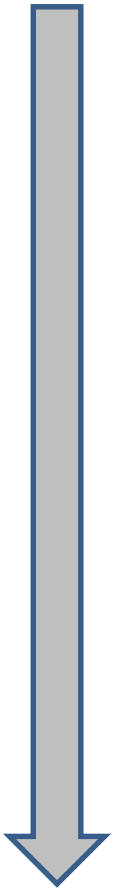
1. фактически медленные запросы к ОП подменяются «на лету» на более быстрые запросы к КЭШ-памяти уровня L1,
2. схемы КЭШ-памяти реализованы на более быстром ЗУ типа SRAM (по сравнению с DRAM),
3. использование ассоциативного принципа поиска информации вместо адресного (различают КЭШ разной степени ассоциативности – с различными принципами отображения строк),
4. использование специальных методов повышения эффективности КЭШ

Принцип поиска данных в КЭШ



степень ассоциативности КЭШ
(различные принципы отображения строк)

- прямое,
- n-канальное множественно-ассоциативное,
  (или наборно-ассоциативное)
- полностью ассоциативное.



Степень ассоциативности

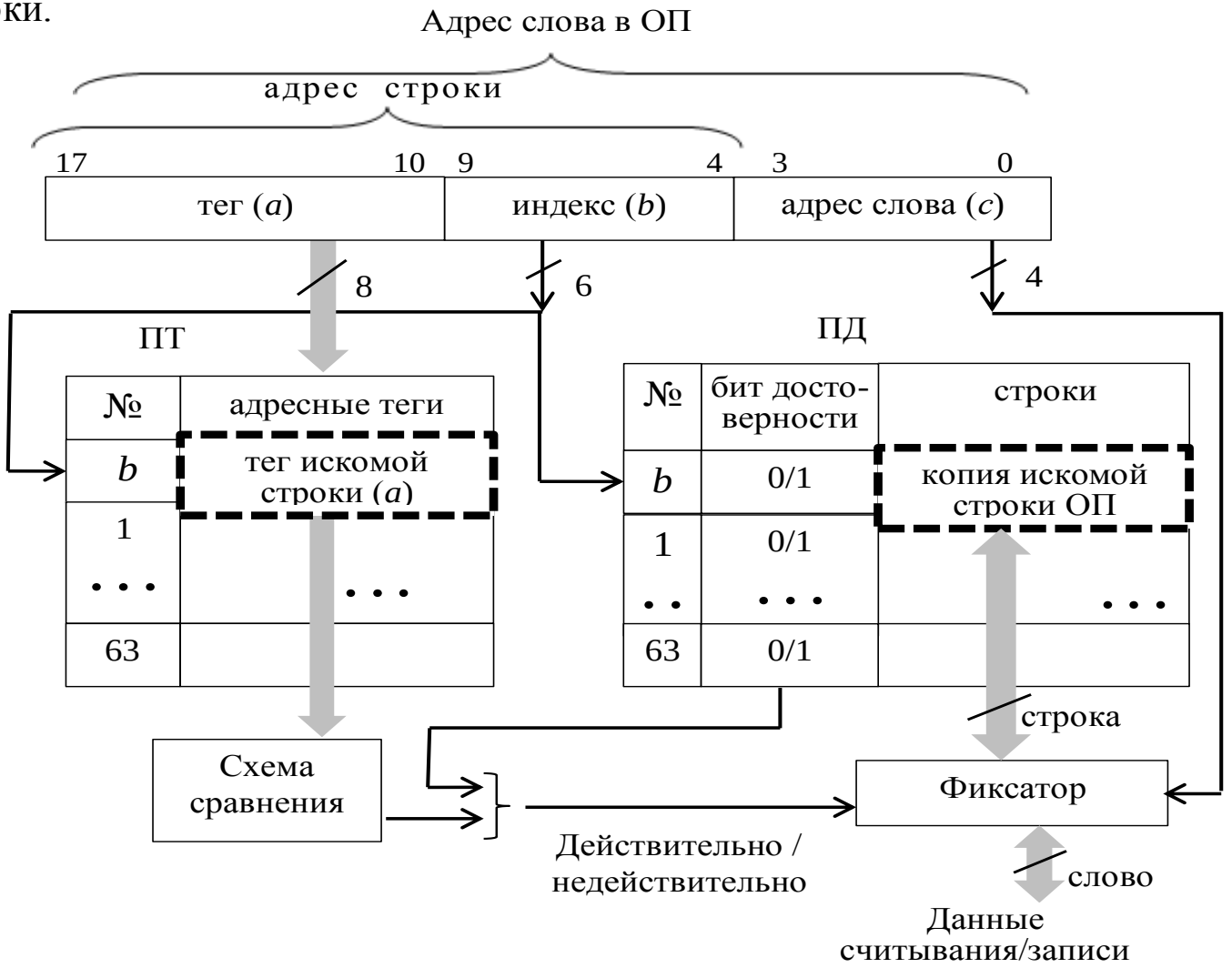
Прямое отображение

(степень ассоциативности = 0)

размер строки - 16 слов (2^4),

объём КЭШ - 64 строки (2^6) ,

объём ОП – 2^{14} строки.



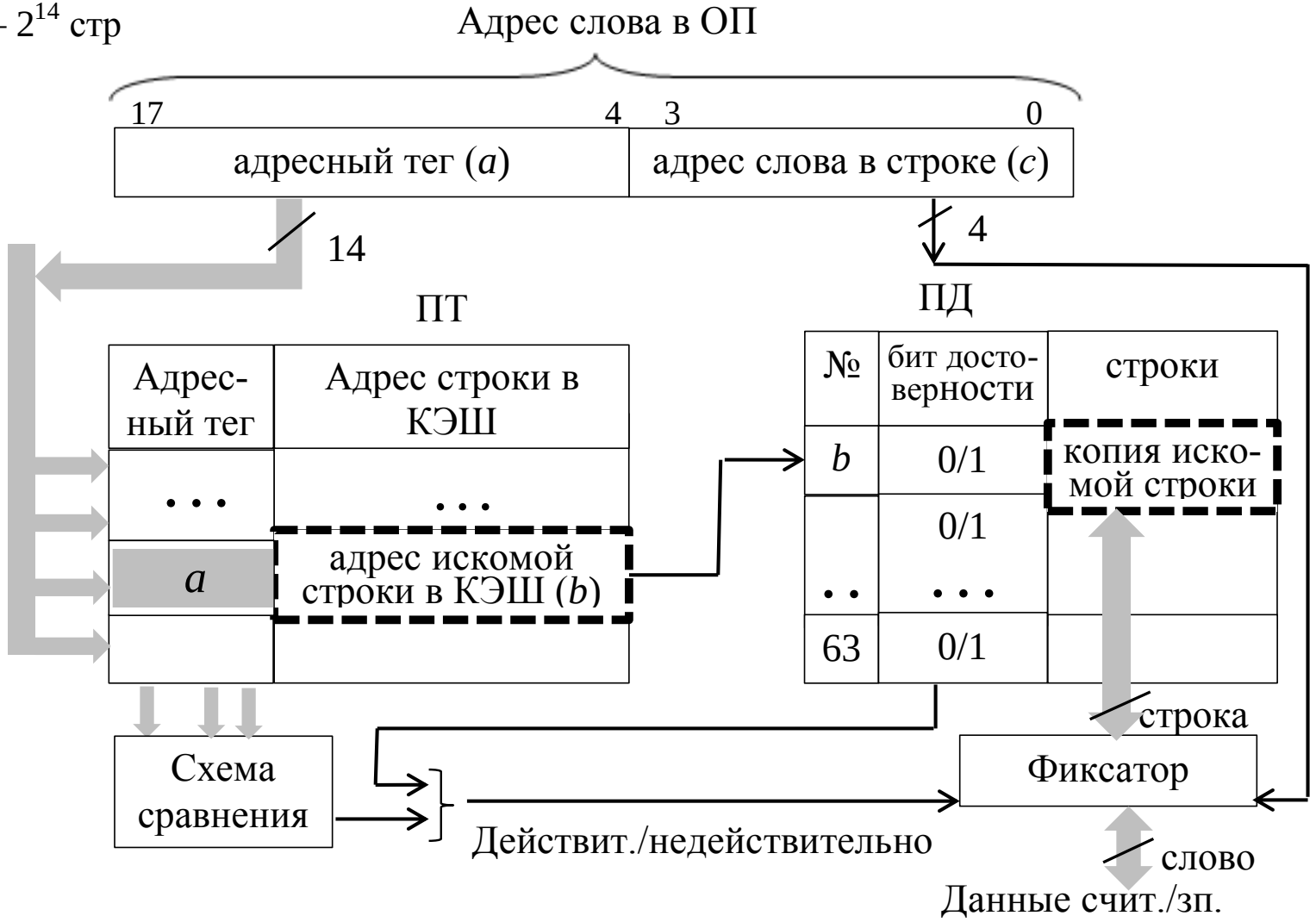
Полностью ассоциативное отображение

(степень ассоциативности = max)

размер строки - 16 слов (2^4),

объём КЭШ - 64 строки (2^6) ,

объём ОП – 2^{14} стр

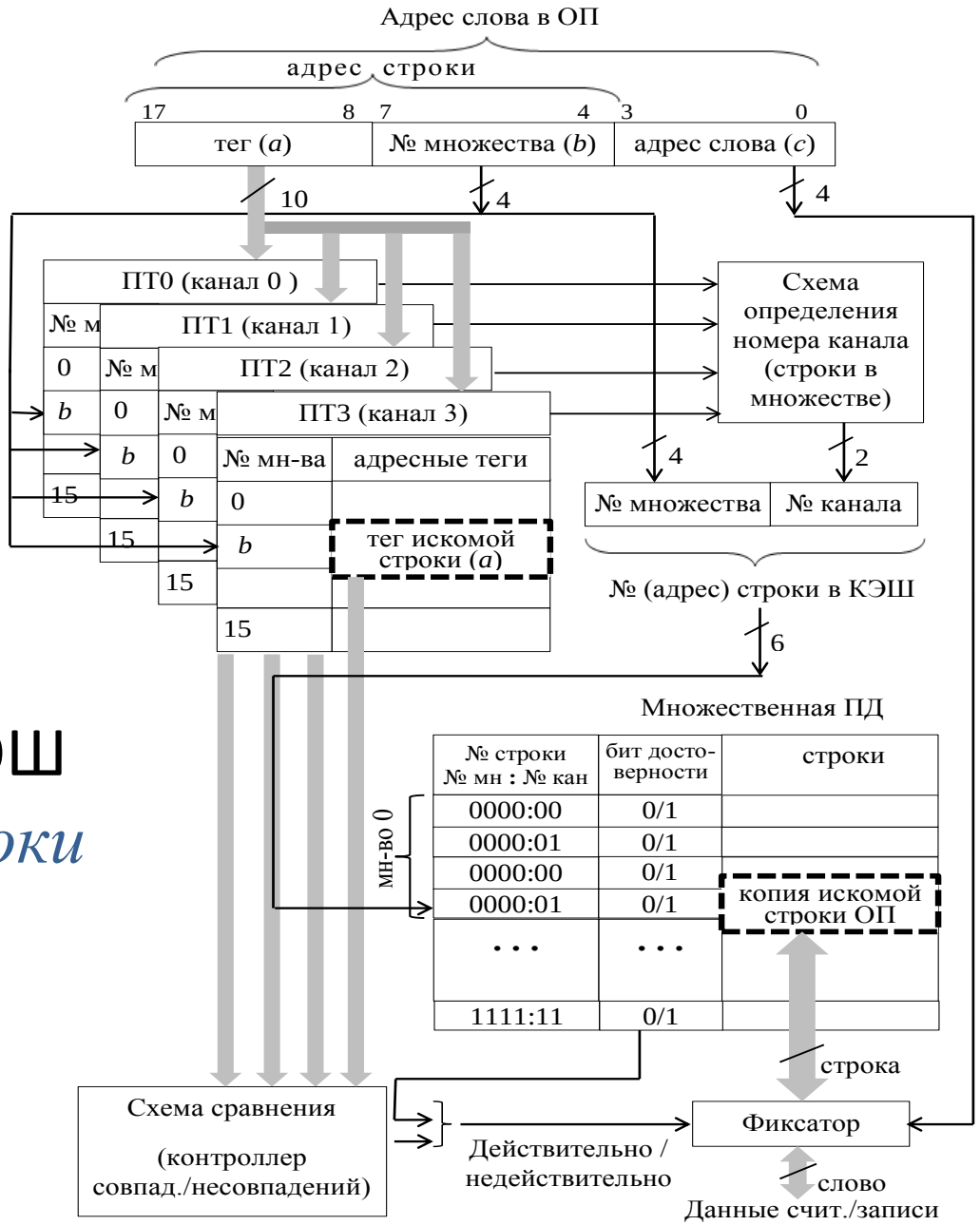


Наборно-ассоциативное отображение

размер строки - 16 слов (2^4),

объём КЭШ - 64 строки (2^6) ,

объём ОП – 2^{14} строки.

