X. Процессорные устройства

- Классификация процессорных устройств.
- Обобщенная структура универсального процессорного устройства.
- Архитектура конвейерного суперскалярного процессора P6.
- Устройства управления.
- Арифметико-логические устройства.

Литература

1. Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем: Учебник для вузов. – СПб.: Питер, 2004. – 668 с.: ил.
2. Шагурин И.И., Бердышев Е.М. Процессоры семейства Intel P6. Архитектура, программирование, интерфейс. – М.: Горячая линия – Телеком, 200. – 248 с., ил.
3. Бессонов О. Обзор микроархитектур современных десктопных процессоров. www.ixbt.com
4. IA-32 Intel® Architecture Software Developer's Manual
5. Касперски К. Техника оптимизации программ. Эффективное использование памяти. – СПб.: БХВ-Петербург, 2003. – 464 с.

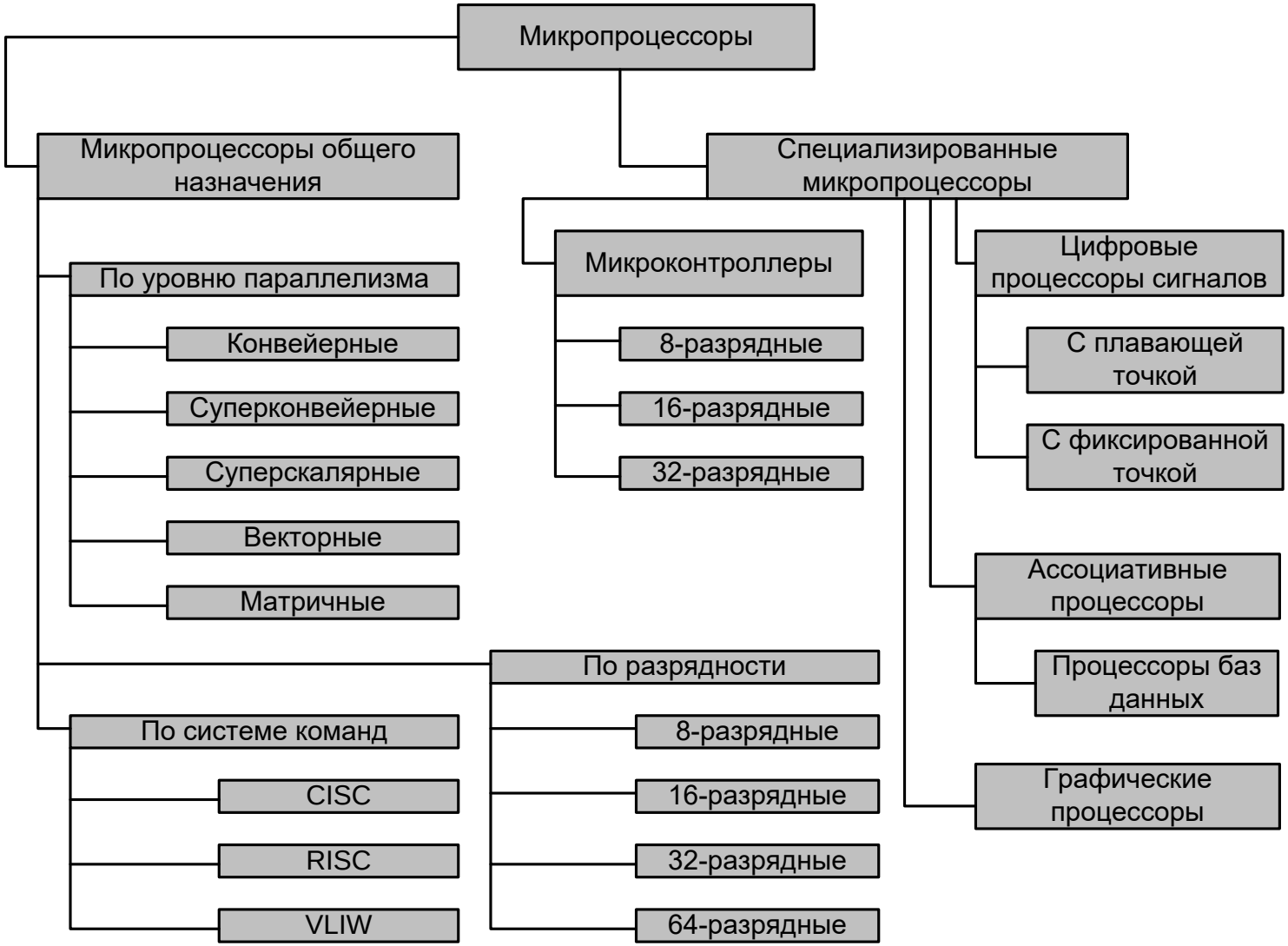
Процессором (процессорным ядром) называется устройство ЭВМ, непосредственно осуществляющее процесс переработки информации и управление им в соответствии с заданным алгоритмом, который, как правило, представлен программой.

ЭВМ может содержать несколько процессоров. Процессор, управляющий вычислительным процессом, называется центральным.

Микропроцессором называется функционально законченное устройство, представляющее собой вариант процессора (или нескольких процессорных ядер) современной ЭВМ и реализованное в виде одной или нескольких СБИС.

Микропроцессорный комплект представляет собой совокупность микропроцессора и специализированных ИС, совместимых по временным, электрическим и конструктивным параметрам, совместное использование которых позволяет реализовать основные функции ЭВМ.

Классификация процессорных устройств



Обобщенная структура универсального процессорного устройства

Архитектурные особенности:
-Конвейерное исполнение команд.

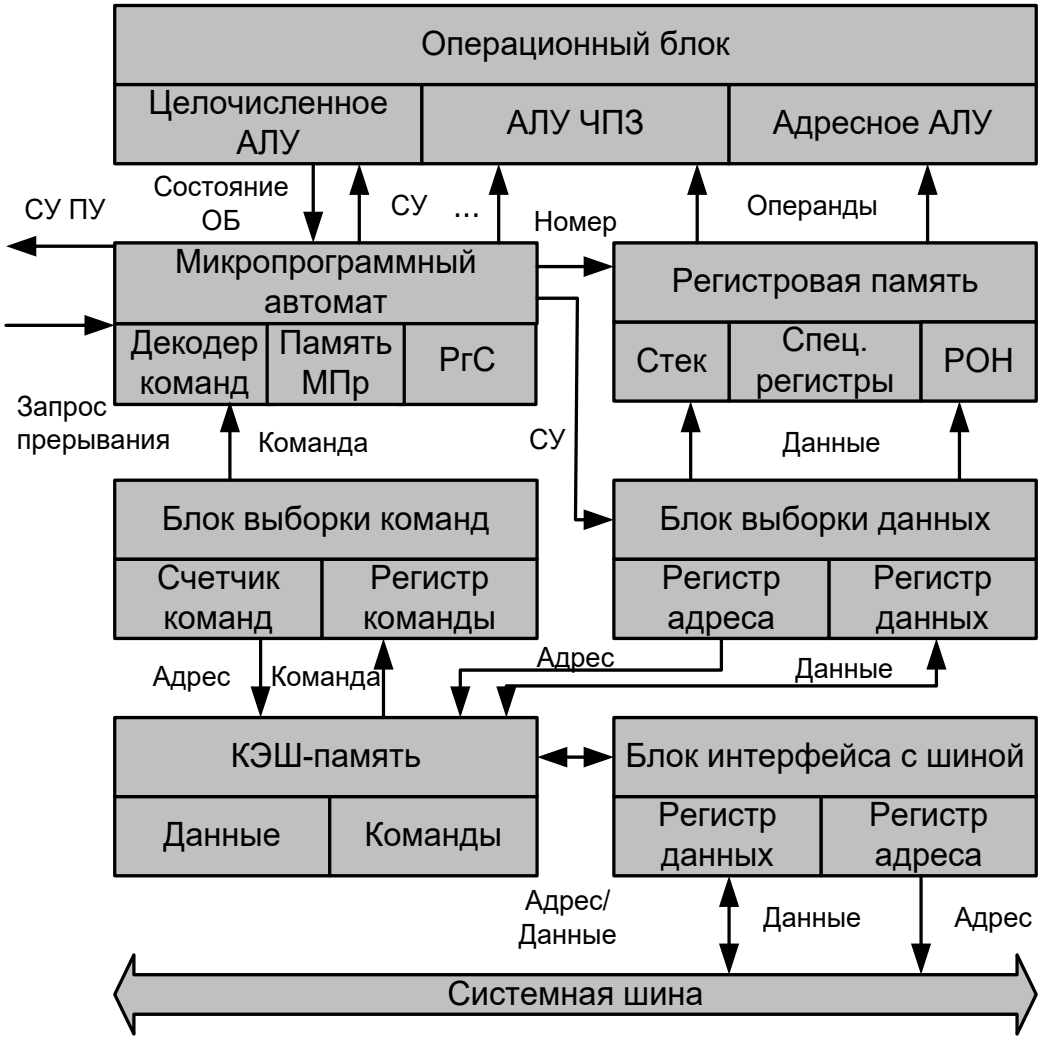
-Внутренняя КЭШ-память.

-Целочисленное АЛУ.

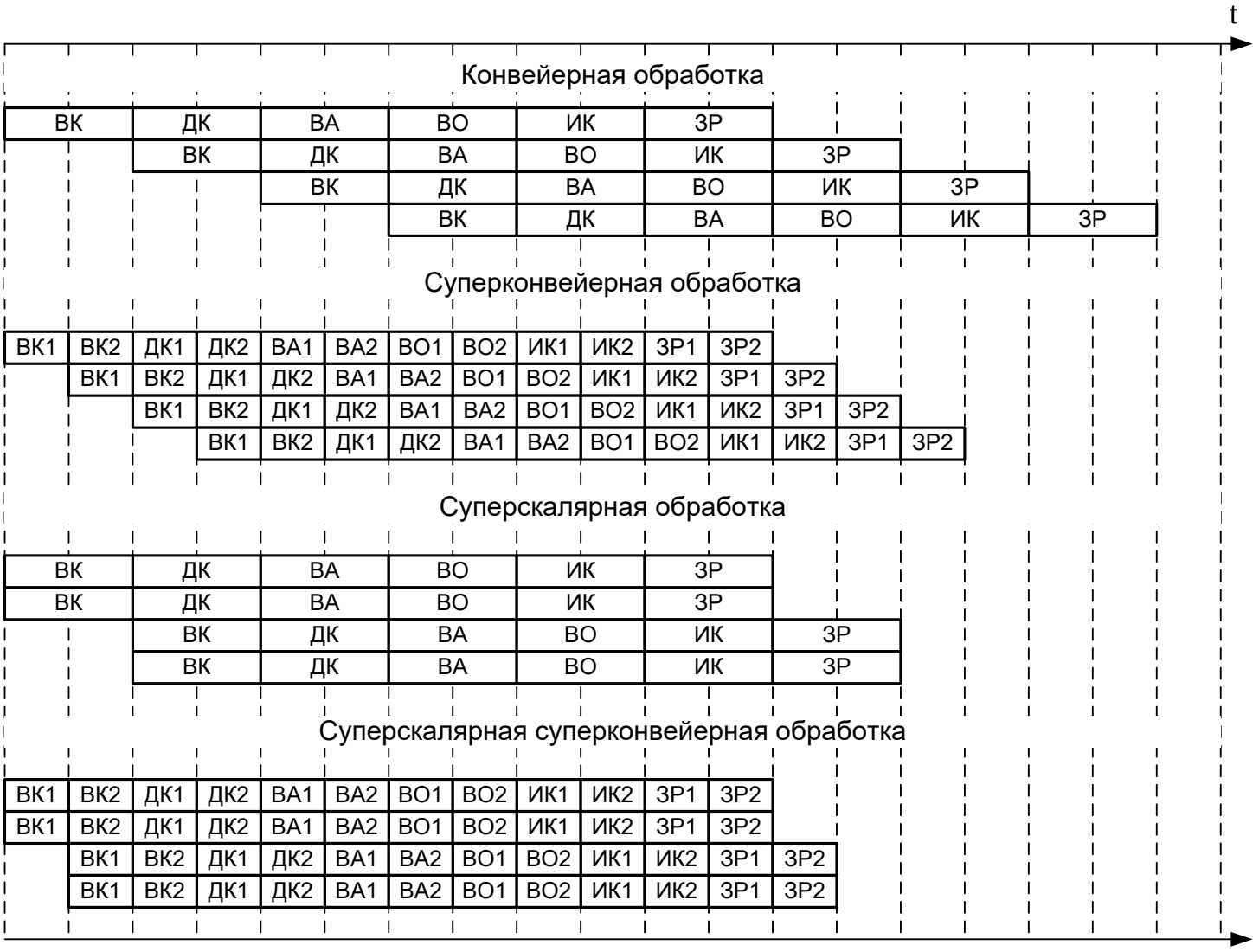
-Устройство выполнения операций над числами с плавающей запятой.

-Обработка прерываний от ПУ.

-Поддержка мультипроцессорной обработки.



Сравнение способов организации параллельных вычислений



Конфликты в конвейере (риски)

1. Структурный риск

   Команды одновременно обращаются к одному и тому же ресурсу (например, к ОП).

2. Риск по данным

   Команды имеют зависимость по данным.

   O(i) – множество ячеек, изменяемых командой i;
   I(j) – множество ячеек, читаемых командой j.

А) Чтение после записи (ЧПЗ).

$$O(i) \cap I(j) \neq \varnothing$$

Б) Запись после чтения (ЗПЧ).

$$I(i) \cap O(j) \neq \varnothing$$

В) Запись после записи (ЗПЗ).