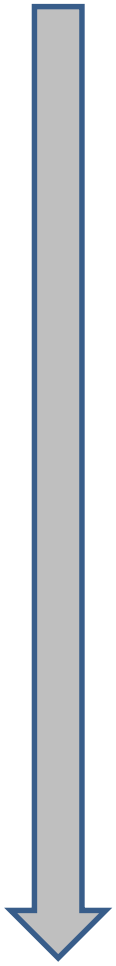


4-канальный (4-way) КЭШ

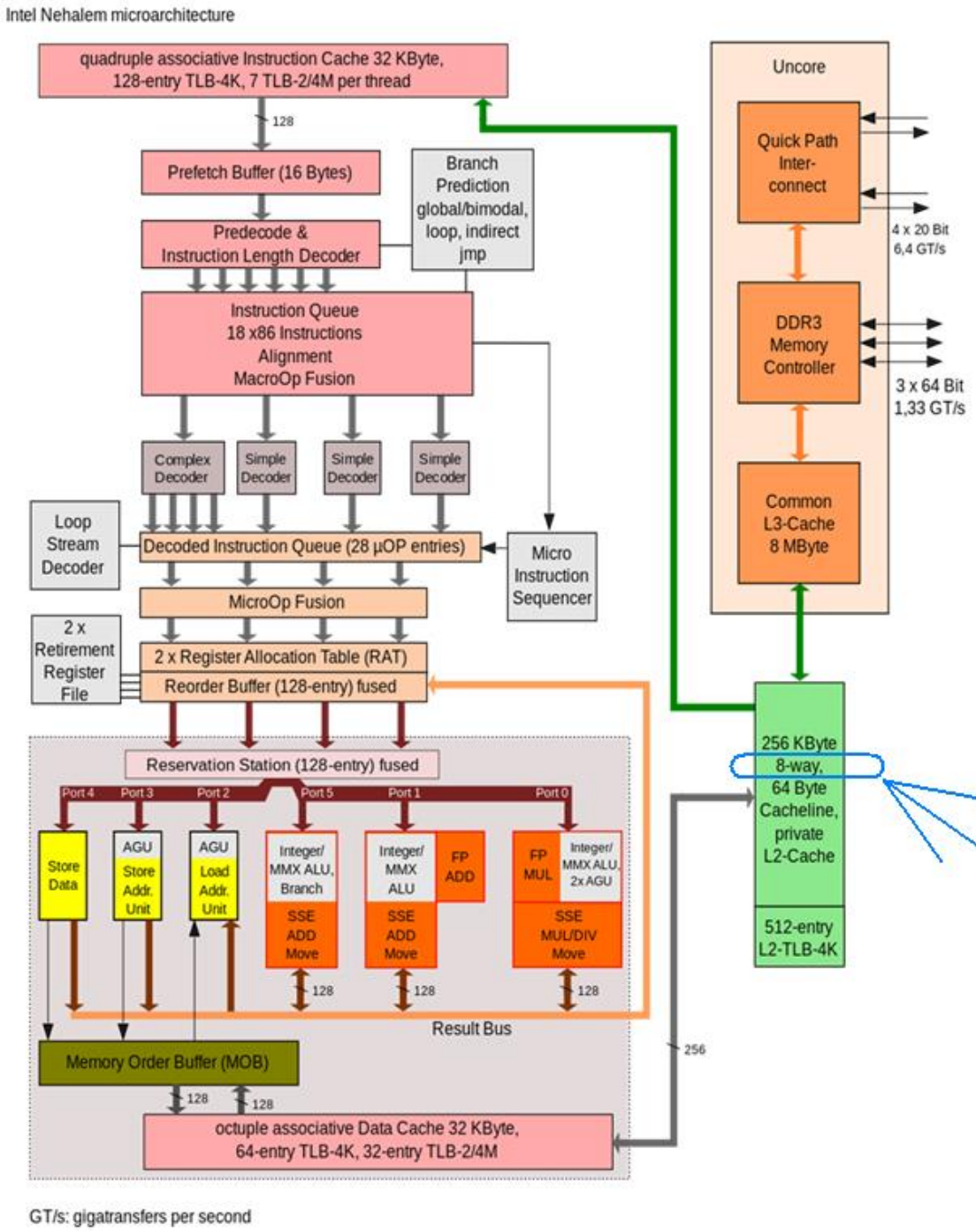
16 множеств по 4 строки

степень ассоциативности КЭШ
(различные принципы отображения строк)

- КЭШ прямого отображения,
- 2-канальный (2-way) КЭШ,
- 4-канальный (4-way) КЭШ,
- 8-канальный (8-way) КЭШ,
- 16-канальный (8-way) КЭШ,
- …
- полностью ассоциативный КЭШ.



Степень ассоциативности



Множественно-ассоциативное отображение

МЕТОДЫ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ КЭШ-ПАМЯТИ

1. Увеличить ёмкость КЭШ-памяти *(недостаток – увеличивается латентность=задержка при поиске данных)*.
2. Правильно предугадывать, что может понадобиться и размещать заранее именно нужные данные в КЭШ.

а) Использование блоков предсказания переходов (branch target buffer — BTB) и интеллектуальных алгоритмов предсказания (для команд ветвления: статических (всегда по одной и той же схеме) и динамических (с использованием таблицы истории переходов и блока предсказаний), отдельно для пар команд вызова/возврата из подпрограмм CALL/RET с использованием специального стека возвратов (return stack).

б) Стратегия помещения данных в КЭШ-память представляет собой алгоритм, определяющий: инклюзивная (предполагает дублирование информации, во всех уровнях локальных КЭШей (L1, L2, L3) и эксклюзивная (предполагает уникальность информации, находящейся в L1, L2 и L3).

3. Своевременное и безошибочное удаление менее важных строк из КЭШ для размещения более важных.

стратегии замещения строк (поиск наименее нужных данных):

*1. алгоритм **LFU** (Least Frequently Used) – выталкивается строка с наименьшим количеством обращений;*

*2. алгоритм **LRU** (Least Recently Used) – выталкивается давно не используемая строка;*

*3. алгоритм **MRU** (Most Recently Used) — выталкивается последняя использованная строка;*

*4. алгоритм **FIFO** (First Input First Output) — выталкивается строка, которая была загружена из ОП раньше всех,*

*5. алгоритм **RND** (randomize-алгоритм) – выталкивается случайно выбранная строка,*

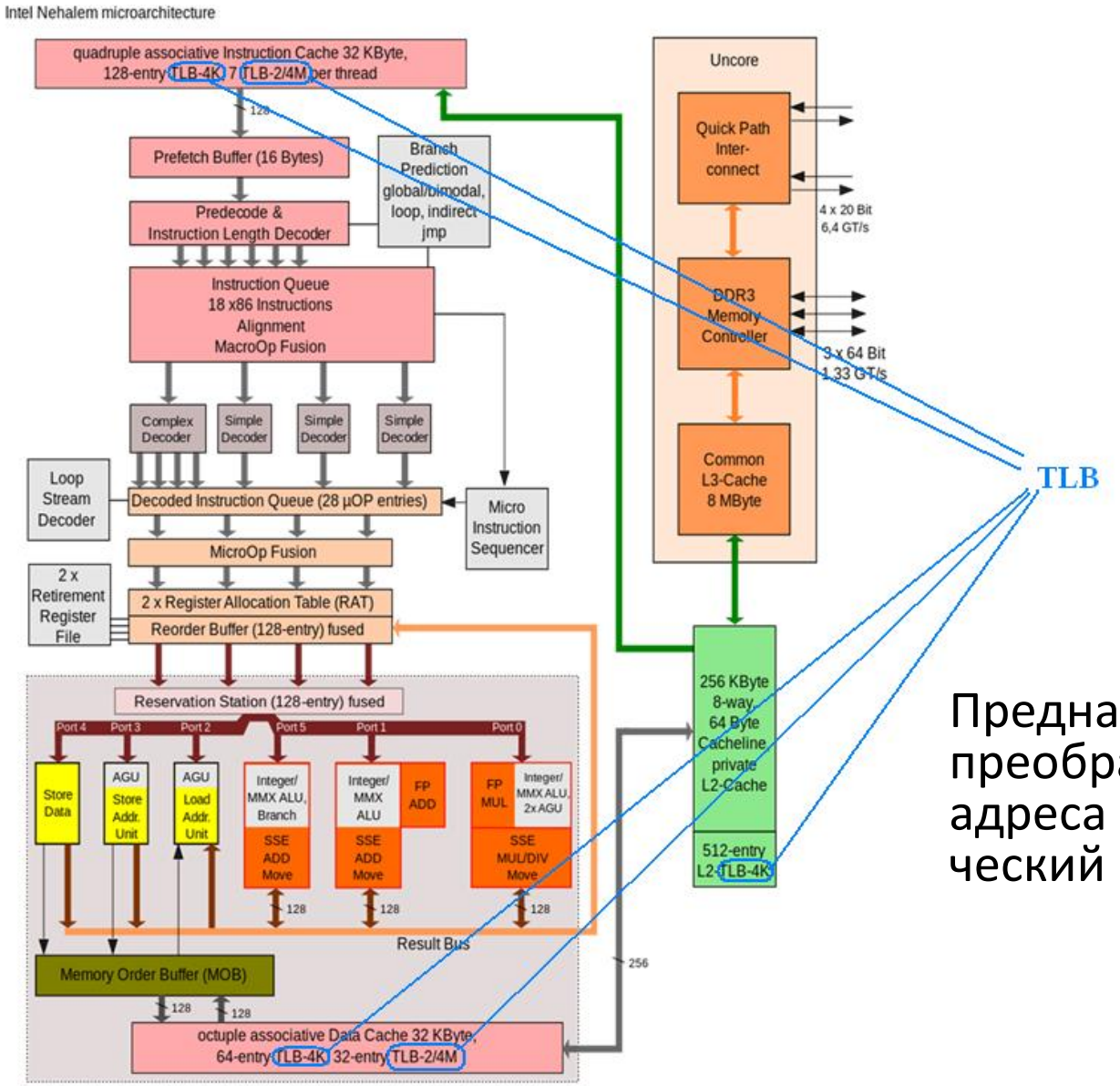
*6. алгоритм **ARC** (Adaptive Replacement Cache), комбинирующий LRU и LFU.*

4. Увеличение количества уровней КЭШ-памяти.

КЭШ L1, L2, L3 интегрированы на кристалл ЦП, (для серверов L4, выполнен в виде отдельного кристалла предметно-ориентированный – хранит адреса последних использованных web-страниц)

ИСПОЛЬЗОВАНИЕ БУФЕРОВ АССОЦИАТИВНОЙ ТРАНСЛЯЦИИ

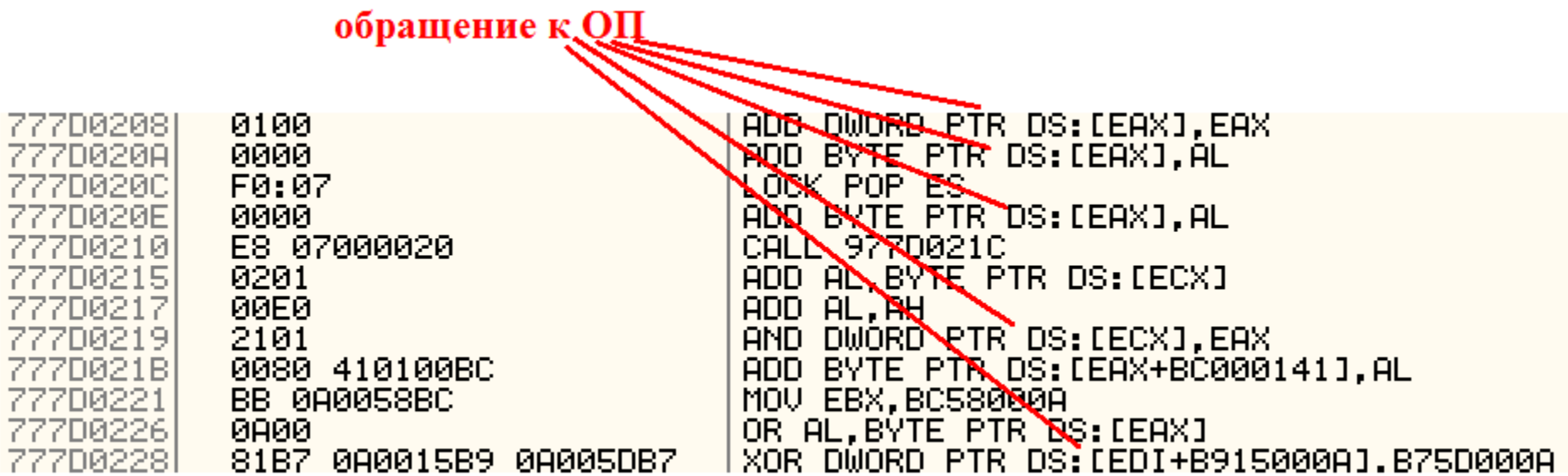
(TLB - *Translation lookaside buffer)*



Предназначены для быстрого преобразования логического адреса из программы в физический адрес (№ ячейки ОП)