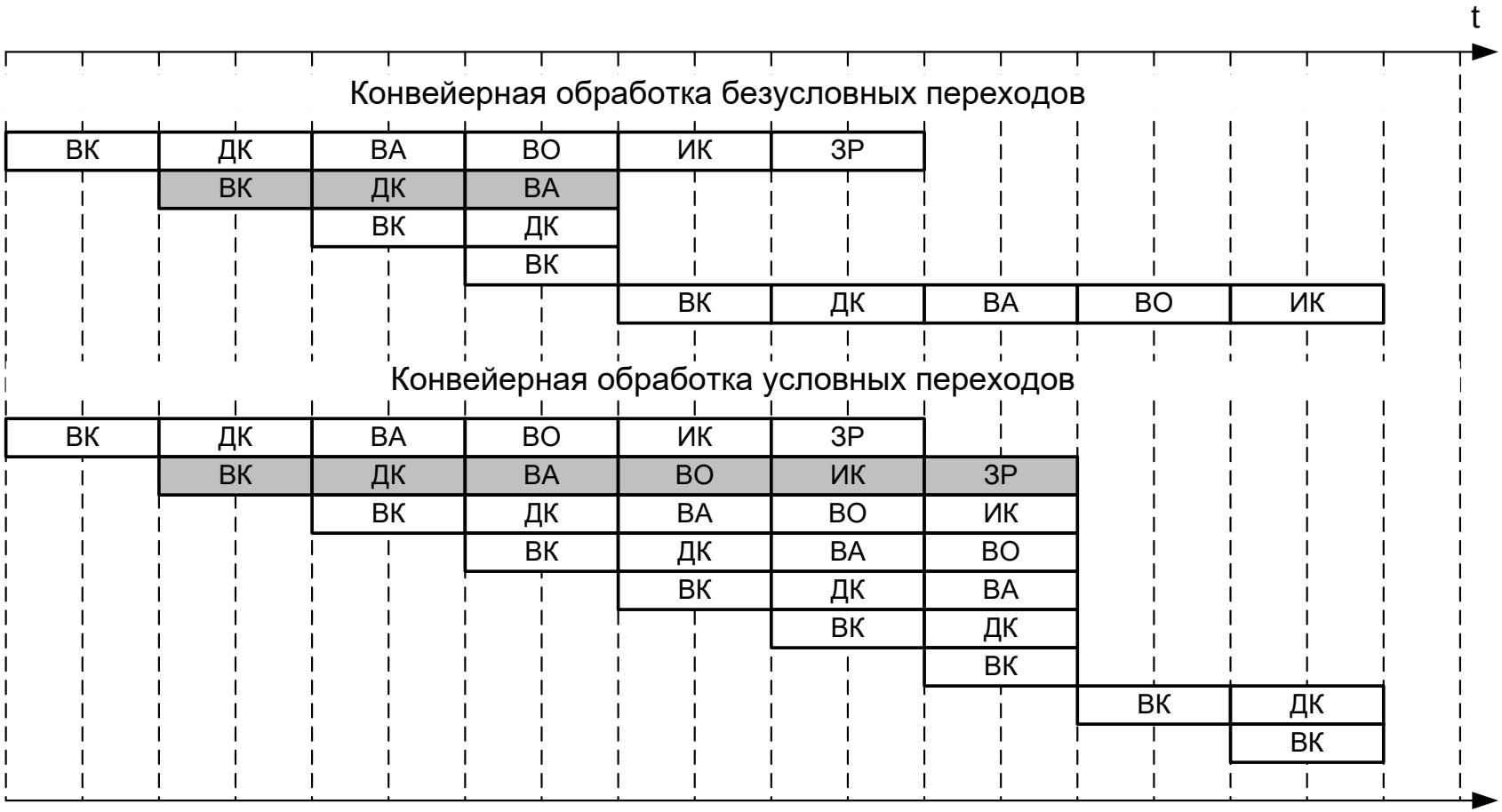
$$O(i) \cap O(j) \neq \varnothing$$

3. Риск по управлению.

Из-за наличия команд перехода (10-20% потока команд) возможна неоднозначность при выборе очередной инструкции.

Потери в лучшем случае: сброс всех поступивших команд за время декодирования команды ветвления.
Потери в худшем случае: сброс всех поступивших команд за время декодирования, выборки операндов и исполнения команды ветвления.

Временные потери при обработке команд переходов

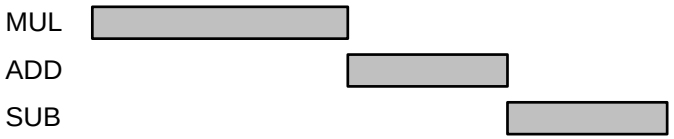


Способы устранения конфликтов по данным, находящихся в регистрах

Пример 1:

```
MUL BX        ;  (DX:AX = AX * BX)
ADD AX,BX     ;  (AX = AX + BX)
SUB BX,2      ;  (BX = BX - 2)
```

ЗПЗ, ЧПЗ, ЗПЧ, ЗПЧ, ЗПЧ

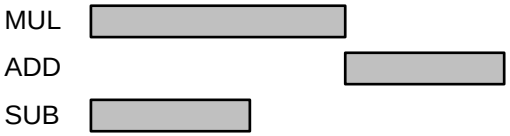


Правило:

Каждый новый результат записывается в новый регистр замещения.

```
MUL BX        ;  (DX':AX' = AX * BX)
ADD AX,BX     ;  (AX'' = AX' + BX)
SUB BX,2      ;  (BX' = BX - 2)
```

ЧПЗ

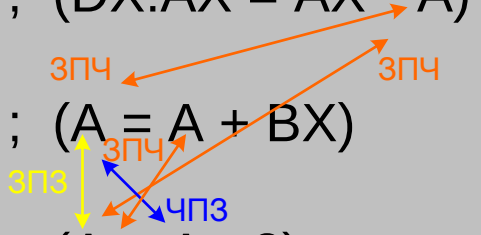


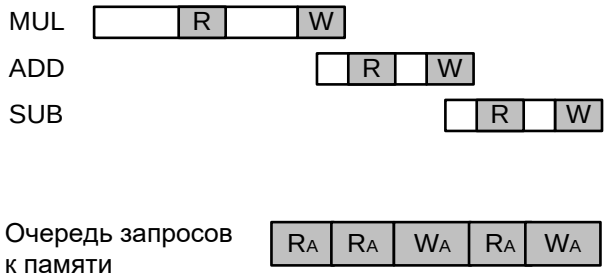
Конфликт типа ЧПЗ по данных, находящимся в регистрах, может быть устранен с помощью бита достоверности

Способы устранения конфликтов по данным, находящихся в памяти

Пример 2:

```
MUL A          ;  (DX:AX = AX * A)
ADD A,BX       ;  (A = A + BX)
SUB A,2        ;  (A = A - 2)
```

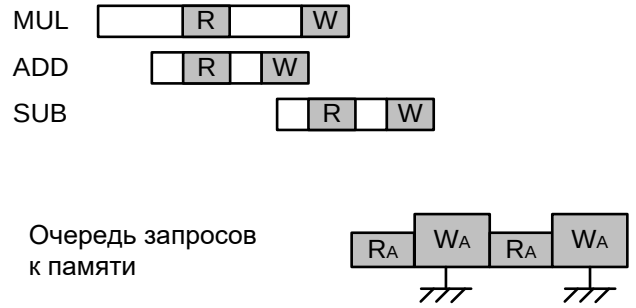




Правило:

При обнаружении конфликтов по данным, находящимся в ОП, запросы на запись результатов в память выполняется упорядочено.

```
MUL A          ;  (DX:AX = AX * A)
ADD A,BX       ;  (A = A + BX)
SUB A,2        ;  (A = A - 2)
```



Способы устранения конфликтов по управлению

-Дублирование ступеней конвейера для обработки обеих ветвей

-Оптимизация кода на этапе компиляции с целью увеличения полезной нагрузки на дублированные ступени конвейера.

-Предсказание переходов.

Способы предсказания переходов

Точность предсказания: отношение числа правильно предсказанных переходов к их общему количеству.

Эффективность алгоритмов предсказания зависит от использования статистических данных, накопленных:
-заранее при компиляции и тестовых прогонах (статическое предсказание переходов);
-полученных в процессе исполнения программы (динамическое предсказание переходов).
-На основе статического и динамического подходов.

Стратегии статического предсказания переходов

-Переход происходит всегда (60-70%).
-Переход не происходит никогда (50%).
-Переход выполняется по результатам профилирования (75%).
-Переход определяется по коду операции (75%).
-Переход выполняется исходя из направления (85%).
-При первом выполнении переход имеет место всегда (90%).

Стратегии динамического предсказания переходов

-Одноуровневое предсказание: использует Шаблонную Таблицы Истории (Pettern History Table).

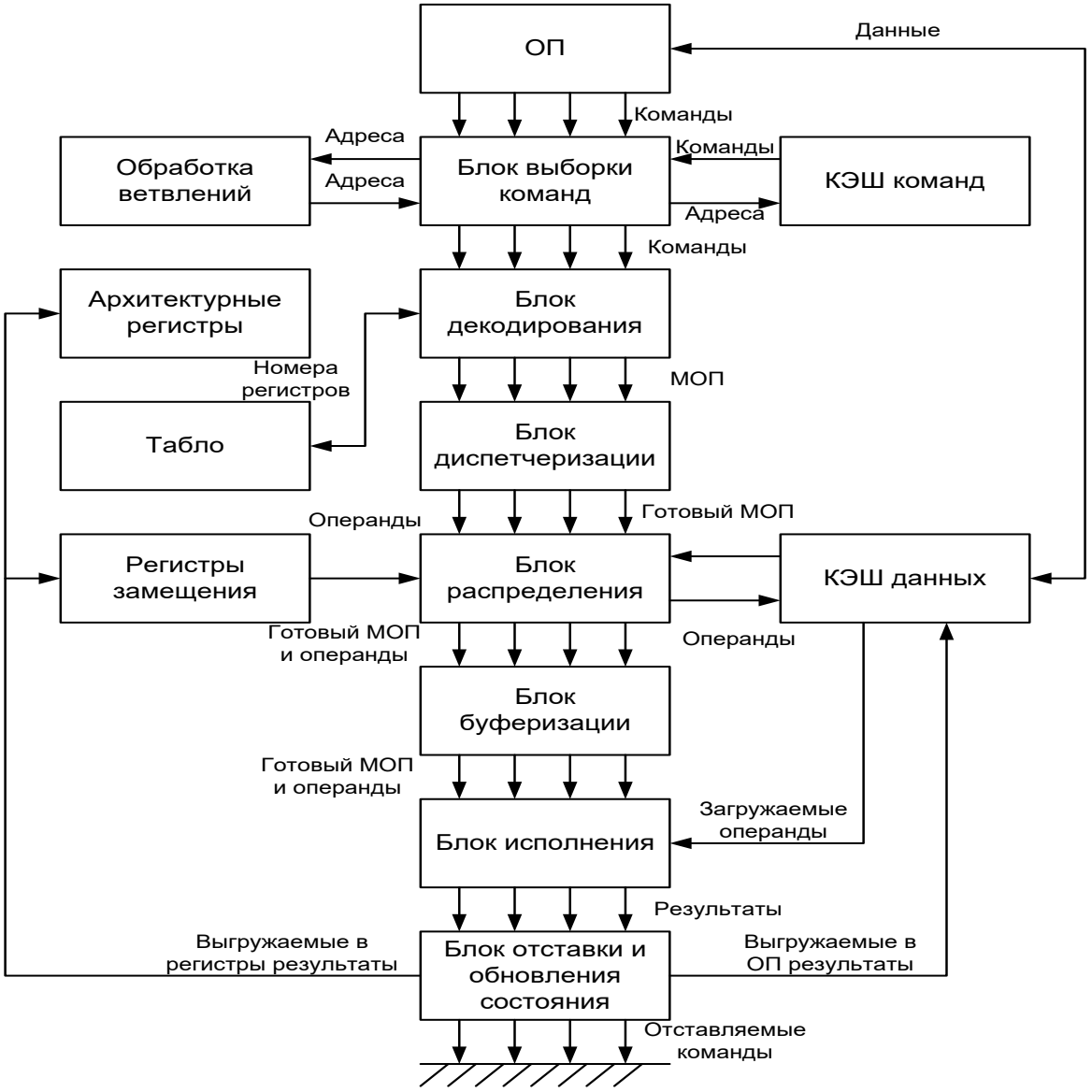
Выборка информации может происходить: по адресу команды перехода; по истории всех команд перехода; по истории исполнения только предсказываемой команды перехода.

Алгоритм предсказания зависит от размера строк РНТ. При хранении одного бита переход предсказывается в соответствии с предыдущим итогом выполнения команды (точность ~78%).

При хранении двух бит учитывается переход для двух последних исполнений команды (точность ~82%).

- Таблица меток перехода (Branch Target Buffer)
-Двухуровневое предсказание.
-Гибридное предсказание

Обобщенная схема суперкскалярного суперконвейерного процессора



Структура процессора P6

История процессоров с архитектурой IA32

Модель	Годы выпуска	Функциональность
8086	1978	16 разрядный микропроцессор. Сегментация. 20-разрядная шина адреса (до 1 Мб).
80286	1982	Защищенный режим с использованием дескрипторного регистра (четыре уровня привилегий, поддержка сегментов только для чтения и только для исполнения, ограничение прав доступа). Поддержка виртуальной памяти. 24-разрядная шина адреса (16 Мб)
80386	1985	32 разрядный микропроцессор. Поддержка 16 разрядного кода. Режим Virtual-8086. Сегментная и непрерывная модель памяти. Страничная организация виртуальной памяти (4 Кб страницы). 32-разрядная шина адреса (4 Гб). Совмещение исполнения команд с обращением к памяти.
80486	1989	Конвейер команд (5 стадий). 8 Кб кэш первого уровня. Интегрированный FPU. Режим энергосбережения.

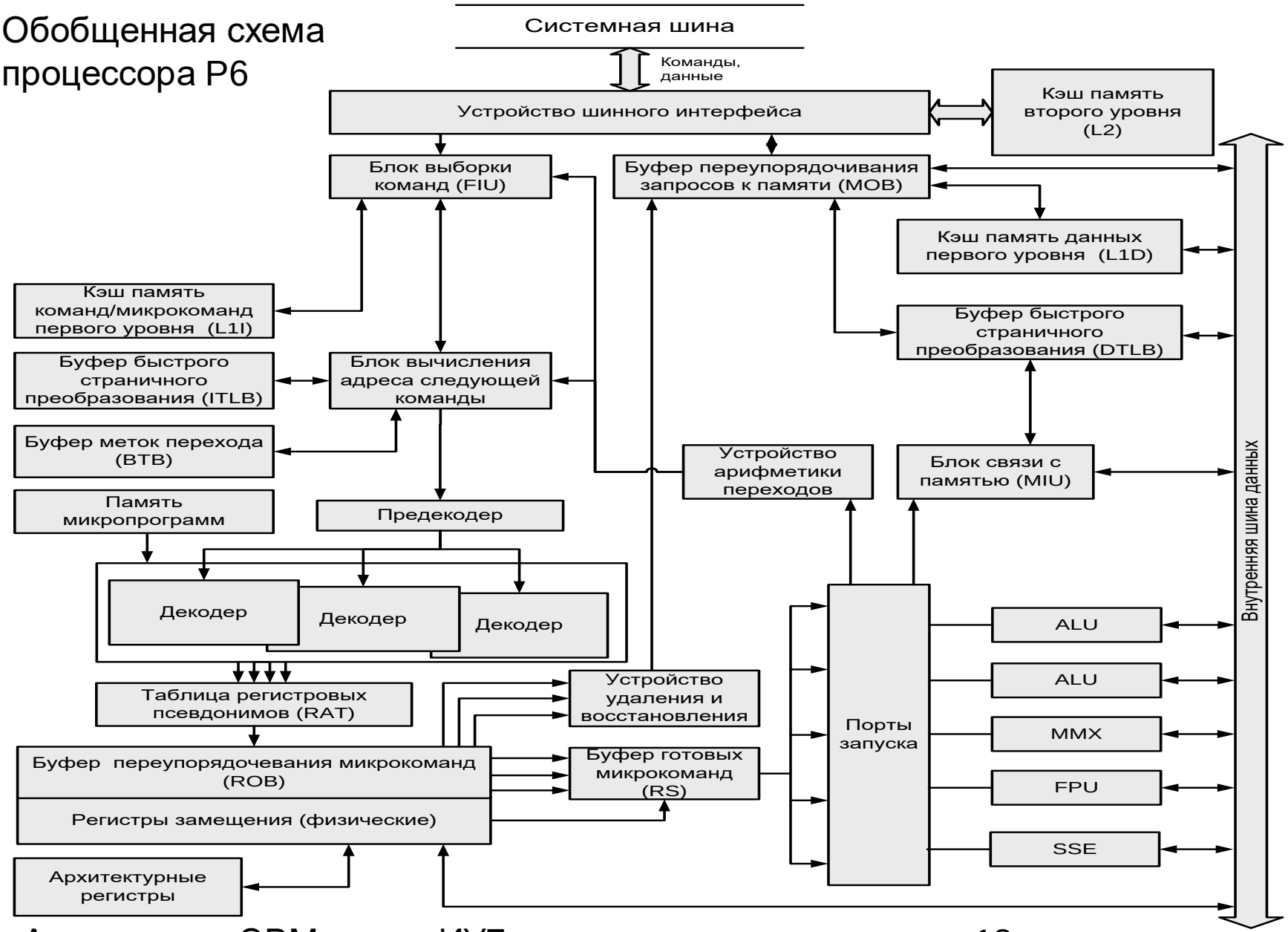
История процессоров с архитектурой IA32 (продолжение)

Модель	Годы выпуска	Функциональность
Pentium	1993	Двухконвейерная архитектура (возможность суперскалярной обработки). Раздельные кэш первого уровня команд и данных (по 8 Кб, протокол MESI Write Back). Предсказание ветвления. 4Кб и 4Мб страницы. 64 разрядная внешняя шина данных. Поддержка пакетного режима. Поддержка построения многопроцессорных ВС. MMX SIMD обработка (64 разряда).
Семейство P6 (Pentium Pro, Celeron, Pentium II, Pentium III)	1995	Суперскалярная обработка на основе техники переупорядочивания (до трех операций за такт). Динамическое исполнение команд (анализ зависимостей по данным, неупорядоченное и спекулятивное исполнение, предсказание ветвления). Интегрированный смешанный кэш второго уровня (до 2 Мб). SSE (128 разрядов)
Семейство NetBurst (Pentium 4, Xeon, Pentium Ex. Ed.)	2000	Быстрое исполнение на удвоенной скорости. Гиперконвейерная суперскалярная организация. Глубокая предвыборка. Кэш трасс. SSE2 и SSE3. Hyper-Threading, Intel Virtualization Technology, Intel 64 Architecture, Dual Core

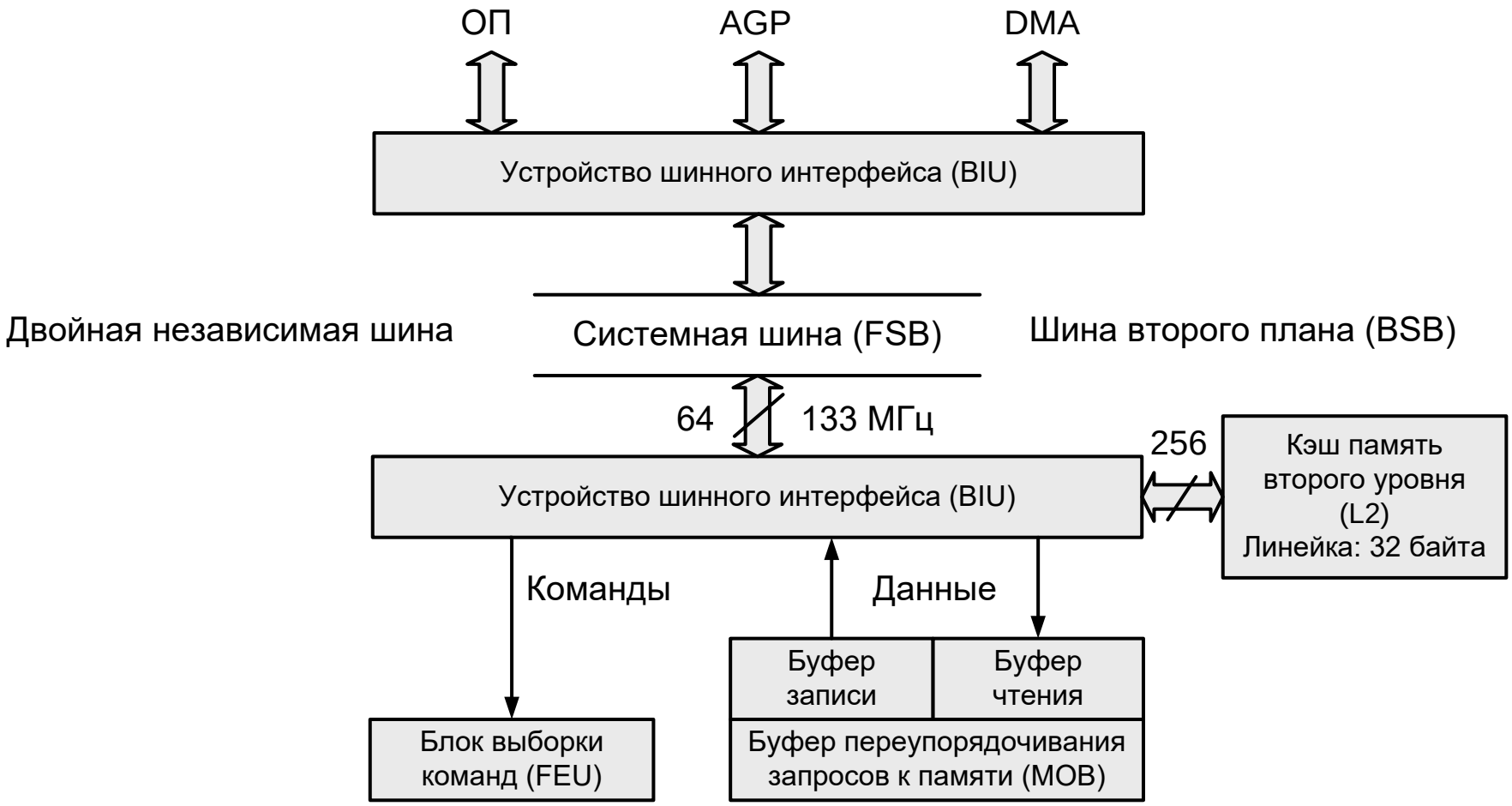
История процессоров с архитектурой IA32 (окончание)

Модель	Годы выпуска	Функциональность
Семейство Pentium M	2003	Низкое энергопотребление, Сбалансированная производительность
Семейство Intel Core (Intel Xeon 5100, 5300, Intel Pentium Dual-Core, Intel Core 2, Intel Core 2 Quad	2006	Многоядерная архитектура, Intel 64 Architecture, Intelligent Power Capability
Семейство Intel Atom (Intel Atom)	2008	Сверхнизкое энергопотребление, Dual Pipeline In-order Execution, Dynamic Cache Sizing
Семейство Intel Nehalem (Intel Core i7, Intel Xeon 5500, 7500)	2008	Выделенный блок управления энергопотреблением, До 8 ядер. SSE4
Семейство Intel Westmere (Intel Core i3,i5,i7, Intel Xeon 5600)	2010	Усовершенствованный блок управления энергопотреблением, Интегрированное графическое ядро, Интегрированный контроллер памяти DDR3

Обобщенная схема процессора P6



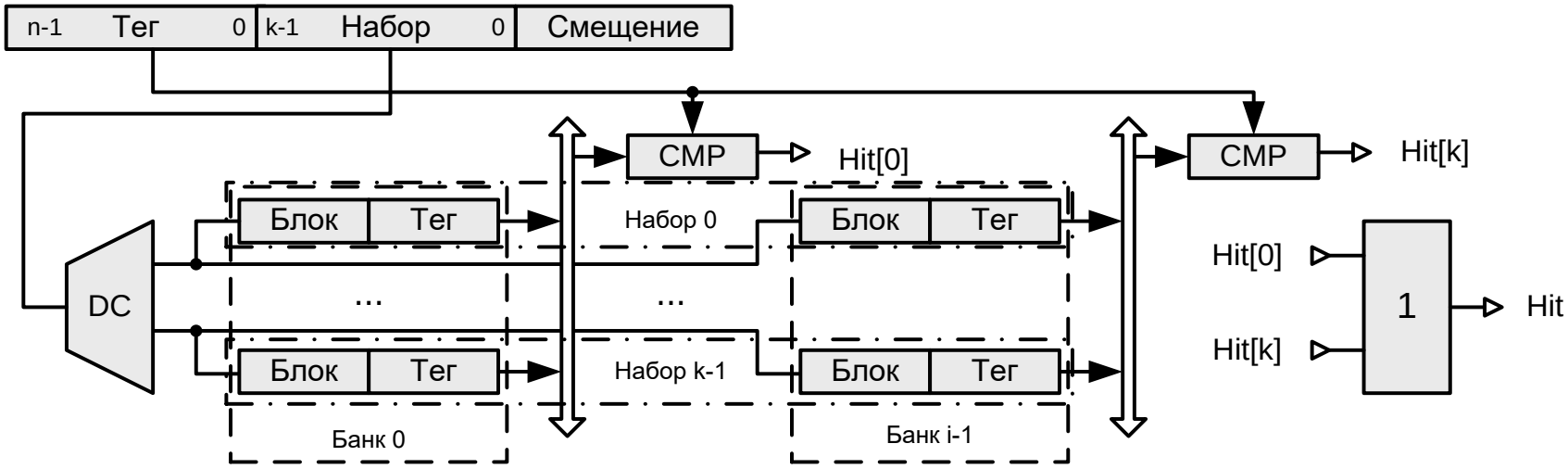
Устройство шинного интерфейса



Обращение к ОП выполняется через L2. Разделение системной магистрали на две независимые шины снижает нагрузку на системную магистраль до 10% от максимальной.

Кэш память второго уровня (L2)

Тип: Наборно-ассоциативная неблокируемая
Размер: до 2 Мб
Размер линейки: 32 байта.
Ассоциативность: 4
Политика записи: Write Back
Алгоритм замещения: LRU
Протокол: MESI



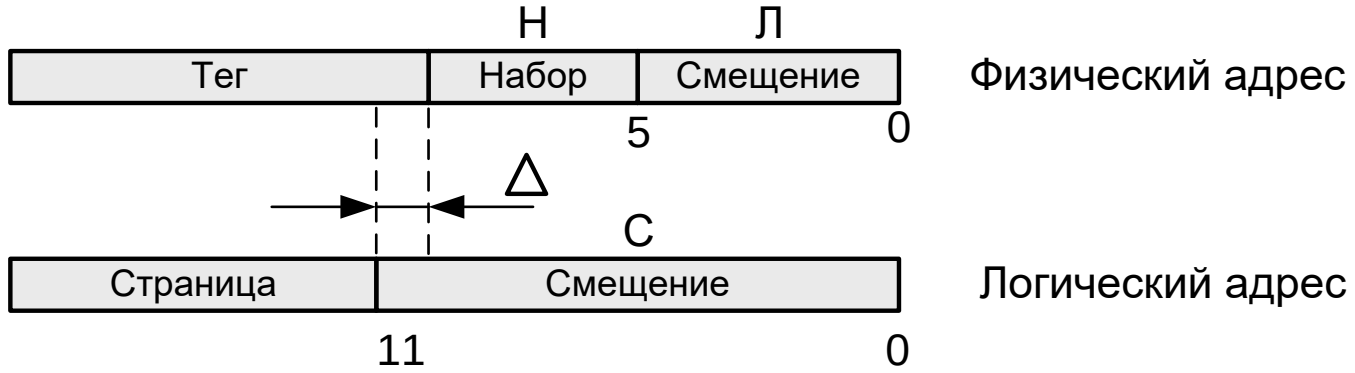
Проблема алиасинга

Н*Л = (Размер кэш) / (Ассоциативность)

При размещении в кэш данных по адресам с шагом **Н*Л** они размещаются в одном и том же наборе (эффективная ассоциативность = 1).