Преобразование адресов ОП

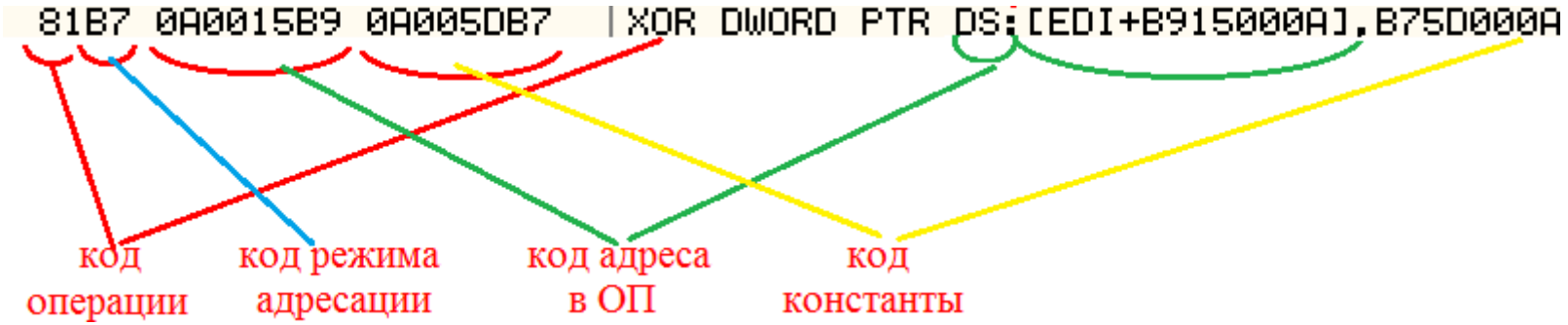
Программа: машинная команда 1
машинная команда 2
машинная команда 3
…
машинная команда M



Преобразование адресов ОП

машинная команда

Код операции	Код режима адресации	Код адреса операнда 1 в ОП	Код операнда 2 (константа в команде)

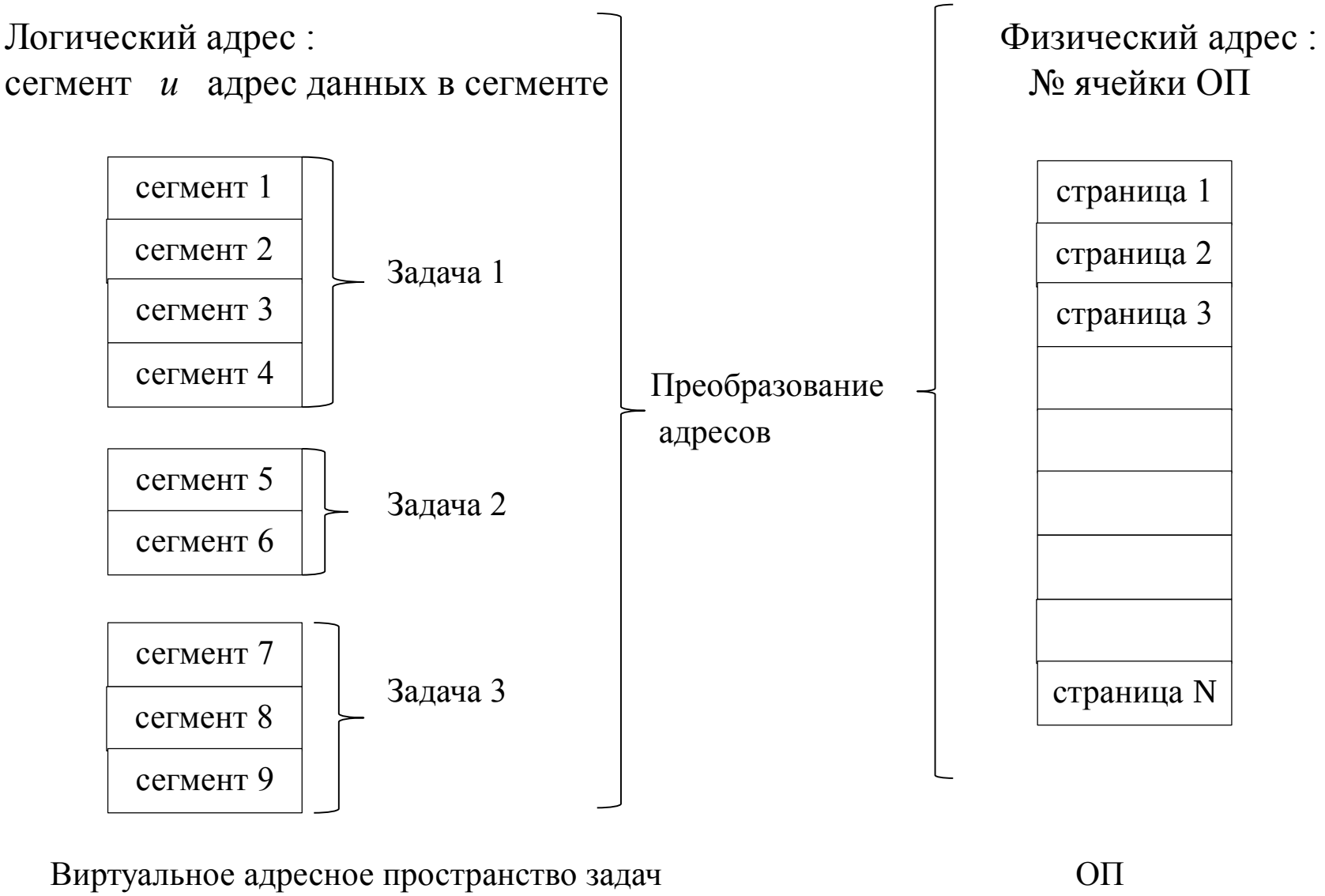


все адреса в программе –

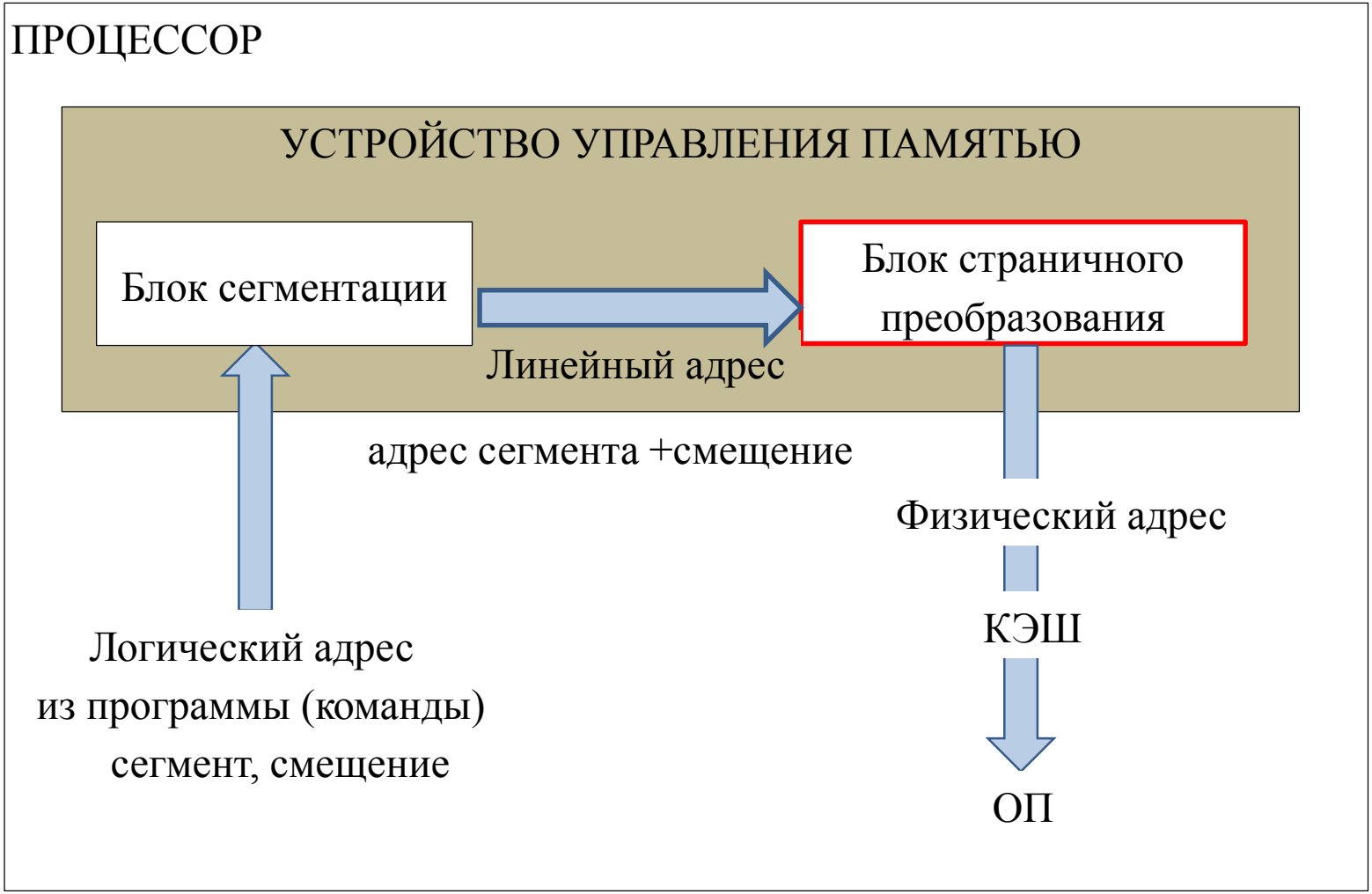
– логические DS:[EDI+B915000A]