Проблема выборки по физическому адресу

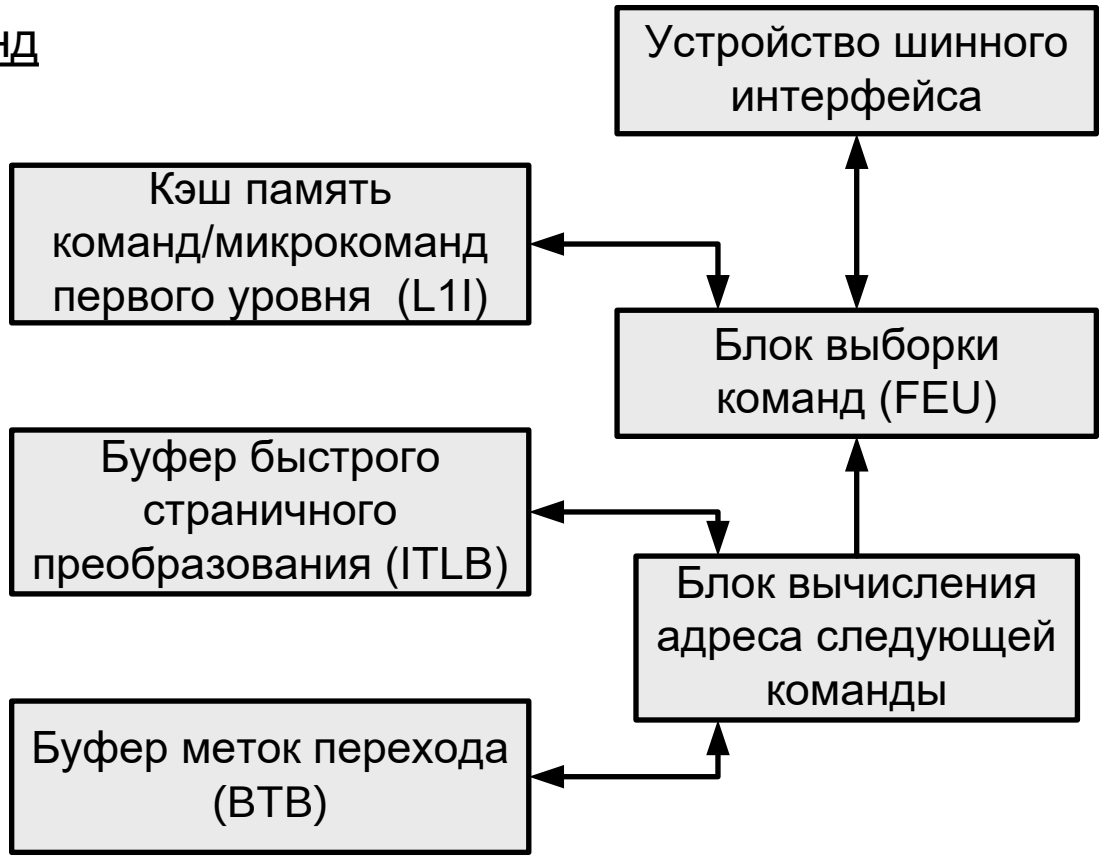
Выборка из кэш-памяти осуществляется по физическому адресу.
Для ускорения доступа используют часть логического адреса.



При обнаружении логического адреса в команде с учетом того, что (**С>=Н*Л**) уже известен набор, а возможно и часть тега.
Тогда можно заранее определить наличие кандидатов на выборку из кэш или заранее обнаружить кэш промах.

Блок выборки команд

Блок выборки команд получает физический адрес очередной команды из Блока вычисления адреса следующей команды. По этому адресу сначала происходит обращение в L1I. Если указанного блока команд (линейки) там нет, то запрос передается в BIU.



Блок вычисления адреса следующей команды реализует механизм статического и динамического предсказания с использованием наборно-ассоц. BTB (Branch Target Buffer). BTB в P6 состоит из 512 элементов (4-х ассоциативный).

Для преобразования логического адреса в физический используется ITLB (Instruction Translation Lookaside Buffer) и DTLB (Data Translation Lookaside Buffer).

Структура TLB

Номер лог. страницы	V	R	M	A	Номер физ. страницы

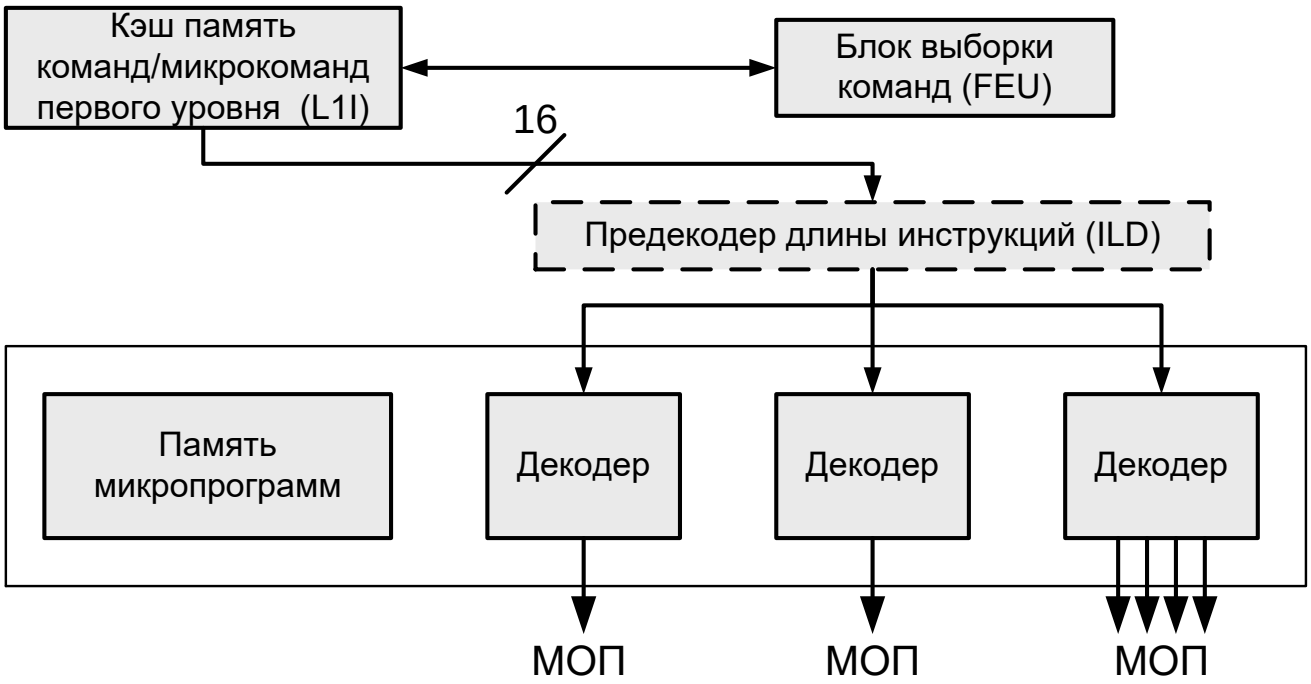
Информация, записываемая в TLB, не подлежит кэшированию.

При доступе к данным или командам по адресу, для преобразования которого информация в TLB отсутствует, необходимо обратиться в оперативную память дважды: сначала за информацией из таблицы страниц, и после преобразования за самими данными или командами.

Этот можно устранить с помощью предвыборки. Команды предвыборки:

Команда	**Pentium III (32 байта)**	**Pentium 4 (128 байт)**	**Примечание для P6**
prefetchNTA	Загрузка только в L1D. В L2 не загружаются	Загрузка в L2. В L1D не загружаются	Загрузка в ближайший кэш для немедленного использования
Prefetch0	Загрузка в L1D и L2	Загрузка только в L2. В L1D не загружается	Загрузка в кэш всех уровней
Prefetch1	Загрузка только в L2. В L1D не загружается	Загрузка в L2. В L1D не загружаются	Загрузка в кэш кроме нулевого (только L2)
Prefetch2	Загрузка только в L2. В L1D не загружается	Загрузка в L2. В L1D не загружаются	Загрузка в кэш кроме нулевого и первого (только L2)

Декодеры



Команды поступают из L1I блоками по 16 байт. В предекодере определяются границы команд и наличие префиксов.

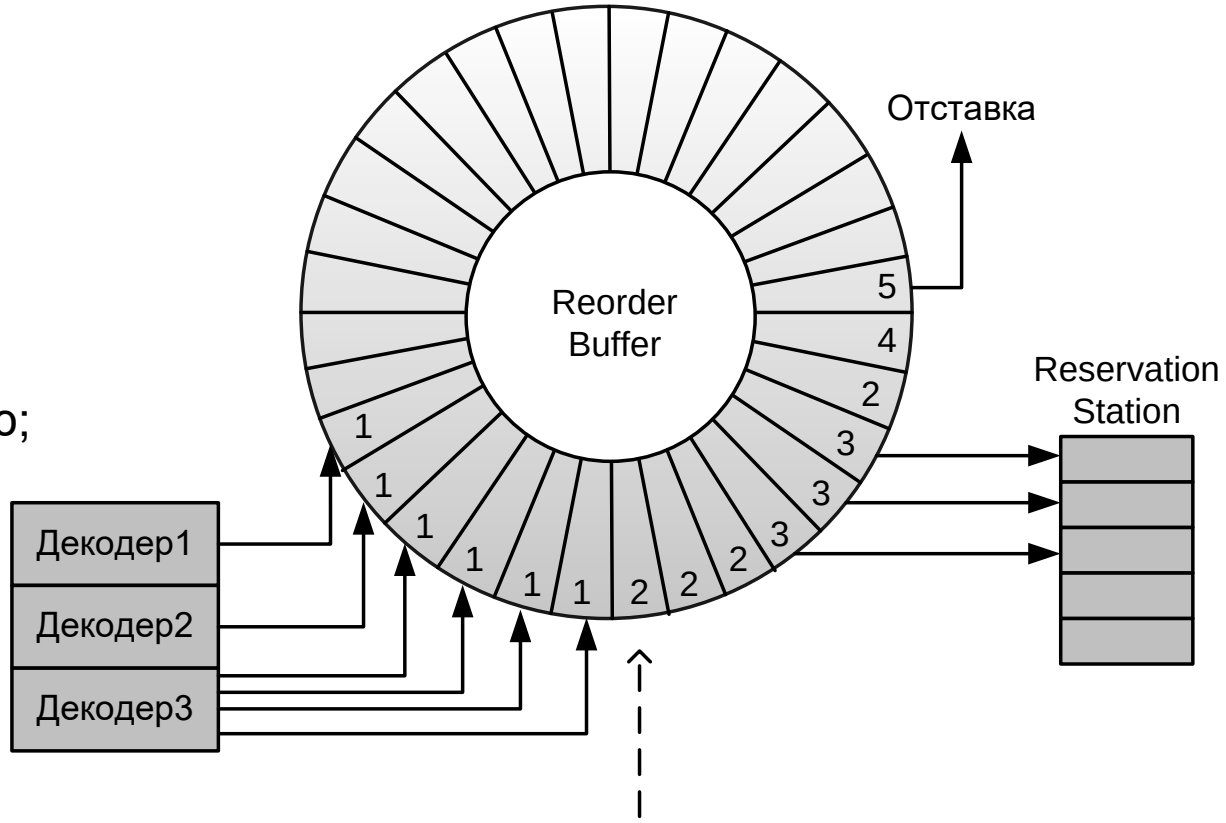
Декодер состоит из трех параллельных каналов: два канала для декодирования простых команд, порождающих одну микрооперацию; один декодер обрабатывает любые инструкции и генерирует по 4 МОП за такт.

Для загрузки операндов и исполнения операций порождаются различные микрооперации. Для вычисления адреса также порождается МОП.

Буфер переупорядочивания микрокоманд

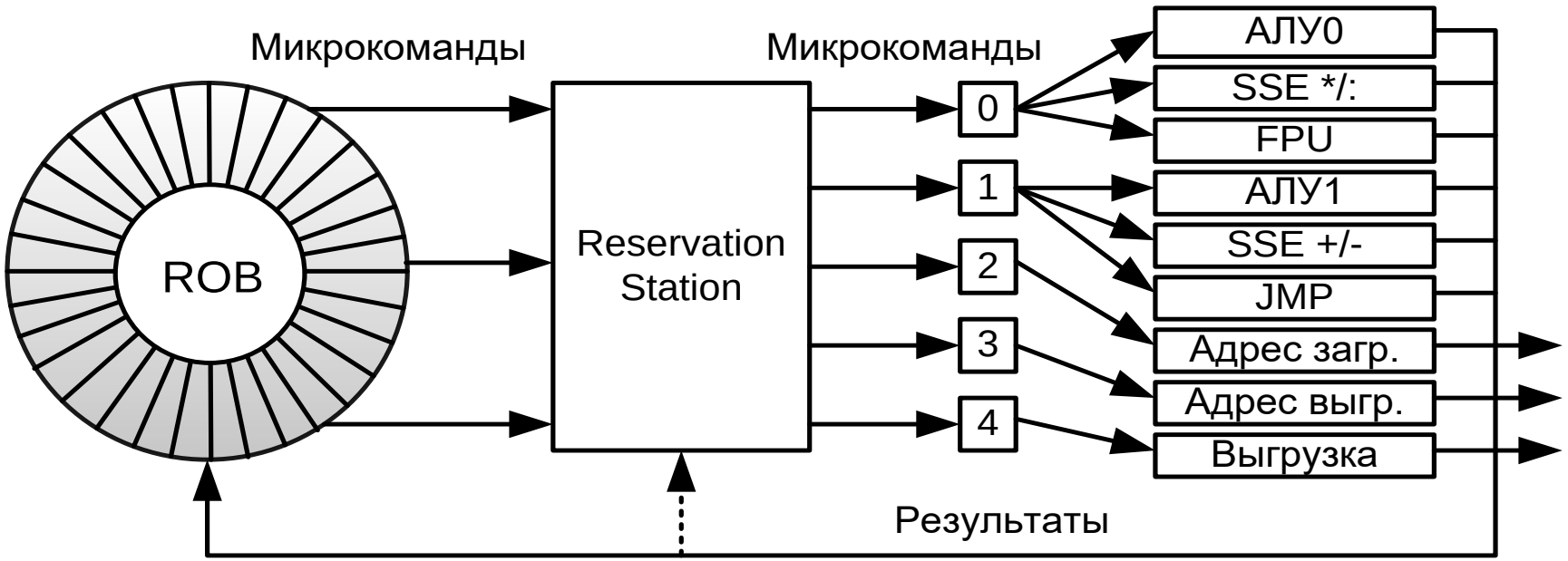
-Микрооперации помещаются в ROB в исходном порядке.
-Исполнение МОП происходит неупорядочено по мере готовности операндов.
-Удаление (отставка) МОП происходит упорядочено из-за: прерываний, исключений, точек останова, неправильно предсказанных переходов.

Микрокоманды в ROB могут находиться в одном из следующих состояний:
1. Не готова к исполнению;
2. Готова к исполнению;
3. Исполняется;
4. Исполнена и ожидает отставки;
5. Находится в процессе отставки.



Команда	Поле операции	Поле результата	Поле операнда1	Д1	Поле операнда2	Д2

Порты запуска и исполнительные устройства



В процессорах P6 пять портов запуска.

В RS в каждом такте могут быть помещены три микрооперации из ROB и пять микроопераций могут быть направлены в порты запуска. Если претендентов на исполнительное устройство несколько, то выбор Производится по алгоритму «псевдо-FIFO».

Загрузка и выгрузка

Выгрузка в память (store) происходит в соответствии с порядком отставки (в исходном порядке). Только после отставки возможно незначительное переупорядочивание для оптимизации работы BIU.
Загрузка (load) может происходить неупорядоченно в случае отсутствия зависимостей по данным.
Для ускорения выполнения микроопераций загрузки выполняется поиск требуемых данных в микрооперациях выгрузки (forwarding of data from stores to dependent loads).
Однако возможны приложения, в которых процессор не может обнаружить зависимость по данным (I/O operations).

Команды управления загрузкой и выгрузкой

Команда	Назначение	Примечание
lfence	Упорядочивание загрузки	Команда позволяет управлять загрузкой, запрещая переупорядочивать микрооперации загрузки до данной команды с микрооперациями после данной команды.
sfence	Упорядочивание выгрузки	Команда позволяет управлять выгрузкой, запрещая переупорядочивать микрооперации выгрузки до данной команды с микрооперациями после данной команды.
mfence	Упорядочивание загрузки и выгрузки	Команда позволяет управлять загрузкой и выгрузкой, запрещая переупорядочивать микрооперации загрузки и выгрузки до данной команды с микрооперациями после данной команды.

Достоинства P6

-Суперскалярная обработка.
-Интегрированная кэш-память.
-Сбалансированность фаз конвейера.

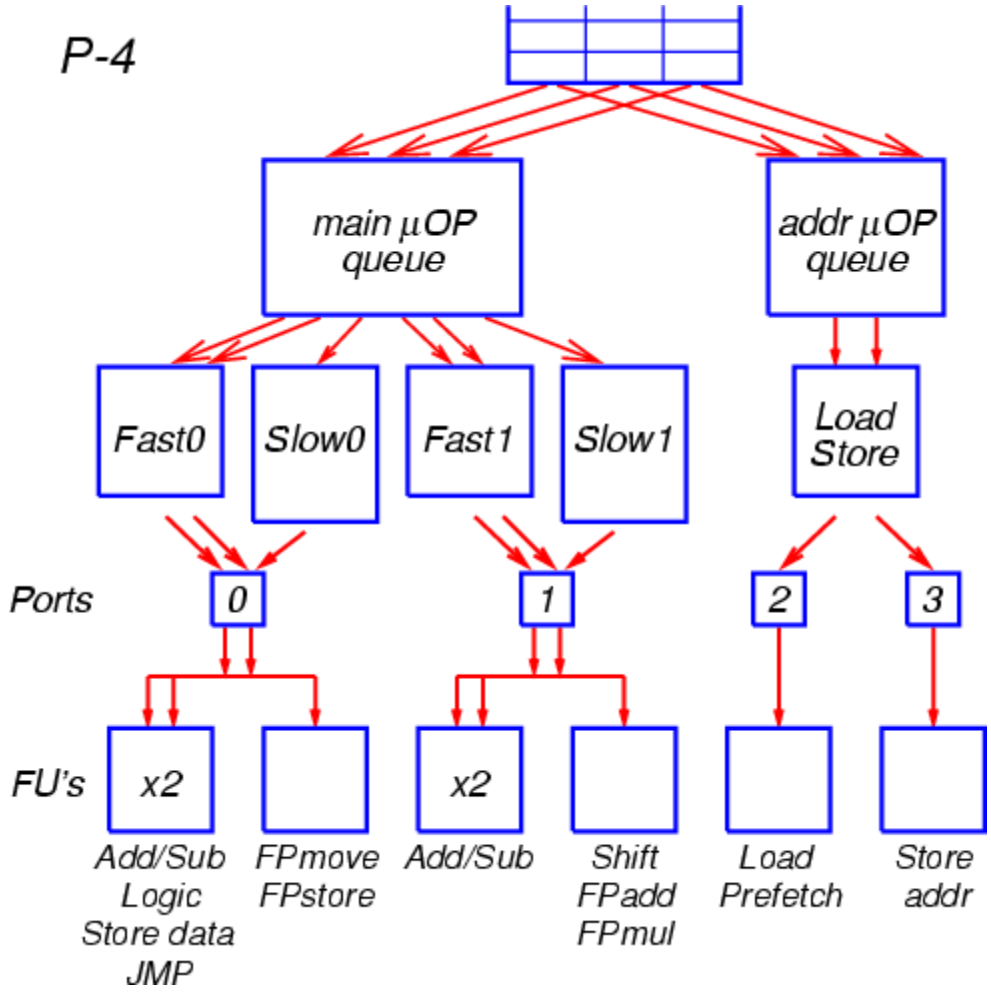
Недостатки P6

-Длительное декодирование сложных команд.
-Отсутствие слияния микроопераций загрузки/выгрузки и обработки.
-Малое количество входов ROB.
-Наличие медленных команд.

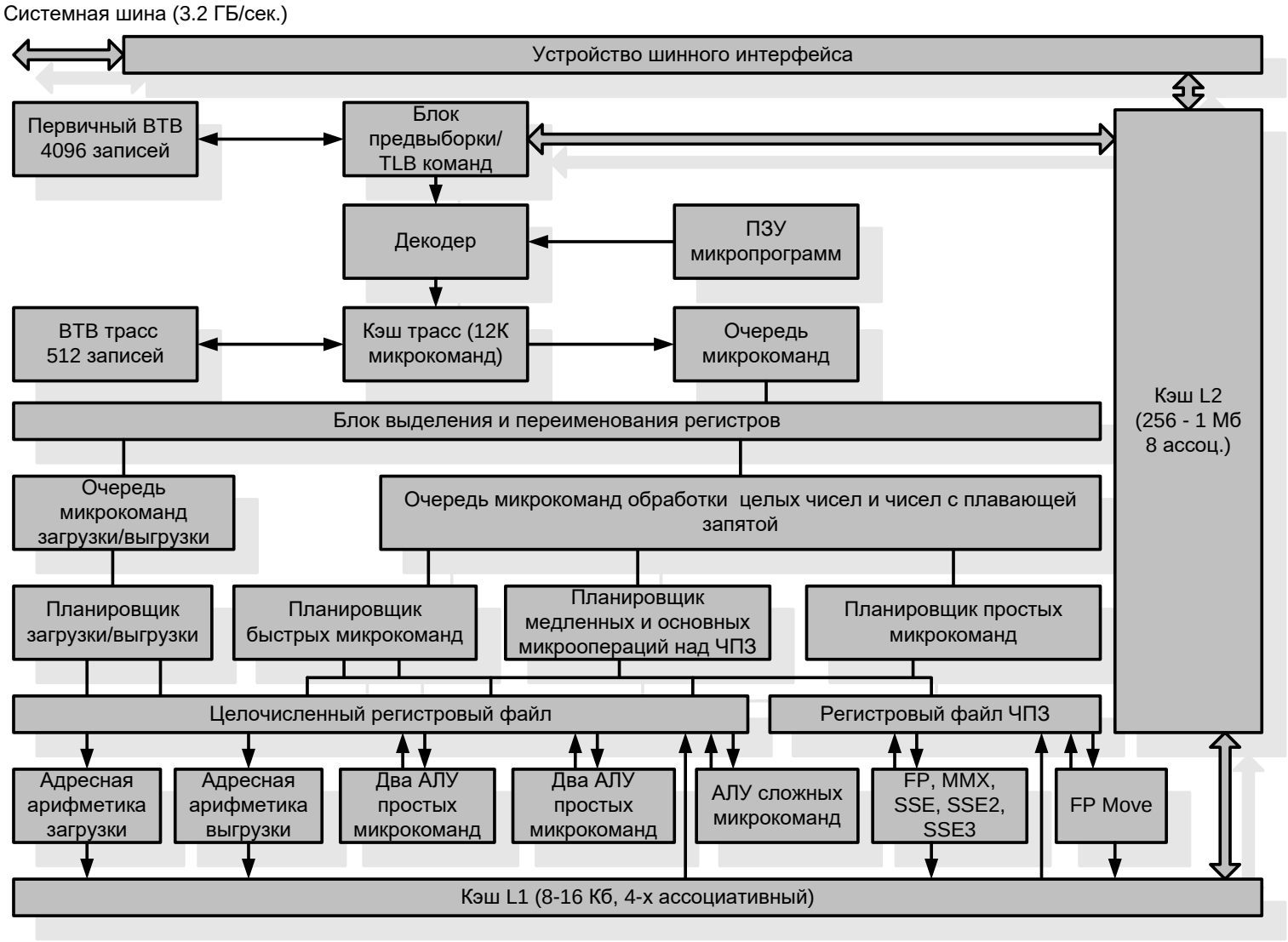
Отличие архитектуры NetBurst от P6

-Использование кэш-памяти первого уровня для хранения декодированных команд (кэш трасс, 12 КМОП). Это позволяет разворачивать циклы, ускоряет декодирование за счет выборки уже декодированных команд.
-Использование TBTB и TTLB для определения адресов в кэш трасс.
-Механизм ранней спекулятивной диспетчеризации, заключающийся в продвижении на исполнение МОПов, ожидающих операнды уже обрабатываемых МОПов.
-Разделение МОПов на медленные и быстрые.
-Работа АЛУ на удвоенной частоте.
-Увеличение длины ROB до 126 входов.
-Увеличение размеров регистров замещения.
-Увеличение размеров других буферов (BTB до 4096 и т.д.)
-Слияние микроопераций загрузки/выгрузки и обработки.