сегмент адрес данных в сегменте

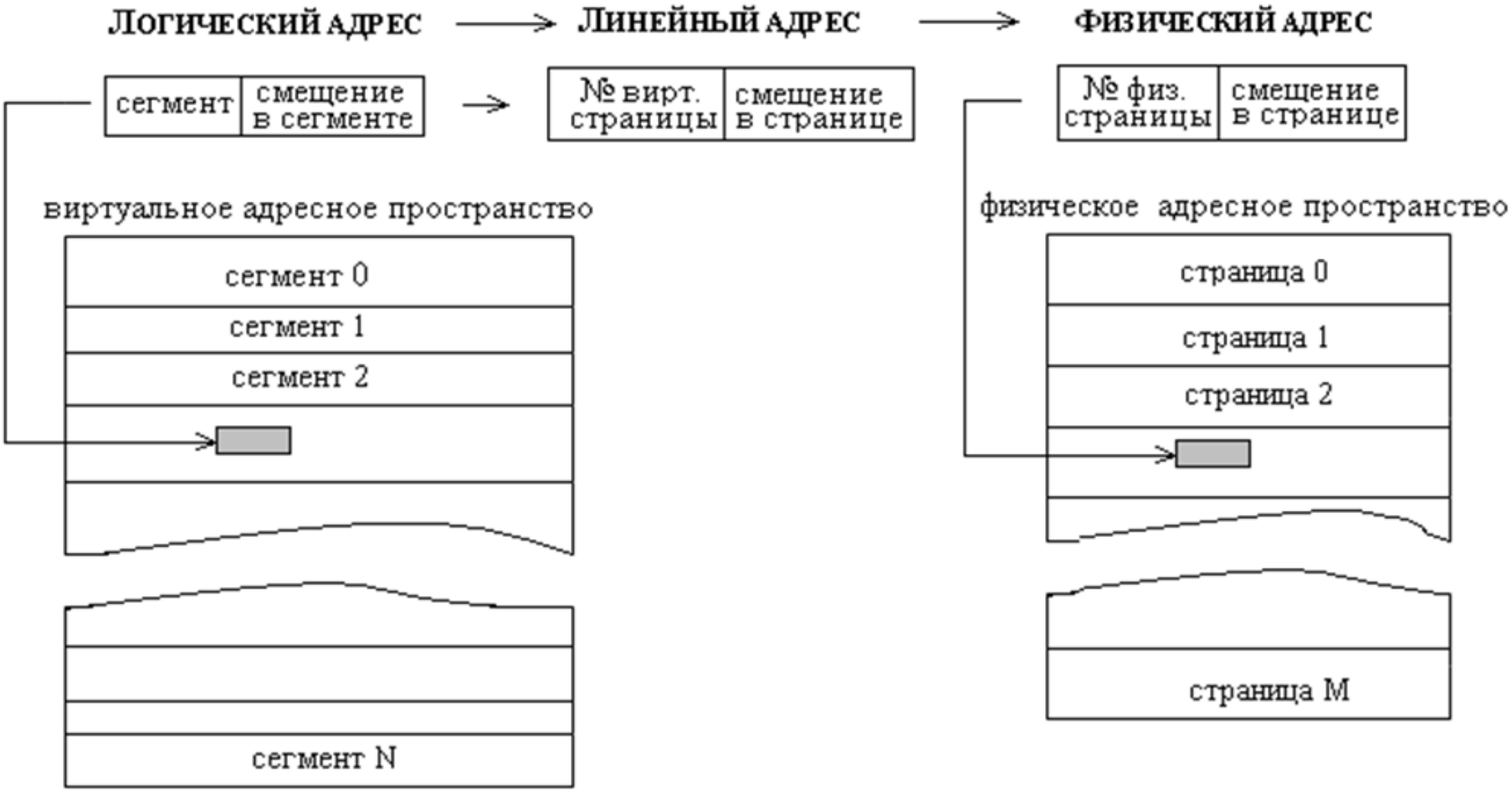
ПРЕОБРАЗОВАНИЕ АДРЕСОВ



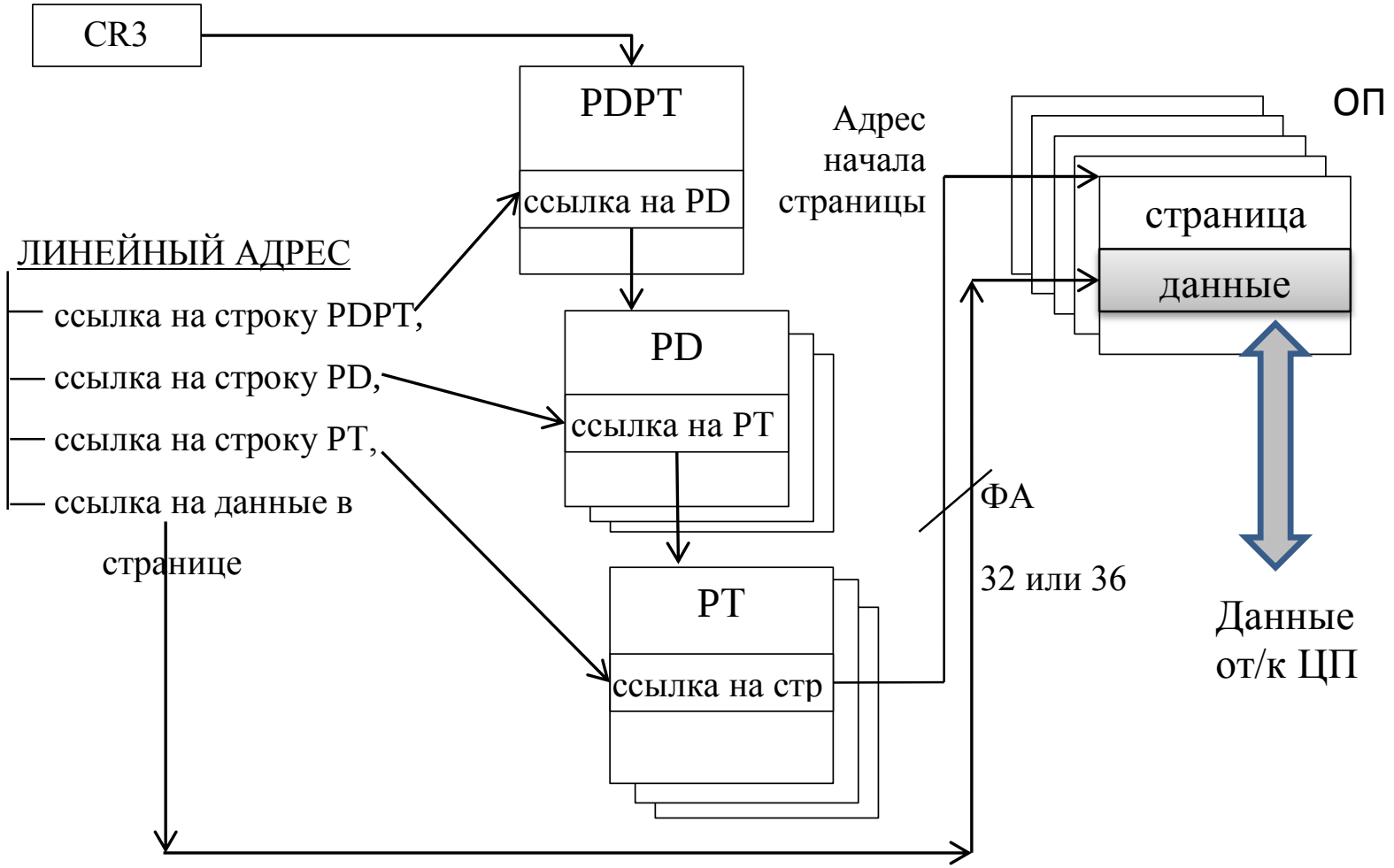
ПРЕОБРАЗОВАНИЕ АДРЕСОВ



ПРЕОБРАЗОВАНИЕ АДРЕСОВ



МЕХАНИЗМЫ СТРАНИЧНОЙ ПЕРЕАДРЕСАЦИИ



Многоэтапная трансляция линейного (виртуального адреса) в физический

Логический адрес

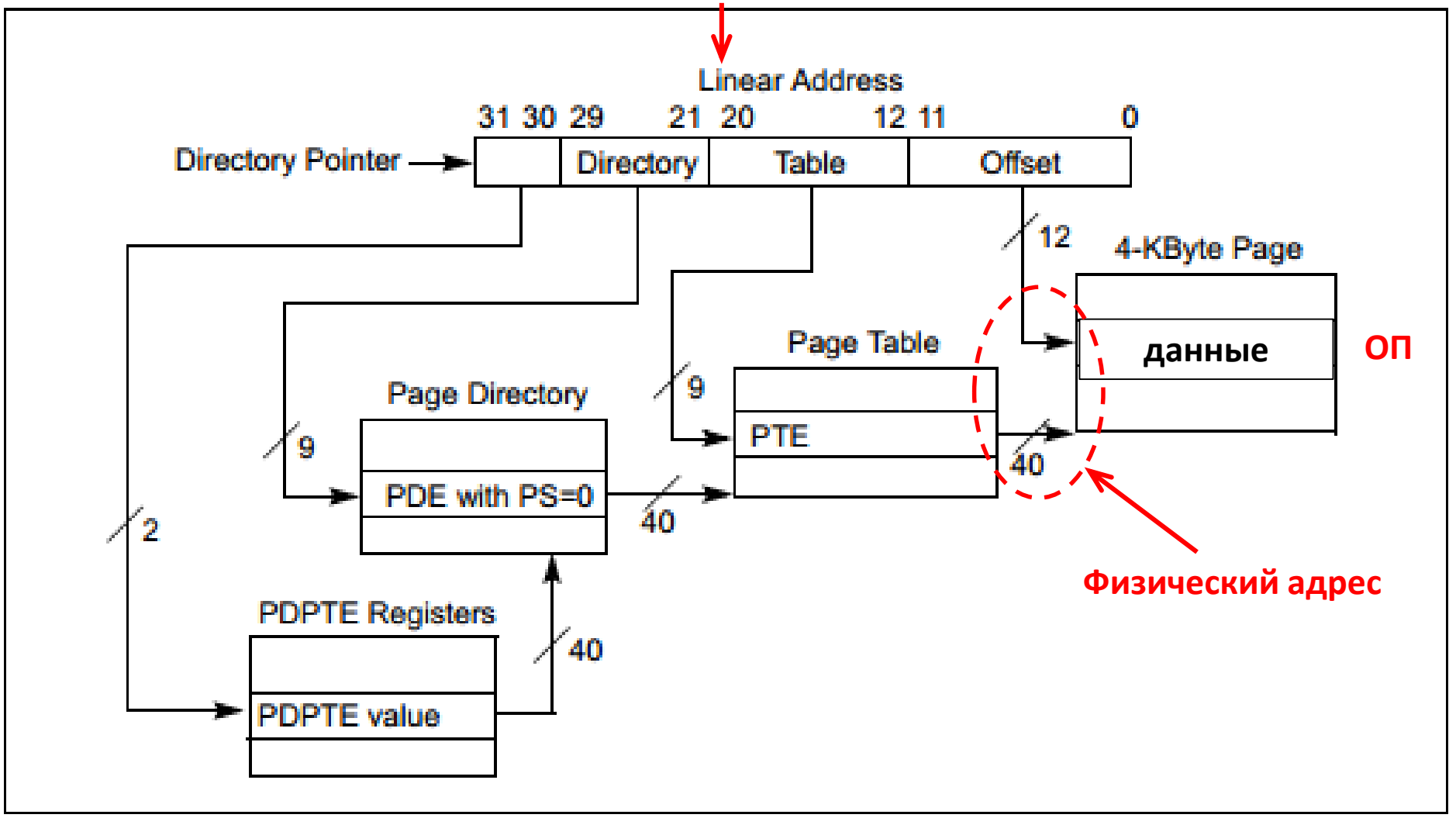
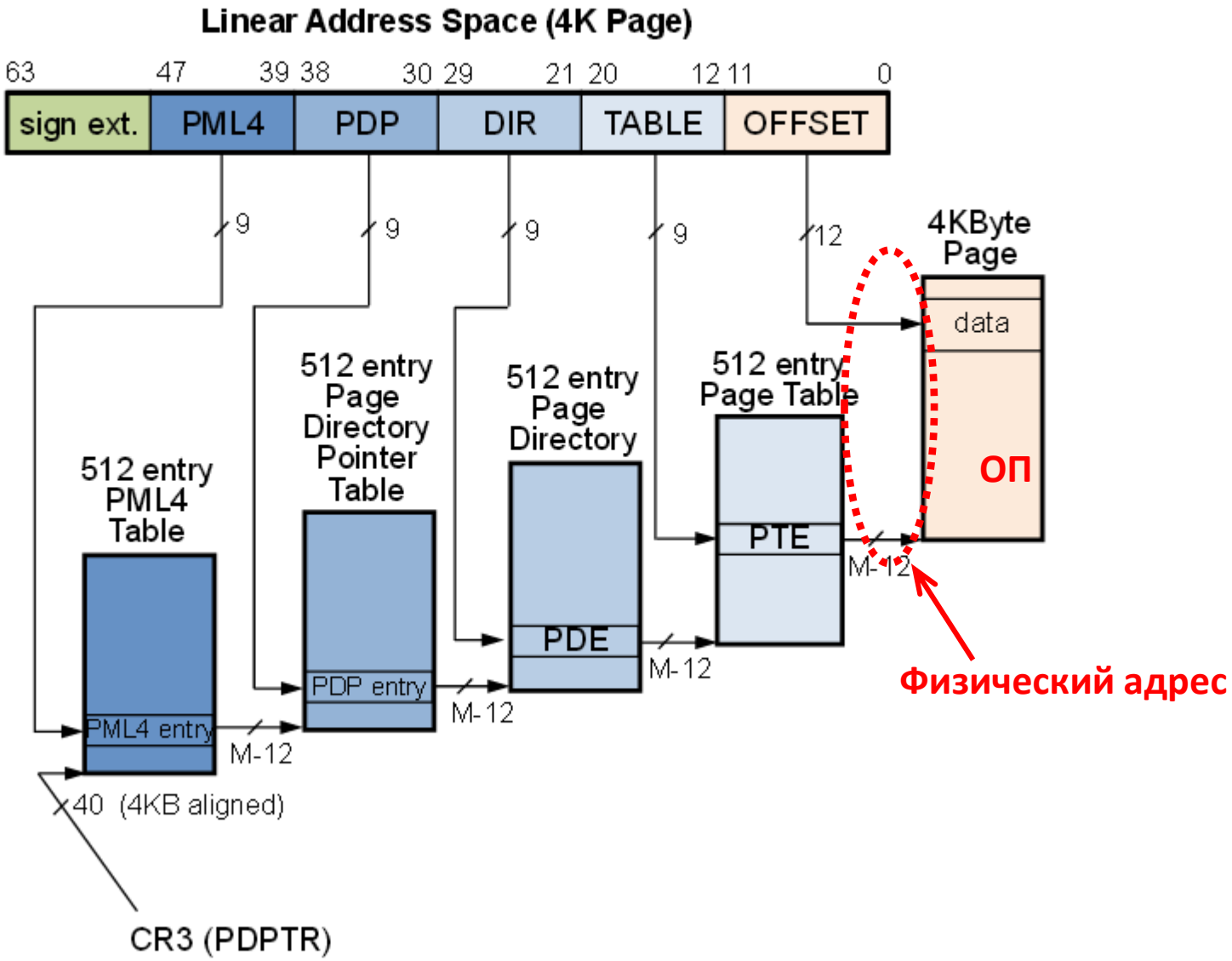


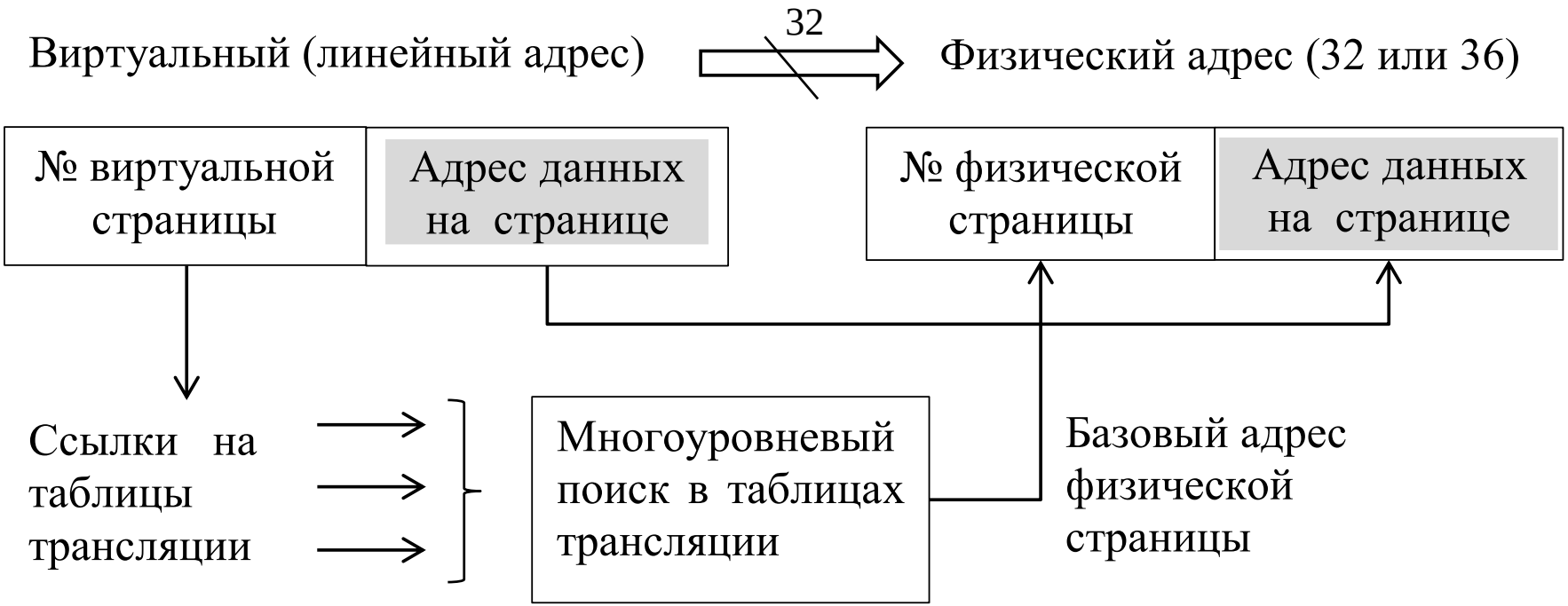
Figure 4-5. Linear-Address Translation to a 4-KByte Page using PAE Paging

http://moodle.technion.ac.il/pluginfile.php/313428/mod_resource/content/1/Core%20uArch%20-%20Advanced%20VLSI%20Arch.pdf

4KB Page Mapping in 64 bit Mode



Преобразование виртуального адреса в физический



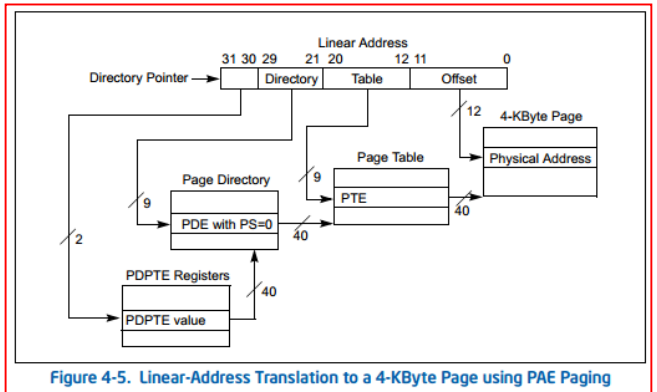
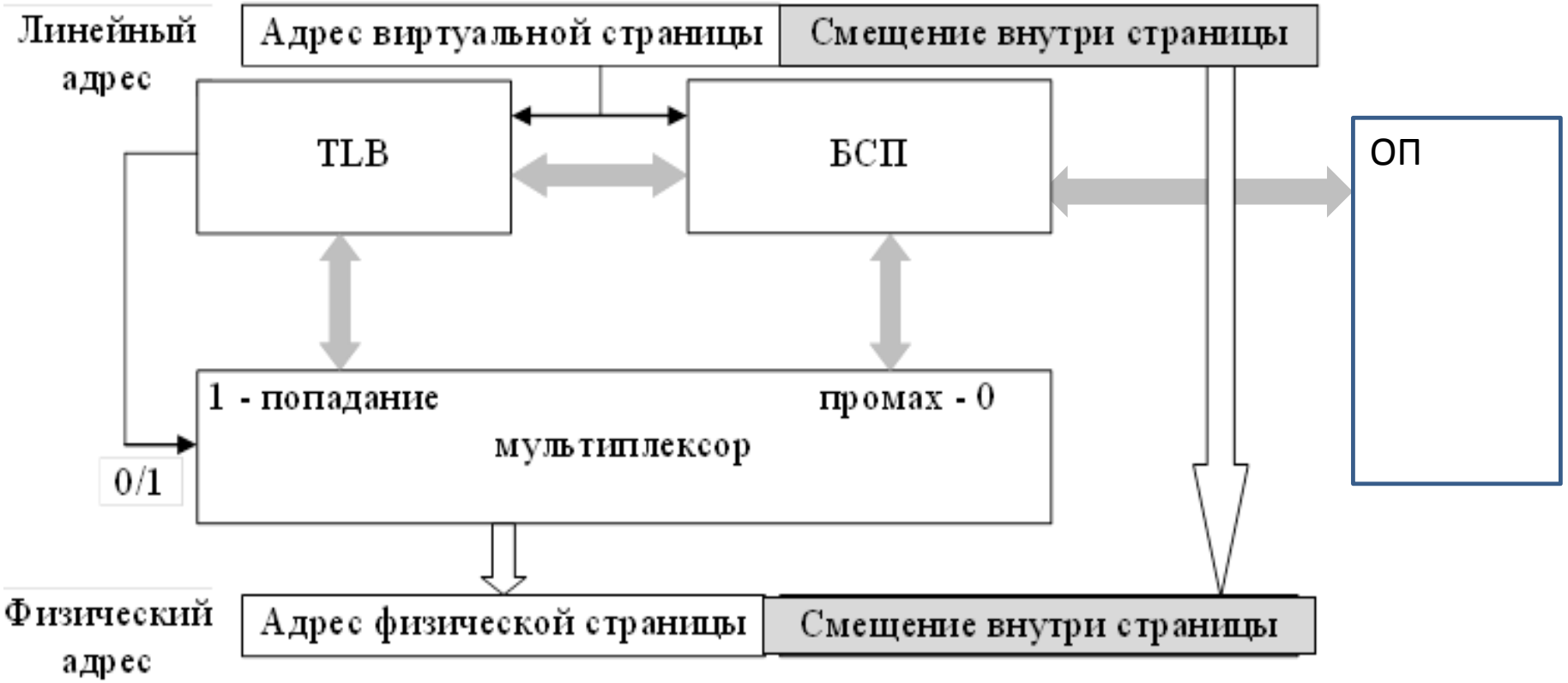


Figure 4-5. Linear-Address Translation to a 4-KByte Page using PAE Paging

Интернет-ресурс. Материалы с сайта фирмы Intel: Intel 64 and IA-32 Architectures Software Developer's Manual.

http://www.intel.ie/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf

Механизм формирования физического адреса с использованием TLB (КЭШ) или БСП (блока страничной переадресации)



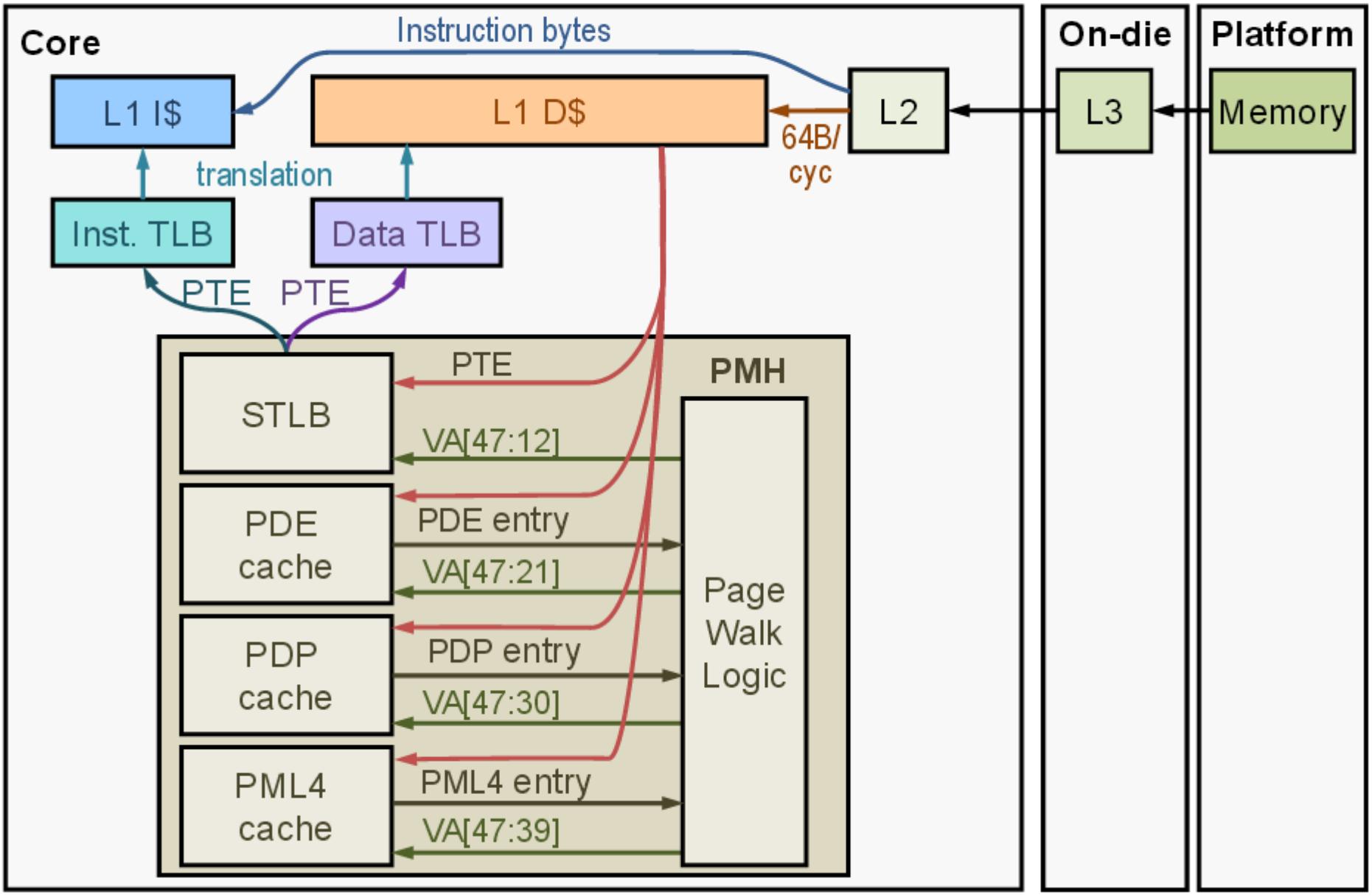
В TLB сохраняется строка последней таблицы трансляции, с непосредственным использованием которой вычисляется физический адрес.

TLB

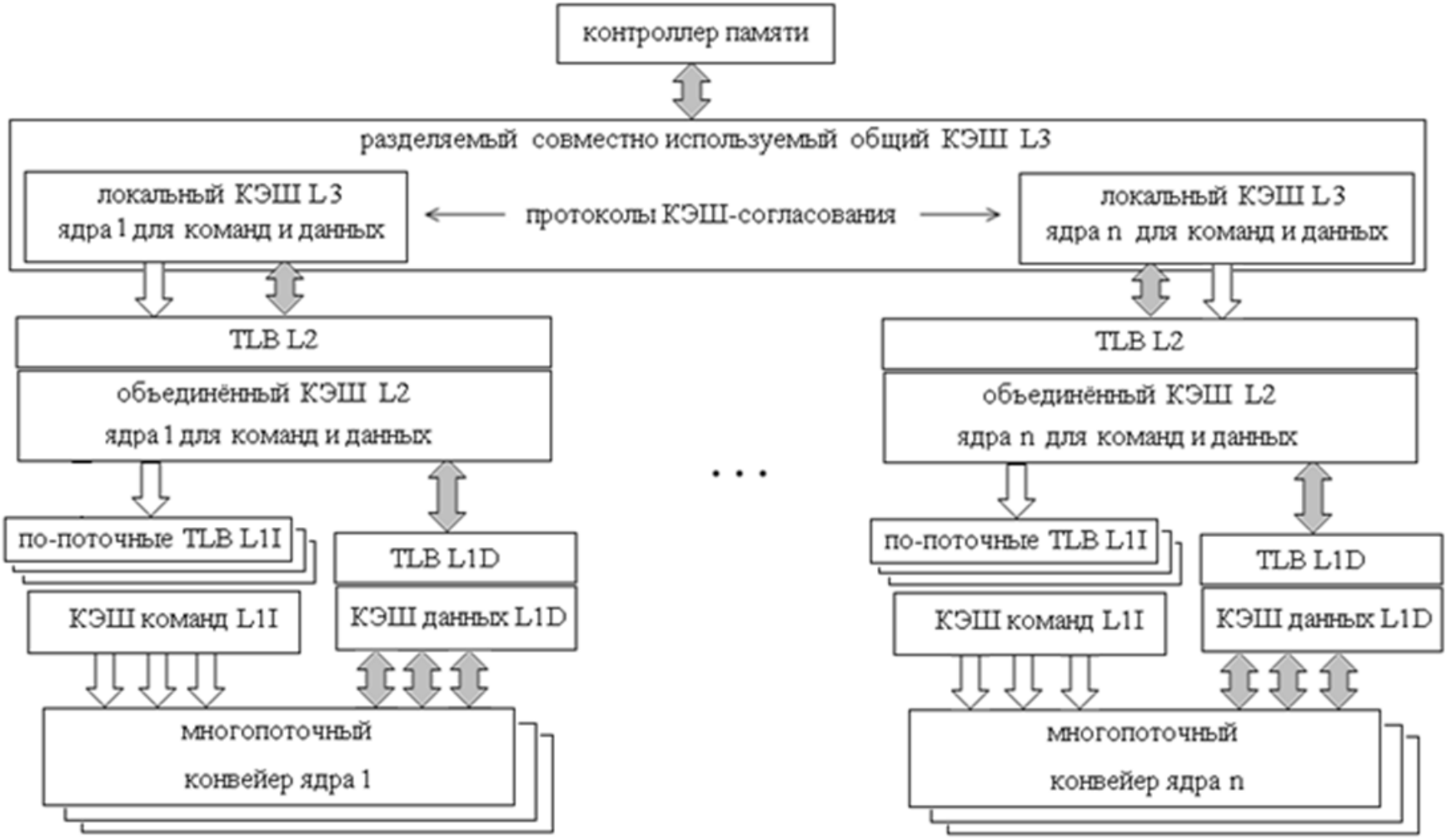
Память тегов	Память данных
адрес вирт. страницы	адрес физ. страницы

http://moodle.technion.ac.il/pluginfile.php/313428/mod_resource/content/1/Core%20uArch%20-%20Advanced%20VLSI%20Arch.pdf