Планировщик в NetBurst



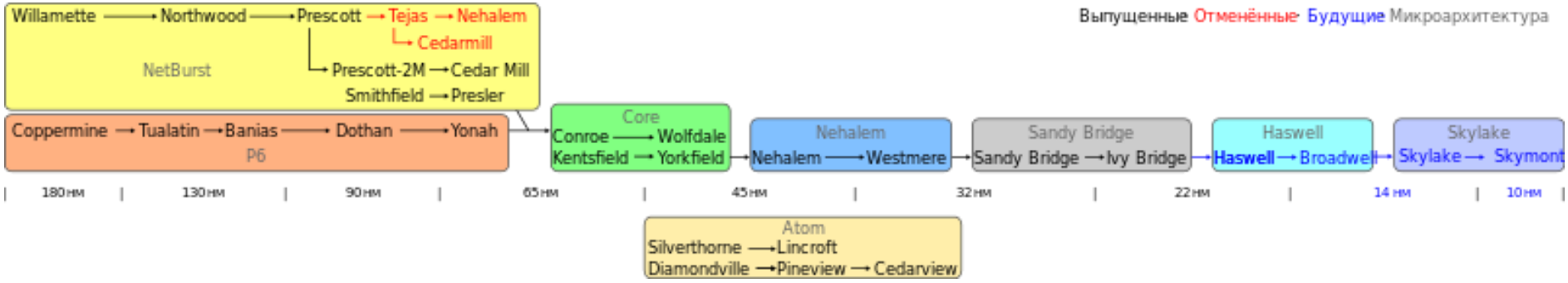
Особенности микроархитектуры NetBurst



Конвейеры микропроцессоров Intel

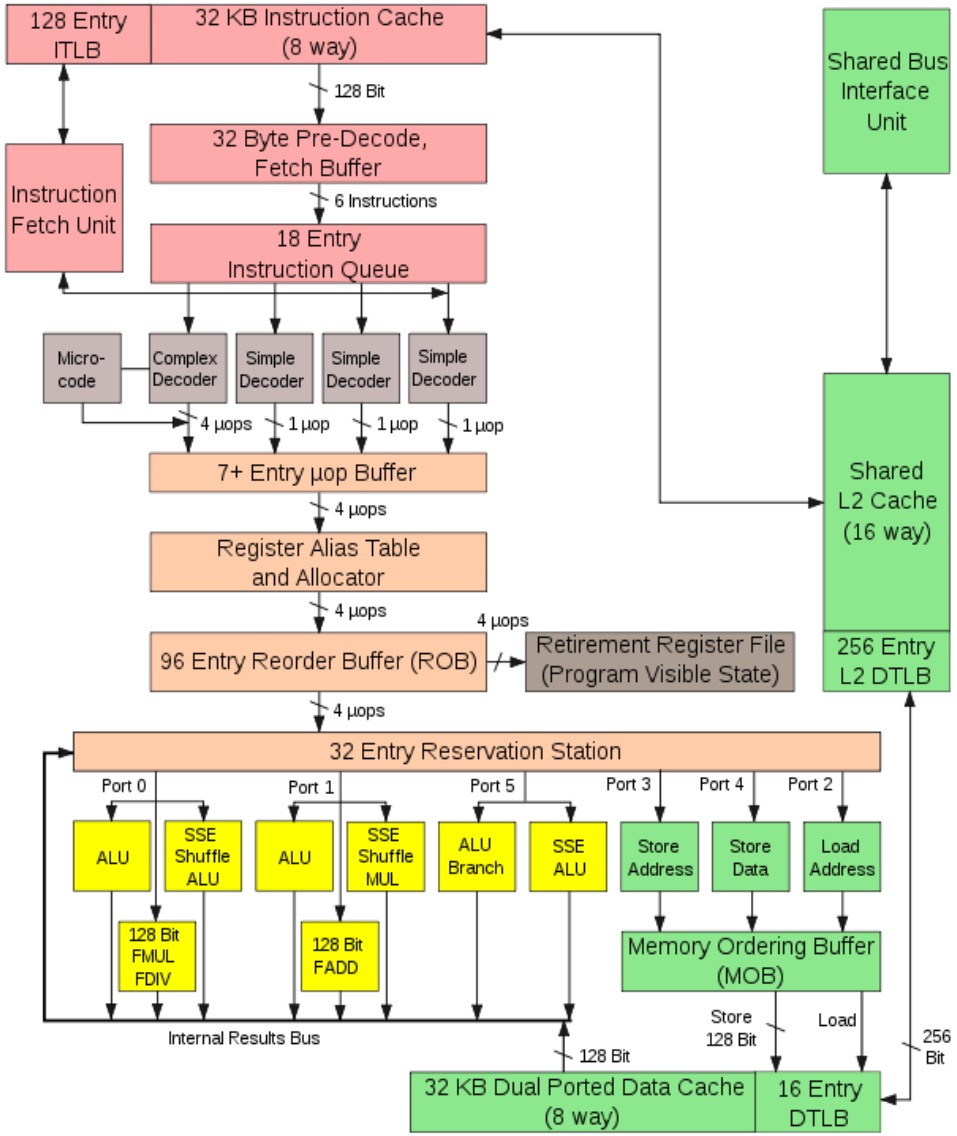
Микроархитектура	Количество стадий конвейера
486 (80486)	3
P5 (Pentium)	5
P6 (Pentium Pro/II)	14 (17 с загрузкой-выгрузкой и отставкой)
P6 (Pentium 3)	8 (11 с загрузкой-выгрузкой и отставкой)
P6 (Pentium M, Yonah)	10 (12 с выборкой и отставкой)
NetBurst (Willamette)	20
NetBurst (Northwood)	20
NetBurst (Prescott)	31
NetBurst (Cedar Mill)	31
Core (Merom/Conroe/Woodcrest)	12 (14 с выборкой и отставкой)
Nehalem	20
Sandy Bridge	14 (16 с выборкой и отставкой)

Микроархитектуры процессоров Intel



http://www.ixbt.com/cpu/sandy-bridge.shtml
https://en.wikipedia.org/wiki/List_of_Intel_CPU_microarchitectureshttps://ru.wikipedia.org/wiki/Sandy_Bridge

Микроархитектура Core (P6+)

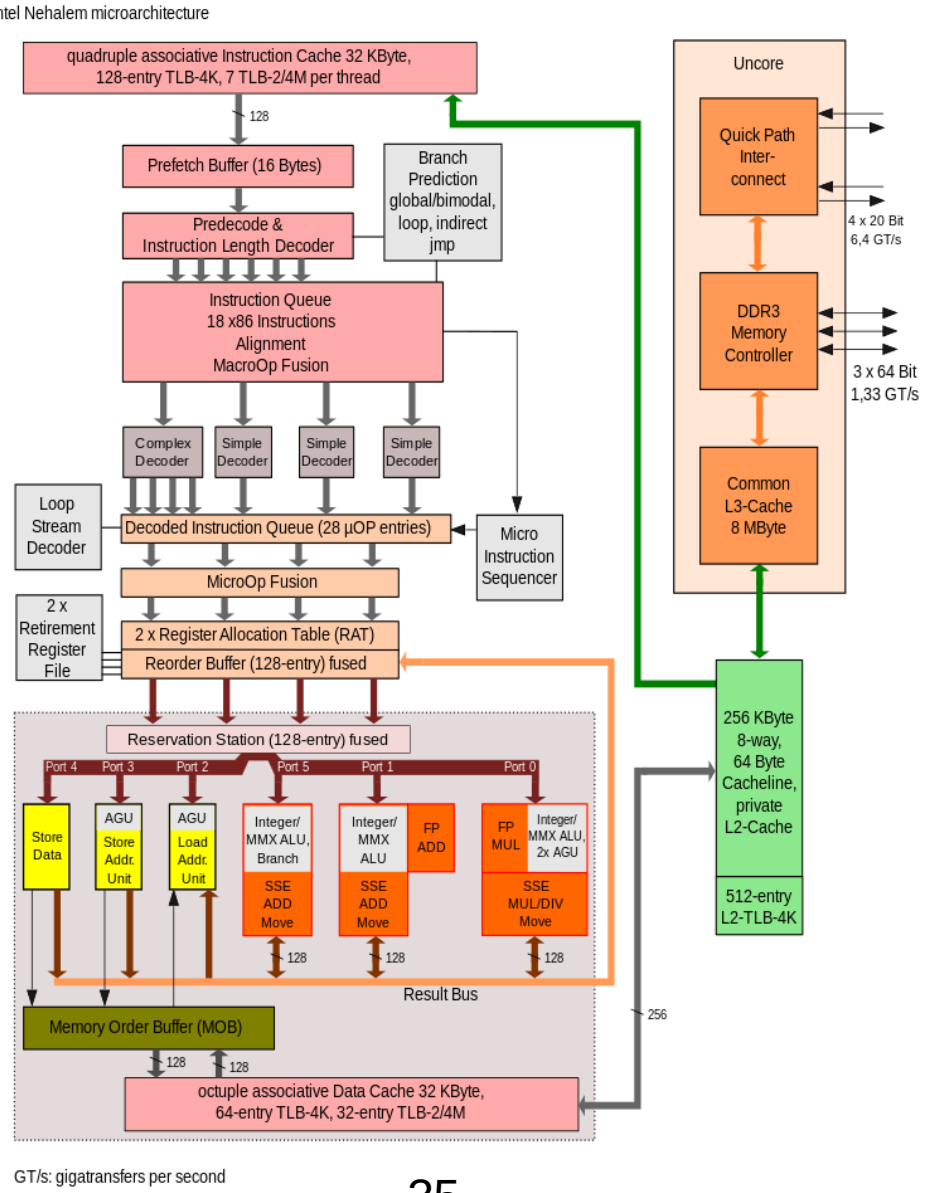


Intel Core 2 Architecture

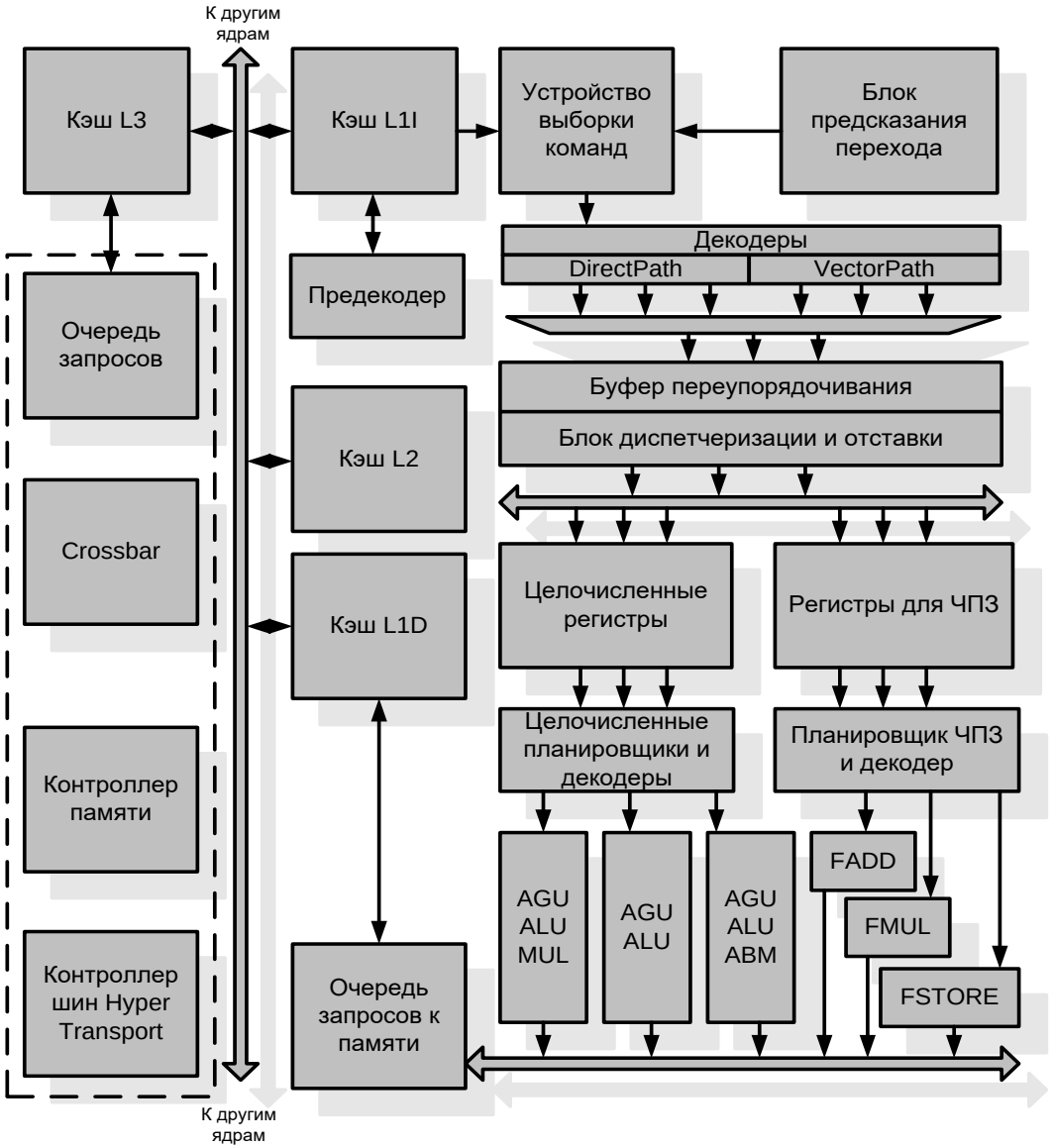
Микроархитектура Nehalem

Особенности микроархитектуры Nehalem:

- 32 KB data + 32 KB L1I (4 clocks) и 256 KB L2 кэш (11 clocks) на одно ядро.
- Разделяемая L3 кэш-память, доступная для графического ядра.
- 64-байта размер кэш-линейки.
- Выполнение до двух команд загрузки/выгрузки за один такт для каждого канала памяти
- Кэш для хранения декодированных микрокоманд (uop cache) и более совершенны блок предсказания направления ветвления.
- Повышенная производительность для математических функций, AES кодирования (AES instruction set), и SHA-1 хэширования.
- 256-битная шина с топологией кольца для связи между ядрами графическим ядром, кэш и System Agent Domain (Advanced Vector Extensions).
- Advanced Vector Extensions (AVX) длина вектора расширена до 256 бит, добавлены новые команды и расширен синтаксис.
- Intel Quick Sync Video - аппаратная поддержка кодирования/декодирования видео.
- До 8 физических ядер (16 логических ядер при Hyper-threading) на кристалл.
- Интеграция GMCH (integrated graphics and memory controller) и процессоров на одном кристалле.
- От 14 до 19 ступеней конвейеров команд (в зависимости от промахов в кэш).

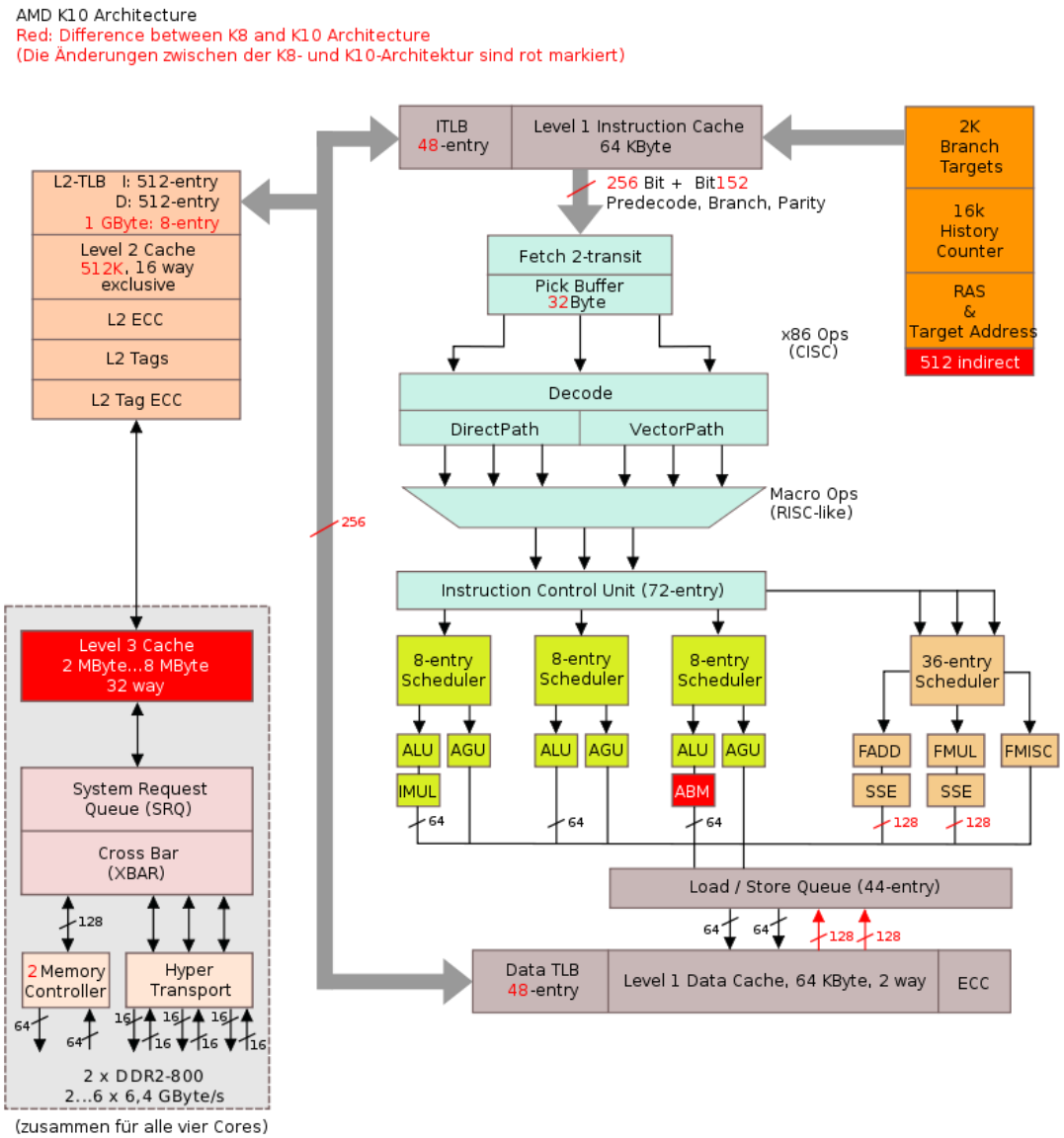


Микроархитектура AMD64

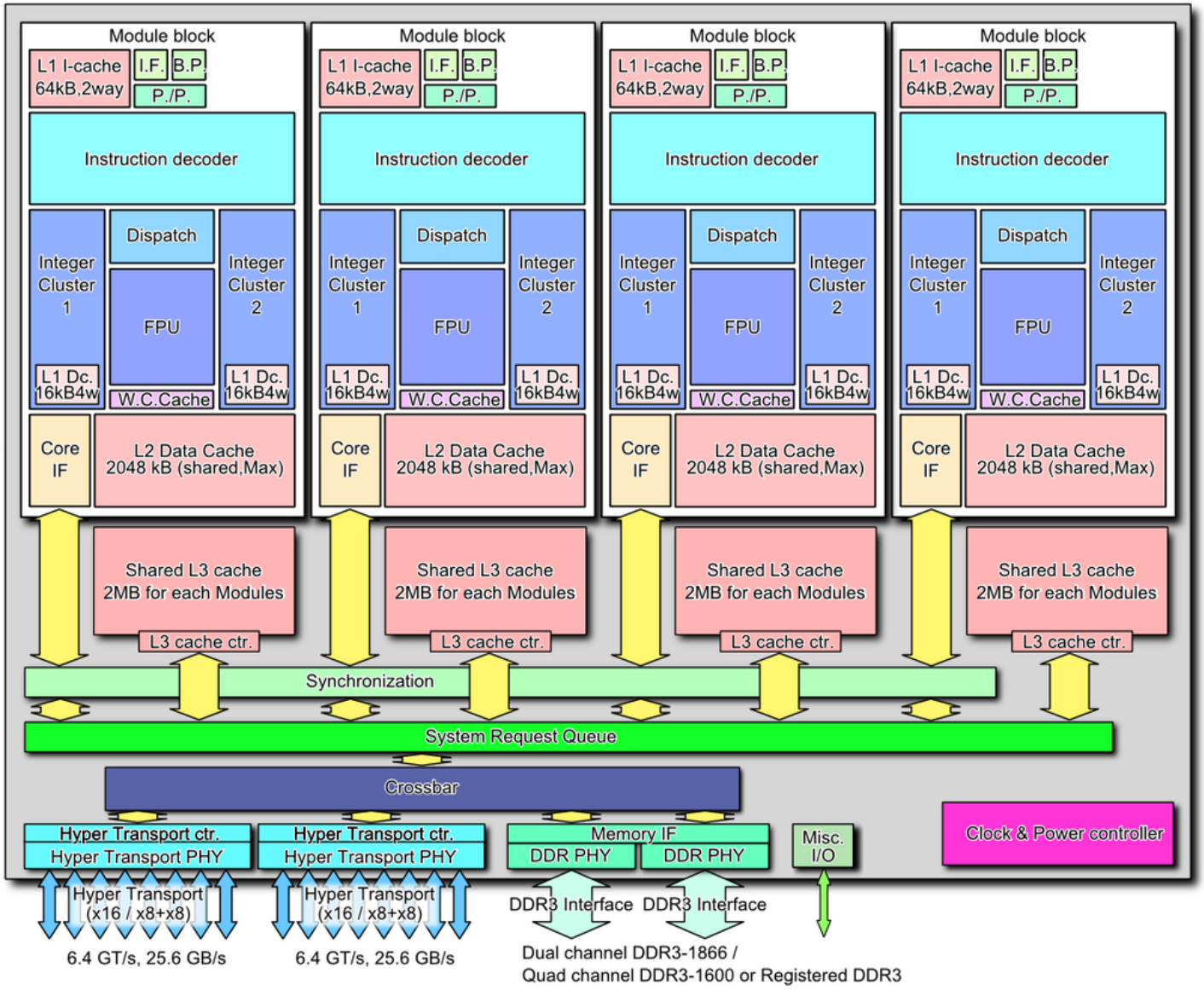


https://en.wikipedia.org/wiki/List_of_AMD_CPU_microarchitectures

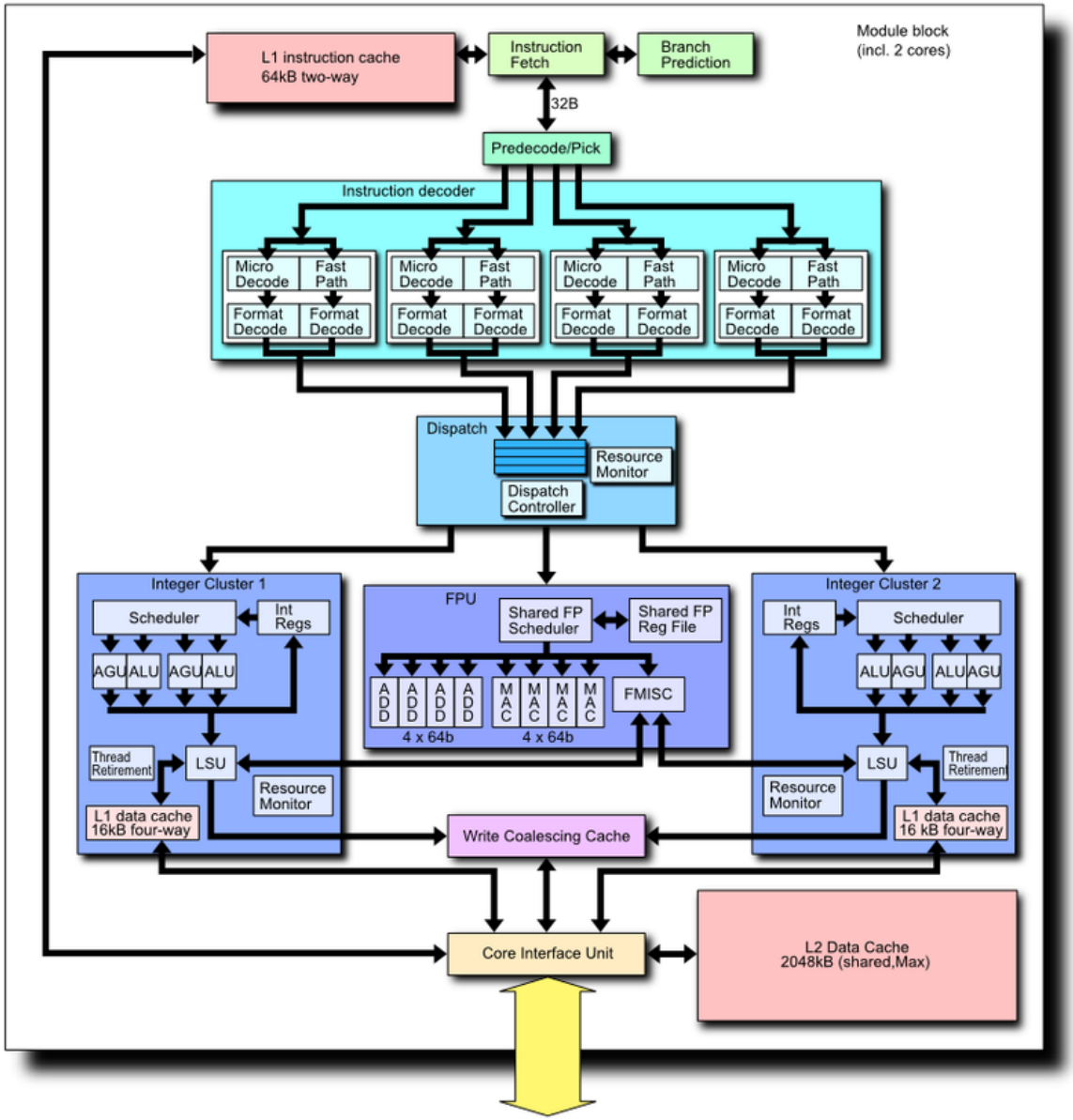
Микроархитектура AMD K10



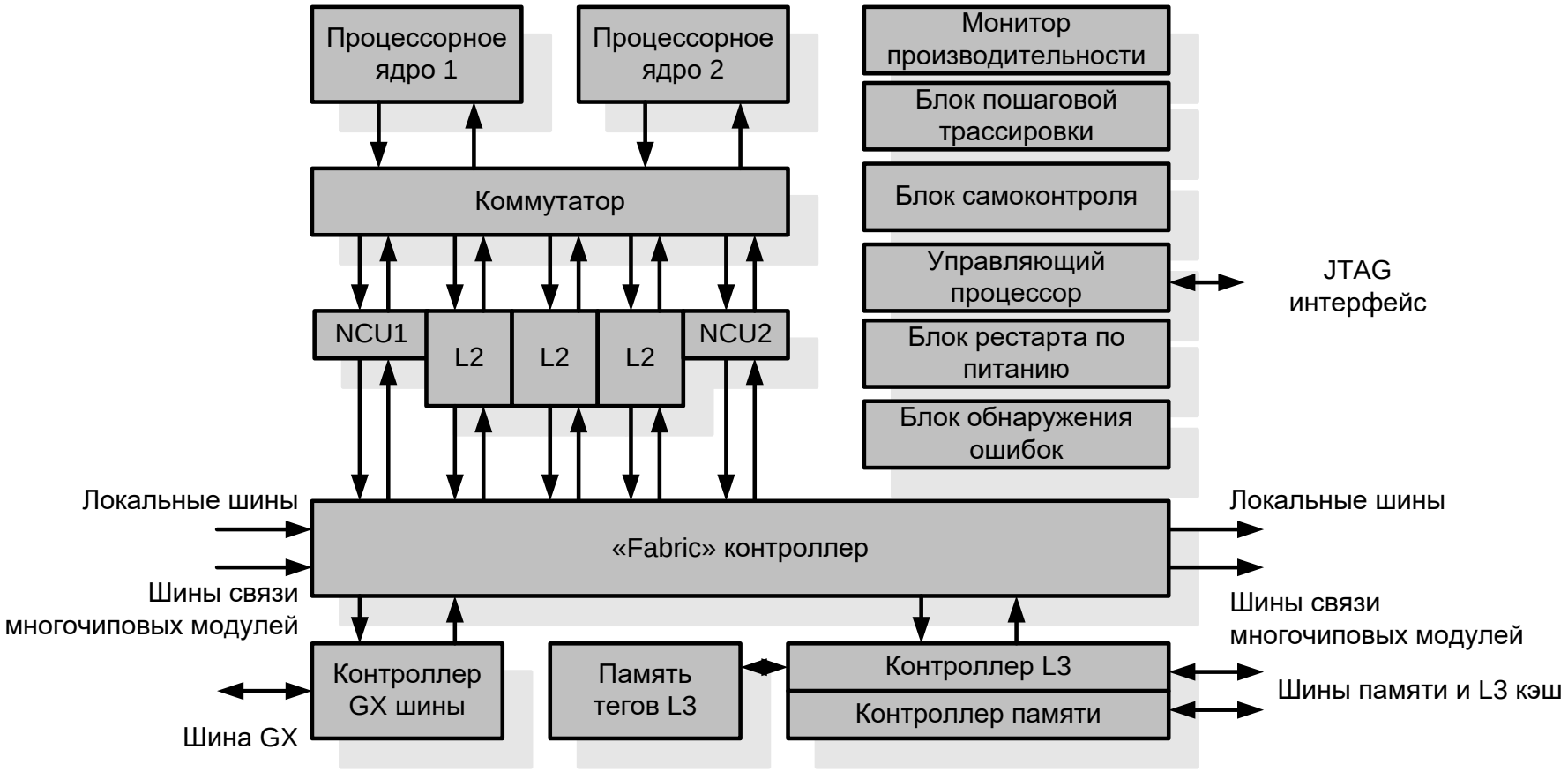
Многоядерный чип AMD Bulldozer



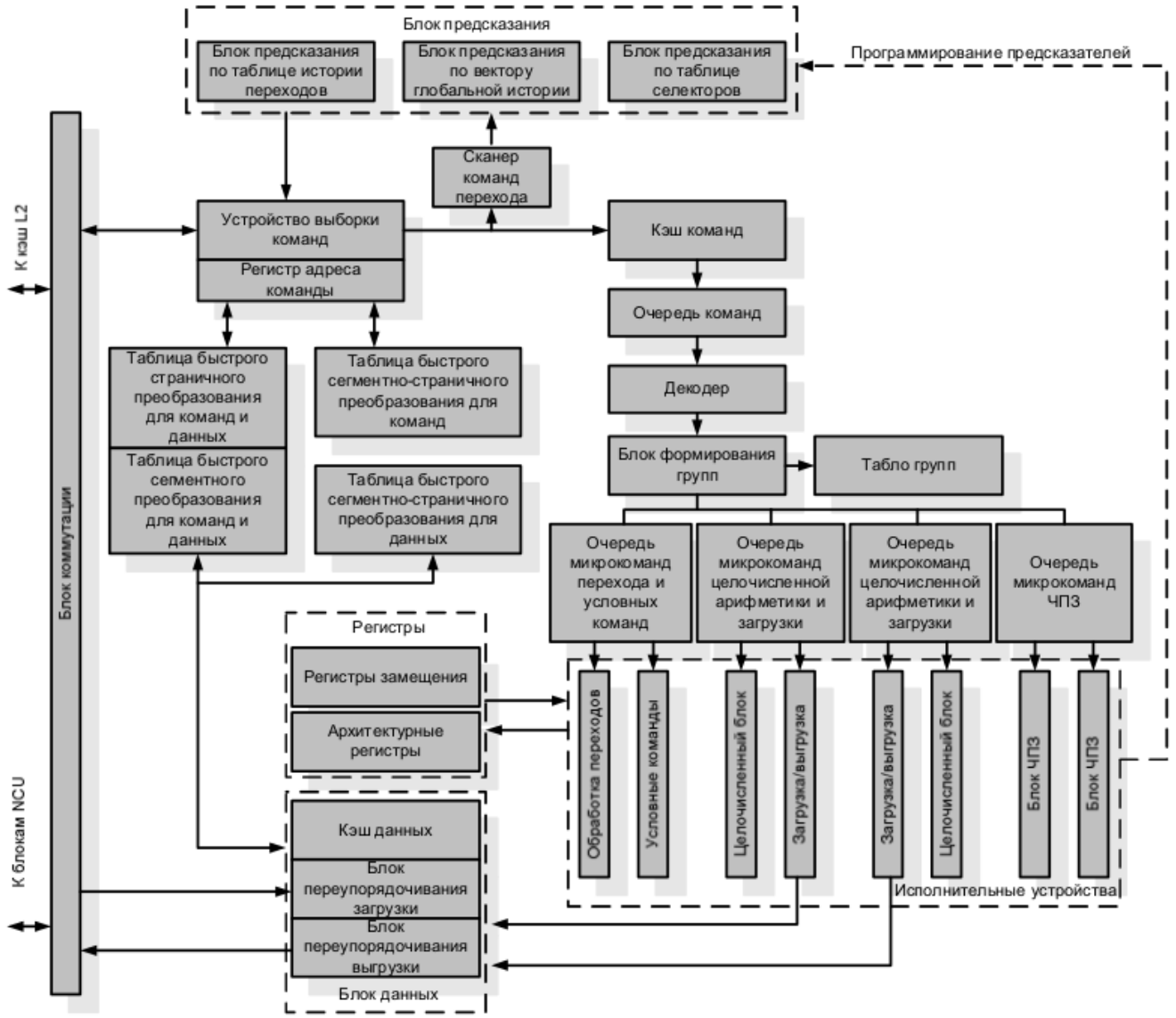
Микроархитектура ядра AMD Bulldozer