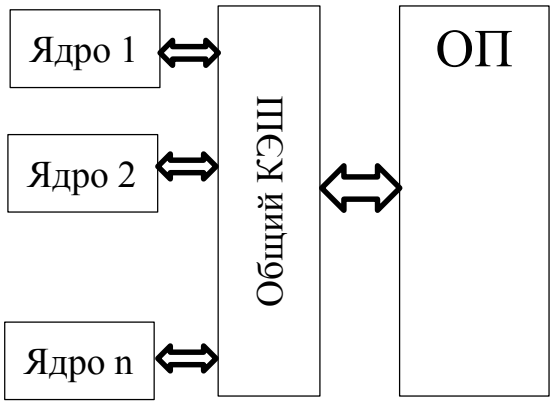
Cache and Translation Structures



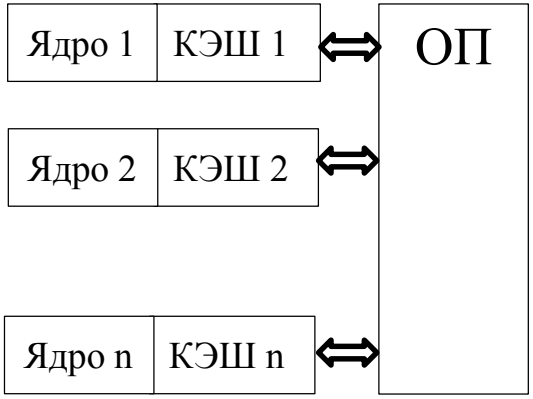
Организация КЭШ-памяти последнего уровня иерархии LLC (last level cache)



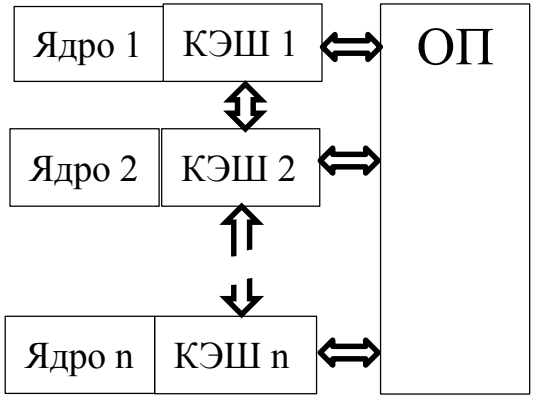
Организация КЭШ-памяти последнего уровня иерархии LLC (last level cache)



А) без поддержания когерентности



Б) MSI,MESI – протоколы

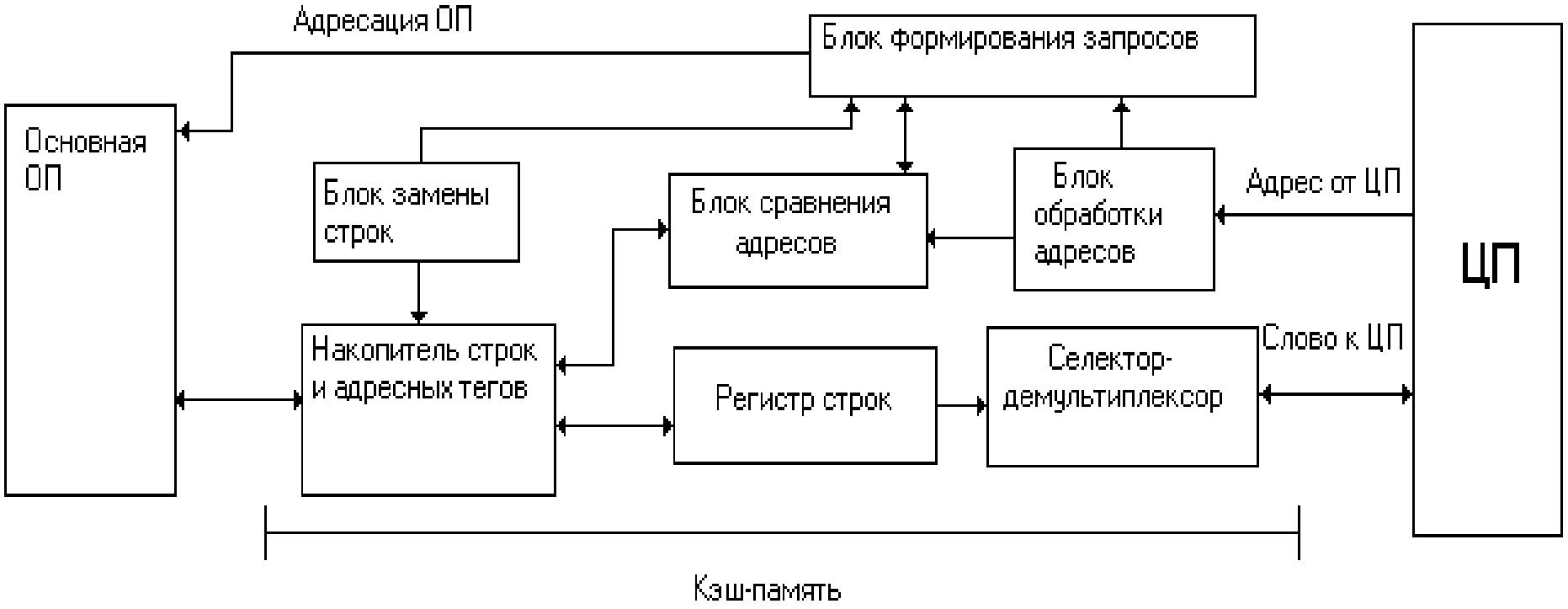


В) MOSI,MOESI, MESIF- протоколы

Общий КЭШ LL: очень большой по объёму = очень дорогой (из-за сложности организации поиска данных в КЭШ - ассоциативной памяти) = медленный (за счёт увеличения латентности=задержки при поиске нужной копии данных).

Раздельный КЭШ LL: несколько маленьких быстрых дешёвых КЭШ.
НО возникает проблема согласования нескольких копий одних и тех же данных - **поддержания когерентности памяти.**

Типовая структура ассоциативной памяти



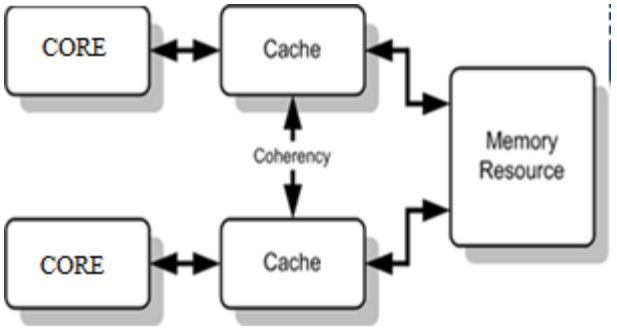
Латентность ассоциативной памяти увеличивается *при увеличении объёма, т.к. происходит увеличение количества сравнений адресов искомых данных с адресами имеющихся копий.*

Для сравнения адресов используются ***схемы побитного сравнения*** *каждого разряда искомого адреса с каждым разрядом хранимых адресных тегов → усложнение схемы ассоциативной памяти →* ***удорожание.***

ПОДДЕРЖАНИЯ КОГЕРЕНТНОСТИ КЭШЕЙ

memory disambiguation[INTEL] (разрешение проблемы неоднозначности)

Под «протоколом поддержания когерентности КЭШей» следует понимать согласованное изменение всех существующих копий данных, для того, чтобы при запросе на доступ к любой копии любой процесс/процессор получал последние внесённые в эти данные (в оригинал или любую из копий) изменения.



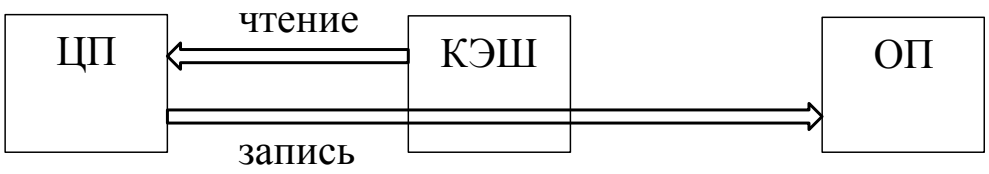
уровень 1. Каждое изменение (операция записи в копию) мгновенно становиться доступна всем; **уровень 2.** Все процессоры видят одинаковую последовательность изменений значений для каждой отдельной копии (возможно с небольшими задержками);	строгая когерентность
уровень 3. Допускают недолгое существование некоторой рассогласованности копий при отсутствии неотложной надобности.	нестрогая когерентность
уровень 4. Разные процессоры могут выполнять разные операции и видеть различные последовательности значений.	отсутствие когерентности

Методы решения проблемы :

- механизм сквозной записи (Write-Through) читай (write through cache);
- MSI-протокол;
- MESI-протокол;
- MOSI-протокол;
- MOESI-протокол;
- MESIF-протокол.

Механизм сквозной записи Write-Through

любые изменения сразу же записываются не только в КЭШ, но и в ОП; остальные процессоры сразу информируются об этом.



Достоинство. Простейший способ поддержания когерентности.

При использовании архитектуры общей шины другие процессоры просто «подслушивают» (snooping-сторожить), что текущий «владелец» шины по ней пересылает и, зарегистрировав, что кто-то выполняет операцию записи в память, обновляют «свои» записи в КЭШе.

При использовании более сложной коммутационной сети требуется ведение каталога, где соответствующие строки в КЭШах помечаются как «достоверные» (valid) или «недостоверные» (invalid); при обращении к «недостоверным» данным необходимо брать копии не из КЭШа, а из ОП.

Недостаток.

Малая эффективность (при каждом изменении требуется обновление локальных копий).

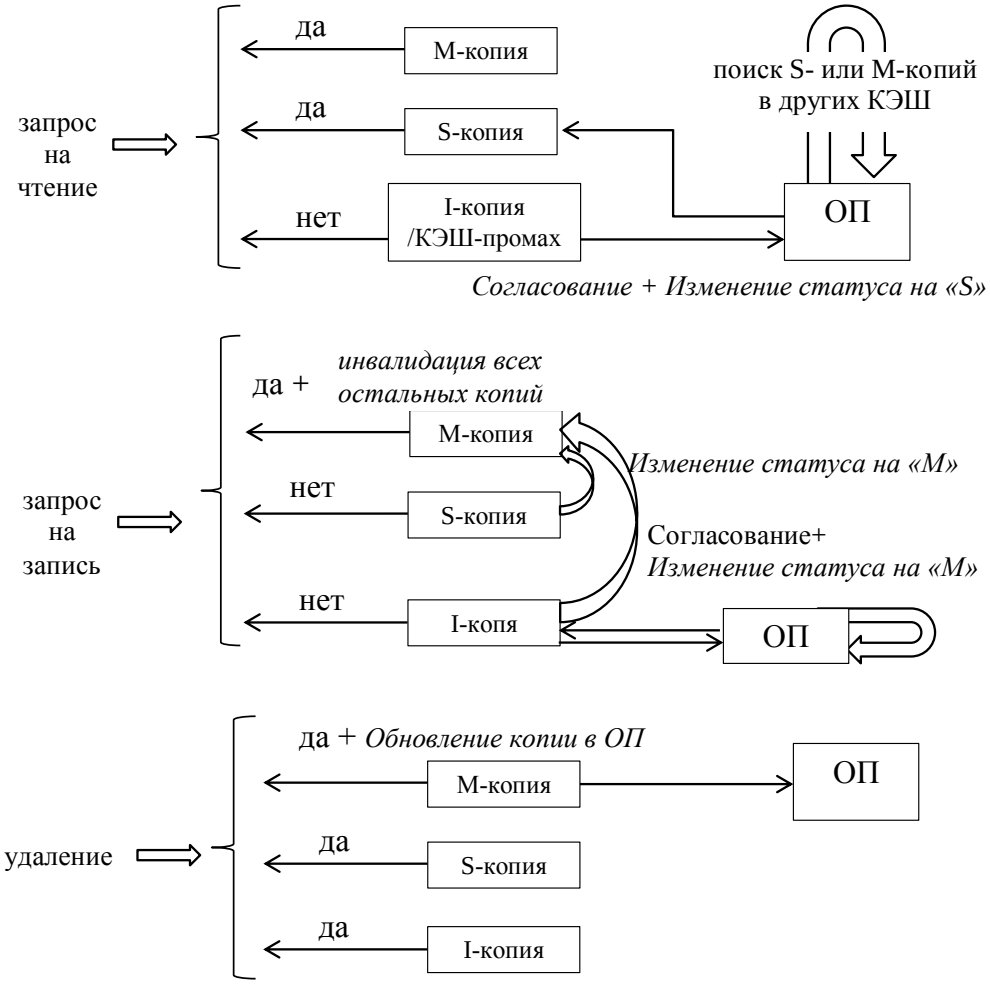
MSI-протокол

	M	S	I
M	✗	✗	✓
S	✗	✓	✓
I	✓	✓	✓

Modified *- блок отличается от ОП, но достоверен. При вытеснении такого блока из КЭШ его оригинал в ОП заменяется на изменённую копию.*

Shared *- этот блок совпадает с оригиналом из ОП и существует, по крайней мере, в одной КЭШ-памяти. Его удаление из КЭШ не повлечёт никаких изменений.*

Invalid *- этот блок является недействительным и должен быть повторно считан из памяти или другого КЭШа.*



MOSI-протокол

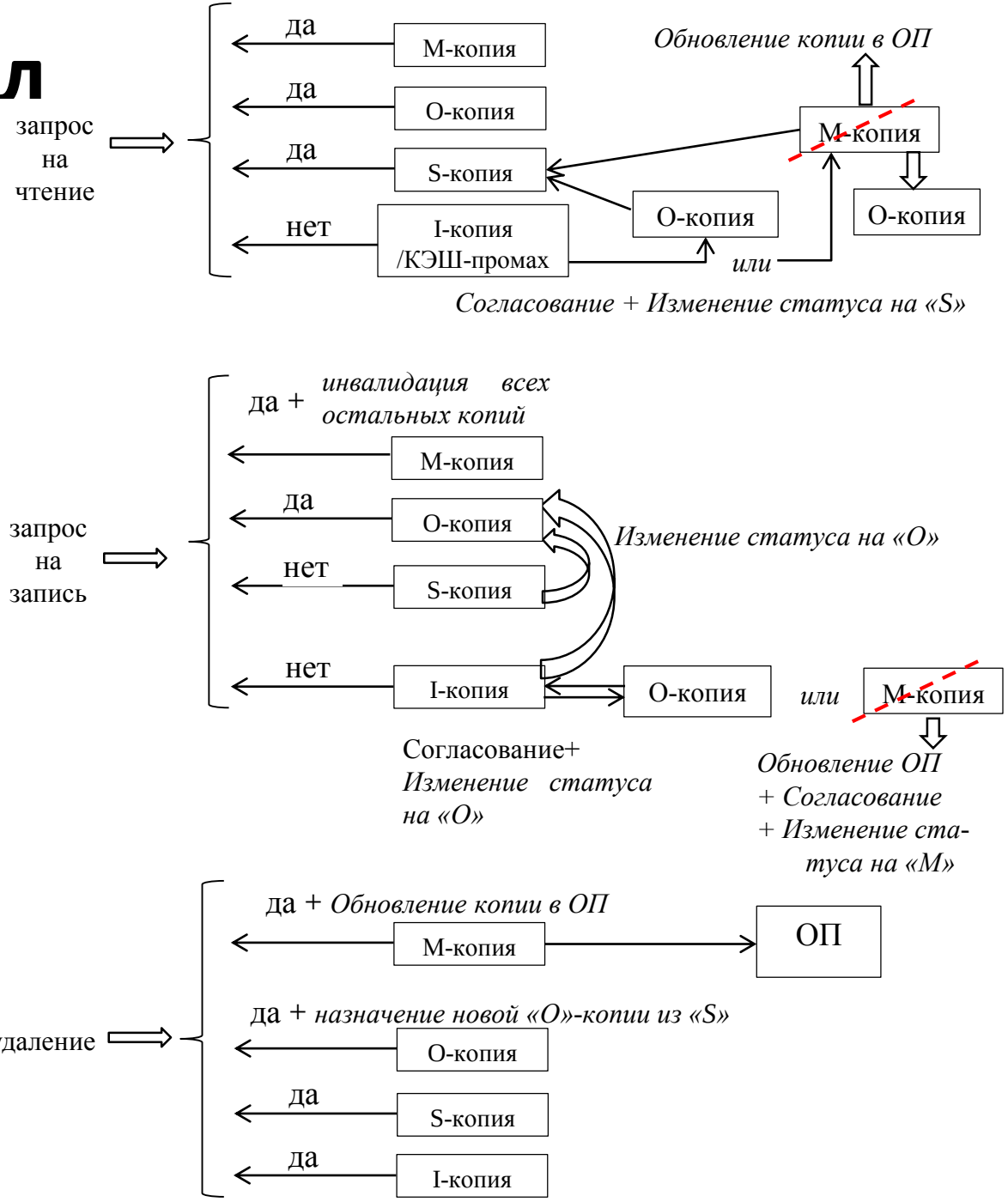
	M	O	S	I
M	✗	✗	✗	✓
O	✗	✗	✓	✓
S	✗	✓	✓	✓
I	✓	✓	✓	✓

Modified - …

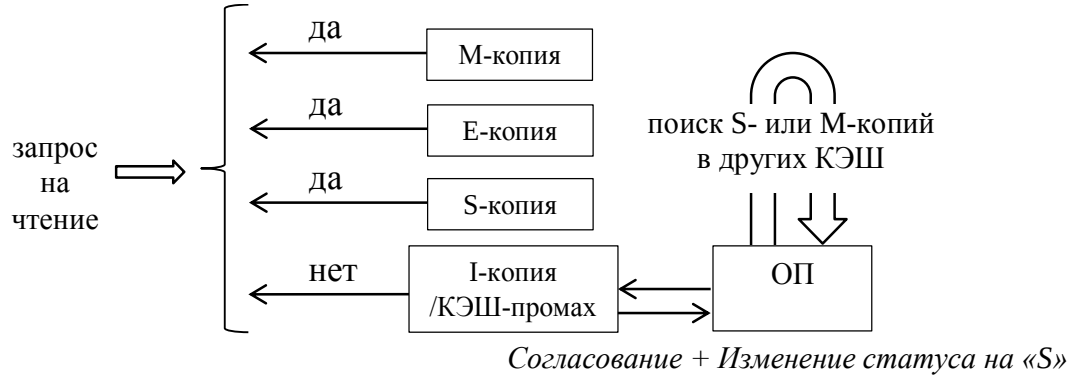
Shared - …

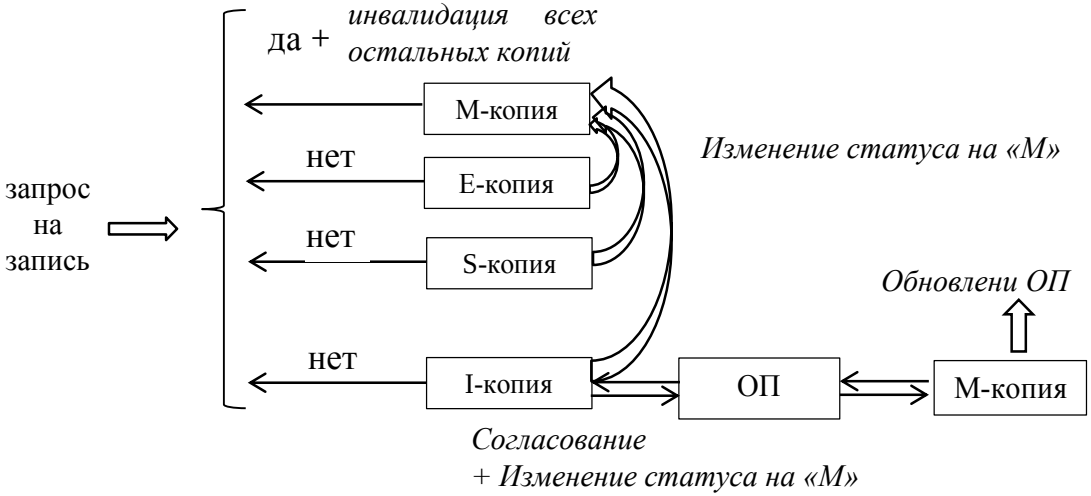
Invalid - …

Owned - *этот блок (O-копия) принадлежит только одному процессору, который будет обслуживать запросы от других процессоров (S-копий) к этому блоку (вместо ОП).*



MESI протокол





	M	E	S	I
M	✗	✗	✗	✓
E	✗	✗	✗	✓
S	✗	✗	✓	✓
I	✓	✓	✓	✓

Modified - …

Exclusive - *блок совпадает с ОП и отсутствует в других КЭШах*

Shared - …

Invalid - …

MOESI протокол

	M	O	E	S	I
M	✗	✗	✗	✗	✓
O	✗	✗	✗	✓	✓
E	✗	✗	✗	✗	✓
S	✗	✓	✗	✓	✓
I	✓	✓	✓	✓	✓

MOSI + MESI

Modified *(Модифицированная) единственная копия строки, содержащая достоверные изменения по сравнению с ОП.*

Owned *(в собственности) одна из нескольких общих копий, но с исключительным правом вносить в неё изменения. КЭШ-владелец O-копии должен транслировать вносимые изменения на все другие КЭШи, имеющие S-копии того же блока.*

Exclusive *(Эксклюзивная) копия строки полностью соответствует содержанию ОП и имеется в единственном экземпляре.*

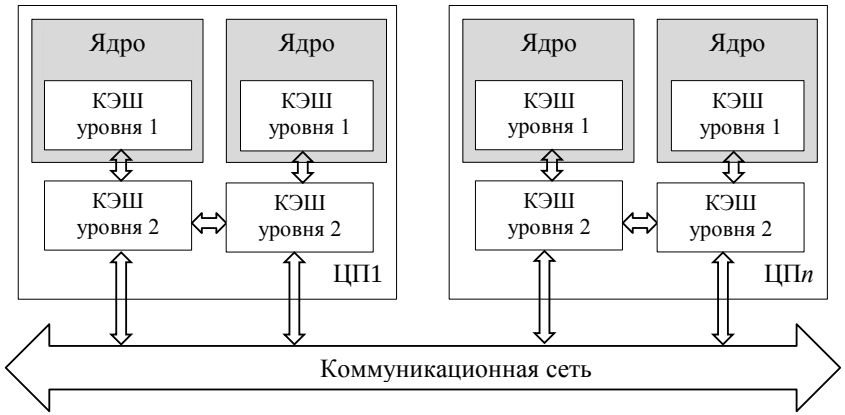
Shared *(общая) копия – одна из нескольких одинаковых копий в системе. В отличие от протокола MESI S-копия может отличаться от копии в ОП. Если ни один КЭШ не провёл свою копию в O-статус, то актуальной является копия в ОП (а не S-копии). Этот КЭШ не имеет права модифицировать копию, но может изменить её статус на "E" или "M" после признания недействительными всех остальных S-копий.*

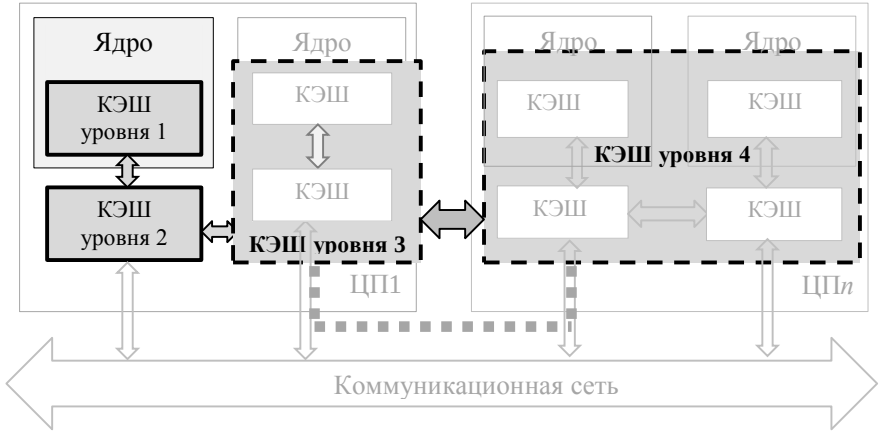
Invalid (недействительная) копия должна быть прочитана заново.

Преимущества MOSI, MOESI

В ВС, реализующих протокол MOESI, требуется наличие высокоскоростных соединений между локальными КЭШами процессоров (как в AMD в отличие от Intel, где есть связь только с ОП).

Тогда любая операция чтения сопровождается проверкой КЭШей соседей – если нужные данные находятся в одном из них (O-, M-, S-, E-копия), то и читаются они прямо оттуда; причем сохранение этих данных в оперативную память при этом не производится. Т.о. процессоры могут эффективно использовать данные из КЭШ-памяти друг друга.





В итоге в мультипроцнссорах с многоядерными процессорами КЭШ-память первого и второго уровней «соседнего» ядра/прпоцессора может работать как КЭШ третьего уровня (L3); а кэш-память «чужих» процессоров в многопроцессорной системе – как КЭШи четвертого уровня.

MESIF протокол

Это протокол поддержания когерентности, разработанный компанией Intel для CC-NUMA (cache coherense) архитектур. Протокол основан на протоколе MESI, в который добавлено еще одно состояние.

Протокол состоит из пяти статусов копий:

Modified (M),
Exclusive (E),
Shared (S),
Invalid (I),
Forward(F)

} *такие же, как и в MESI.*

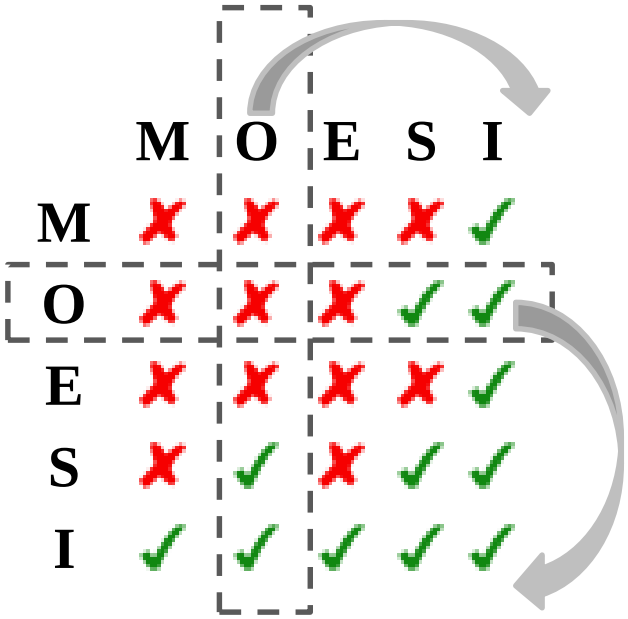
	M	E	S	I	F
M	✗	✗	✗	✓	✗
E	✗	✗	✗	✓	✗
S	✗	✗	✓	✓	✓
I	✓	✓	✓	✓	✓
F	✗	✗	✓	✓	✗

F-статус – это специализированная форма “S”- статуса, аналог “O”-статуса из MOESI. Дополнительное состояние F означает, что кэш является единственным ответчиком для любых запросов к данной копии (КЭШ-линии = строке КЭШ).

При копировании F-строки в соседний кэш новая копия получает F состояние.

Сравнение протоколов MESIF и MOESI

MOESI → MOESIO

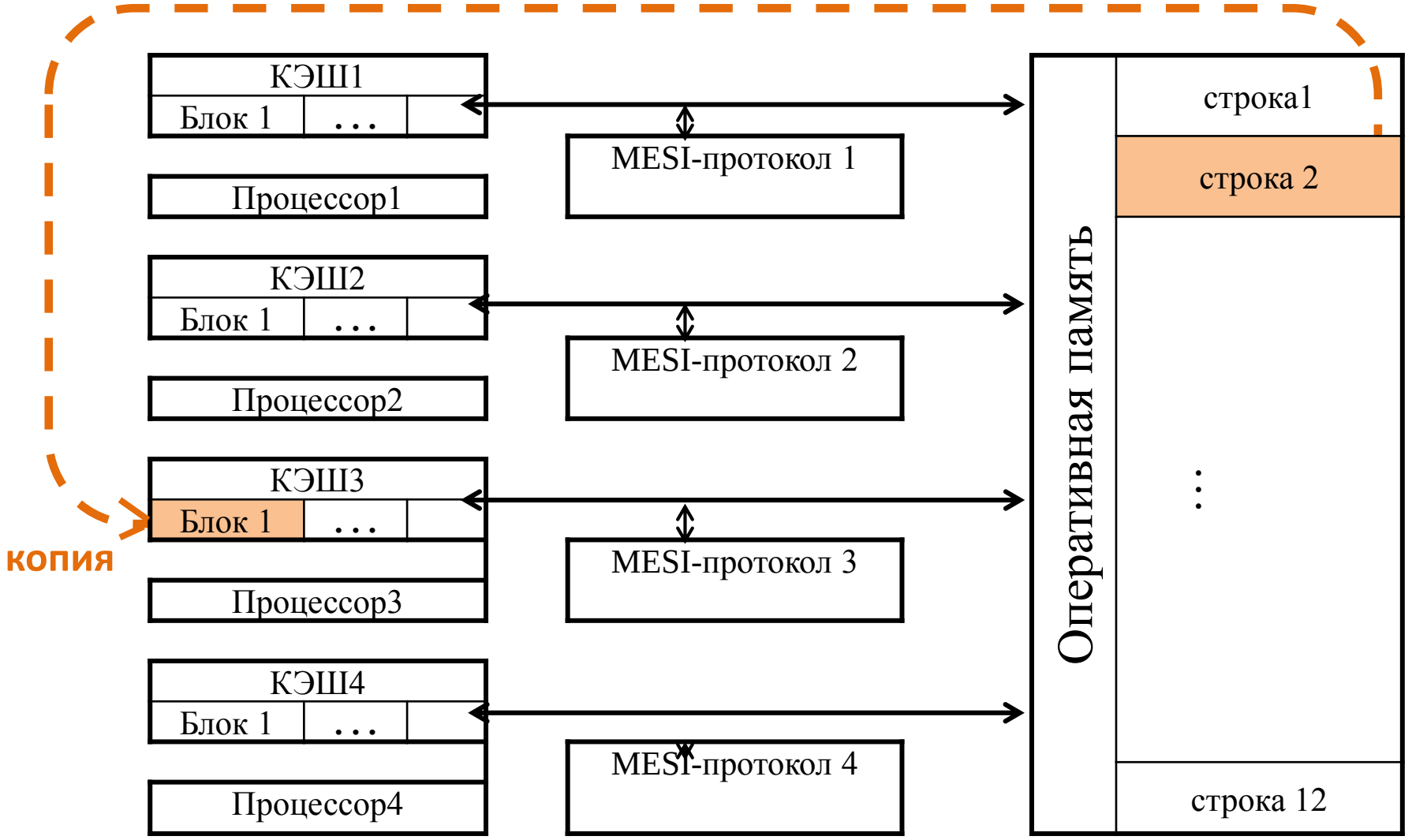


	M	O	E	S	I
M	✗	✗	✗	✗	✓
O	✗	✗	✗	✓	✓
E	✗	✗	✗	✗	✓
S	✗	✓	✗	✓	✓
I	✓	✓	✓	✓	✓

	M	E	S	I	O
M	✗	✗	✗	✓	✗
E	✗	✗	✗	✓	✗
S	✗	✗	✓	✓	✓
I	✓	✓	✓	✓	✓
O	✗	✗	✓	✓	✗

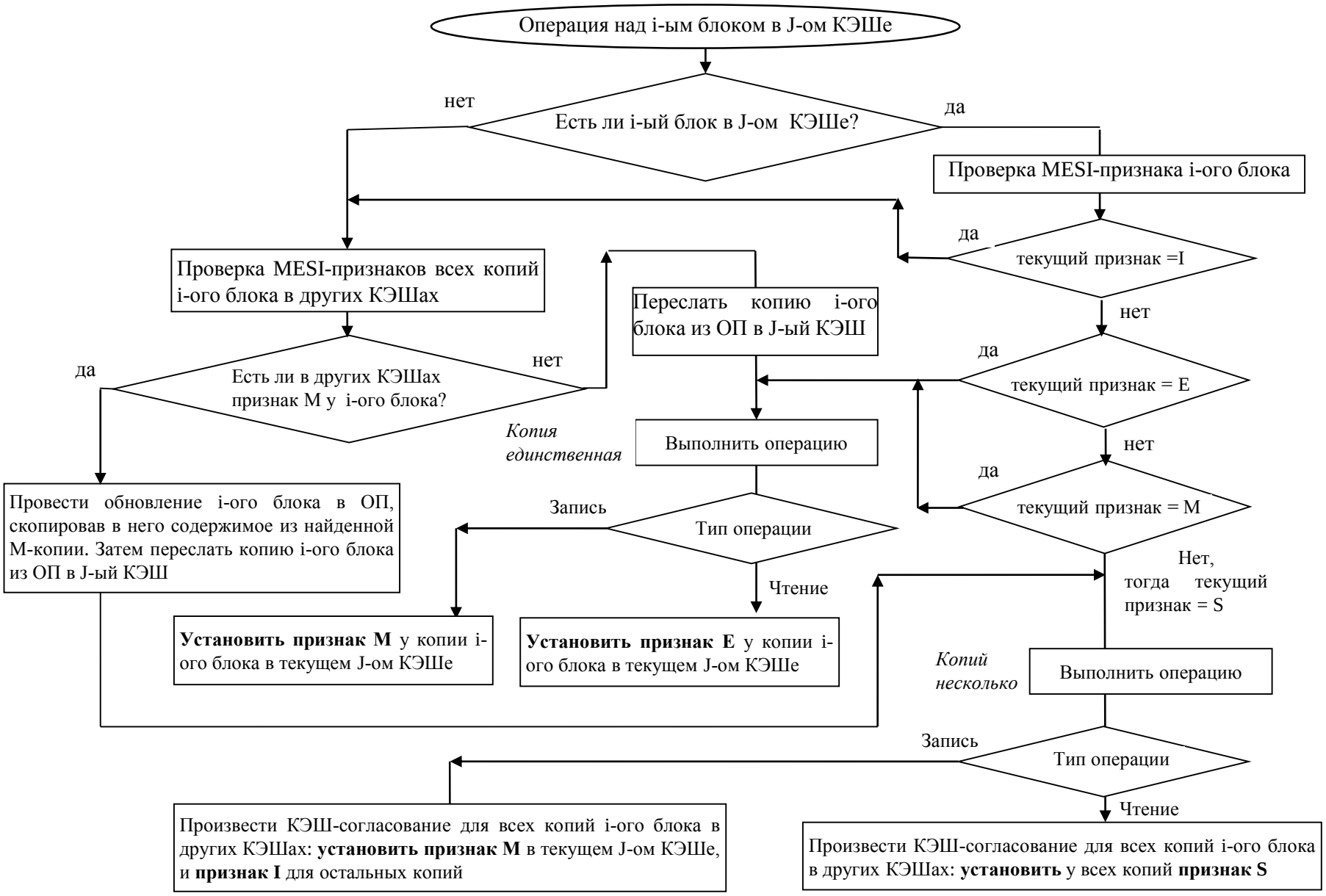
	M	E	S	I	F
M	✗	✗	✗	✓	✗
E	✗	✗	✗	✓	✗
S	✗	✗	✓	✓	✓
I	✓	✓	✓	✓	✓
F	✗	✗	✓	✓	✗

Алгоритм процедуры работы MESI-протоколов



Строка КЭШ = блок КЭШ = КЭШ-линия (cache line)

Алгоритм работы MESI-протокола



Стратегии обновления строк ОП

При записи строк существует несколько методов обновления старой информации в ОП. Эти методы называются стратегией обновления строк основной памяти.

Сквозная запись (Write-Through), при которой все изменения в копиях строк КЭШ-памяти сразу попадают в строки ОП.

Обратная запись (Write-Back). Если адрес объектов, по которым есть запрос обновления, существует в КЭШ-памяти (копия есть), то обновляется только КЭШ-память, а основная память не обновляется. Если адреса объекта обновления нет в КЭШ-памяти (копии нет), то в неё из основной памяти пересылается строка, содержащая этот адрес, после чего обновляется только КЭШ.

Изменения в копиях строк накапливаются, работа текущего процесса с данными происходит только с копиями этих данных из КЭШ, и только в определённый момент все накопленные изменения выгружаются в ОП (либо периодически, например дождавшись освобождения системой шины, либо при необходимости доступа к этим данным, например, со стороны других устройств).

Достоинство: это ликвидирует многочисленные задержки и значительно увеличивает производительность подсистемы памяти.

http://moodle.technion.ac.il/pluginfile.php/313428/mod_resource/content/1/Core%20uArch%20-%20Advanced%20VLSI%20Arch.pdf

Core Cache Size/Latency/Bandwidth

Metric	Nehalem	Sandy Bridge	Haswell
L1 Instruction Cache	32K, 4-way	32K, 8-way	32K, 8-way
L1 Data Cache	32K, 8-way	32K, 8-way	32K, 8-way
Fastest Load-to-use	4 cycles	4 cycles	4 cycles
Load bandwidth	16 Bytes/cycle	32 Bytes/cycle (banked)	**64 Bytes/cycle**
Store bandwidth	16 Bytes/cycle	16 Bytes/cycle	**32 Bytes/cycle**
L2 Unified Cache	256K, 8-way	256K, 8-way	256K, 8-way
Fastest load-to-use	10 cycles	11 cycles	11 cycles
Bandwidth to L1	32 Bytes/cycle	32 Bytes/cycle	**64 Bytes/cycle**
L1 Instruction TLB	4K: 128, 4-way 2M/4M: 7/thread	4K: 128, 4-way 2M/4M: 8/thread	4K: 128, 4-way 2M/4M: 8/thread
L1 Data TLB	4K: 64, 4-way 2M/4M: 32, 4-way 1G: fractured	4K: 64, 4-way 2M/4M: 32, 4-way 1G: 4, 4-way	4K: 64, 4-way 2M/4M: 32, 4-way 1G: 4, 4-way
L2 Unified TLB	4K: 512, 4-way	4K: 512, 4-way	**4K+2M shared: 1024, 8-way**

All caches use 64-byte lines