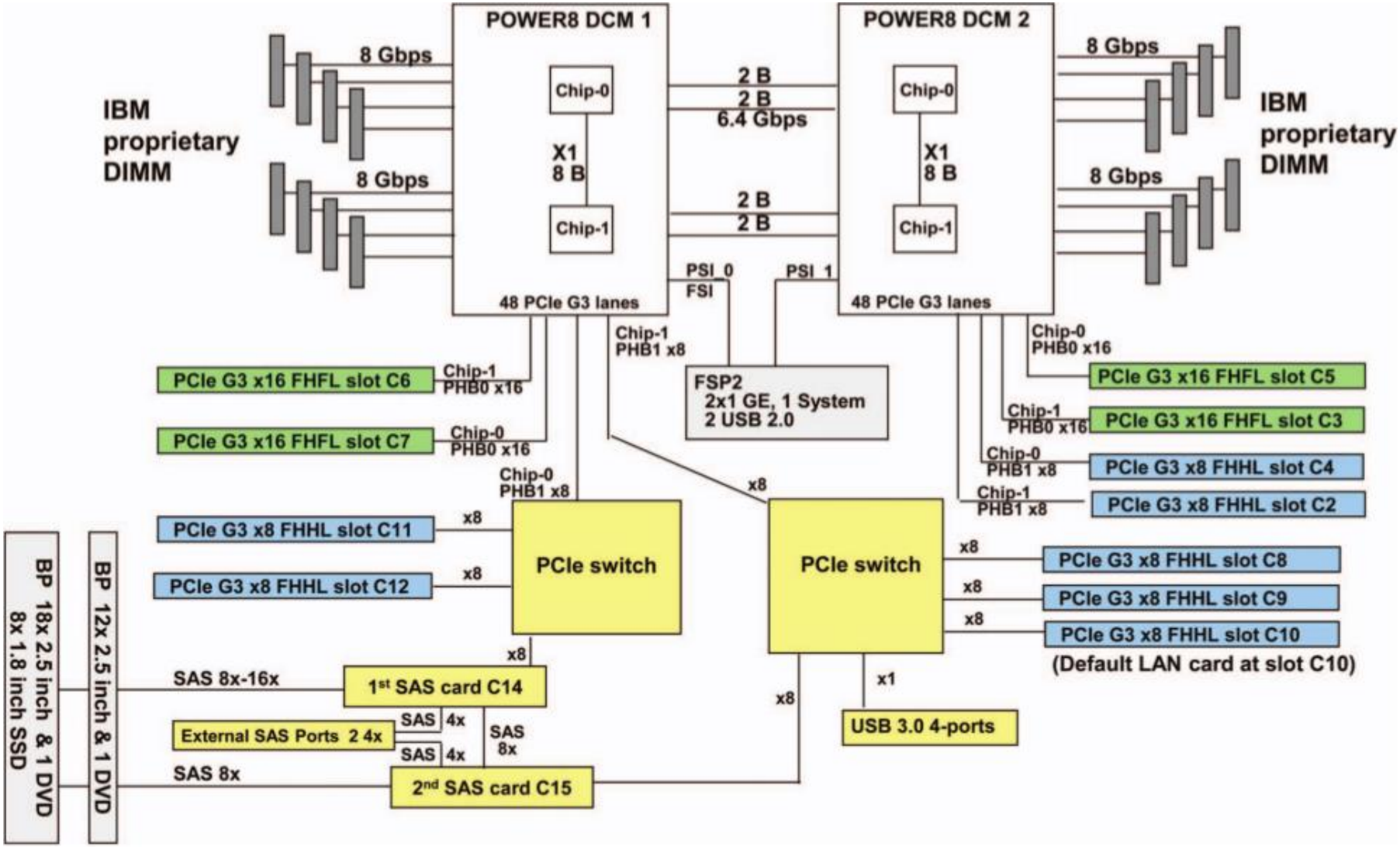
Микроархитектура суперскалярных процессоров IBM POWER4



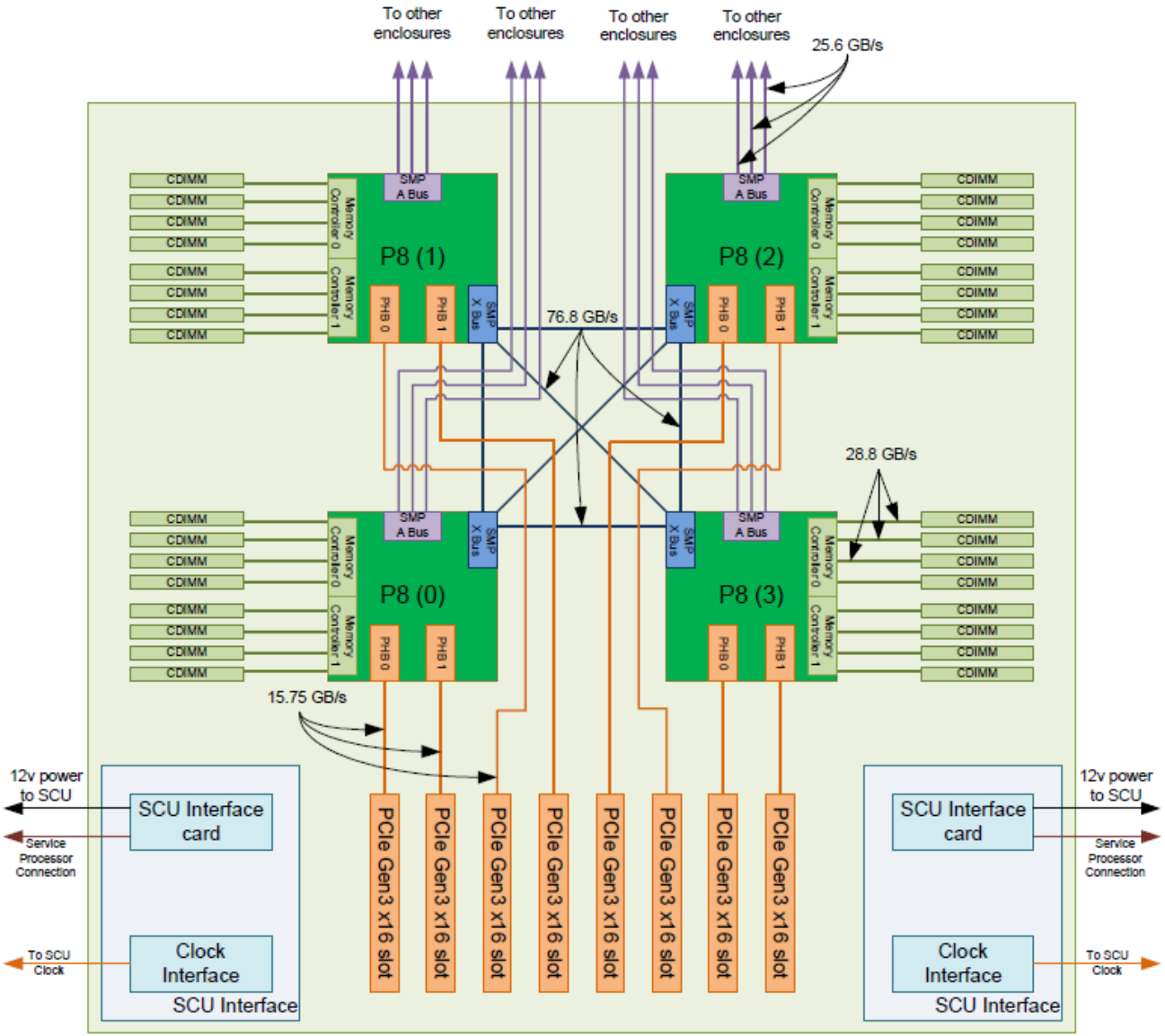
Структура процессорного ядра IBM POWER4



Серверная платформа IBM POWER8

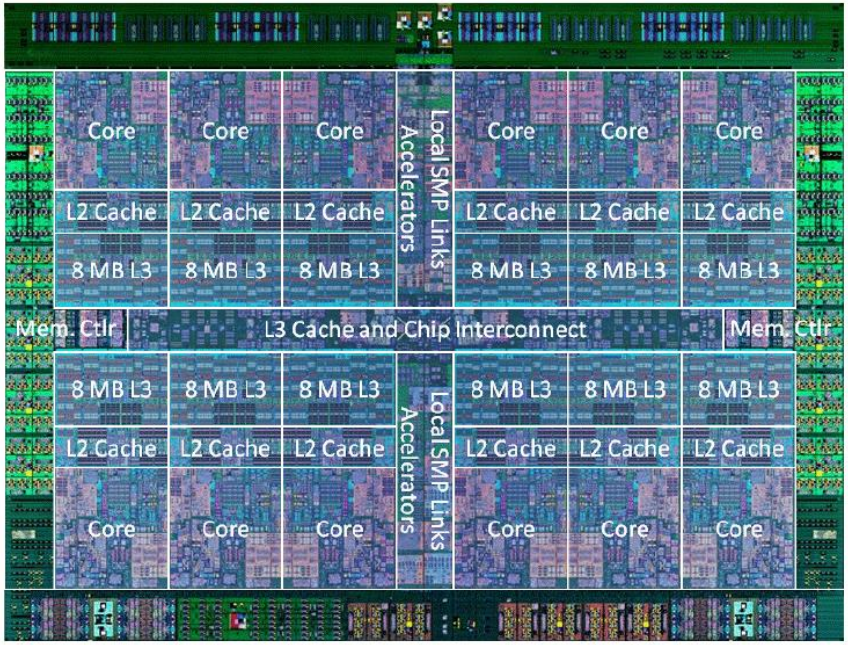


Вычислительный узел серверной платформы IBM POWER8

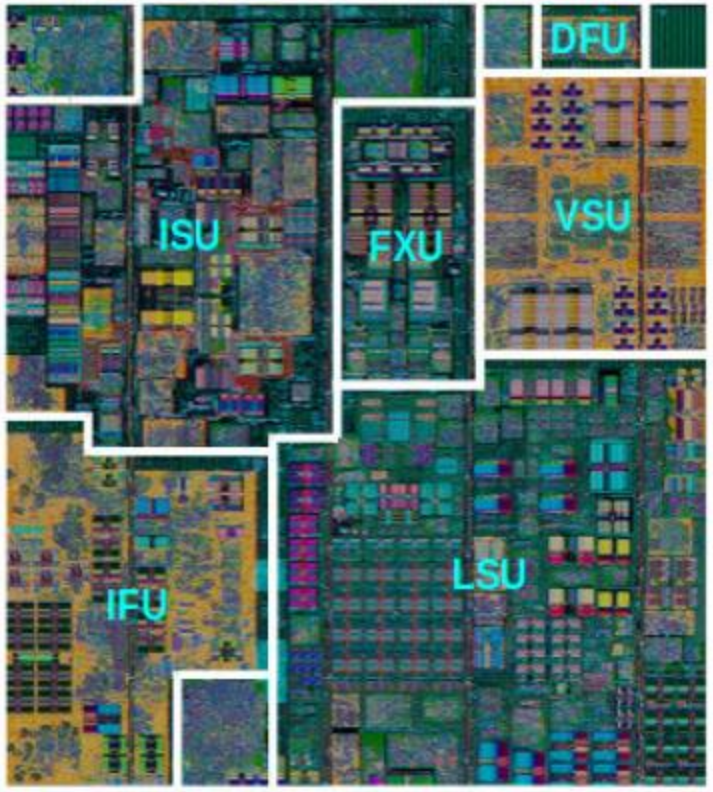


Структура процессорного ядра IBM POWER8

Микросхема POWER8

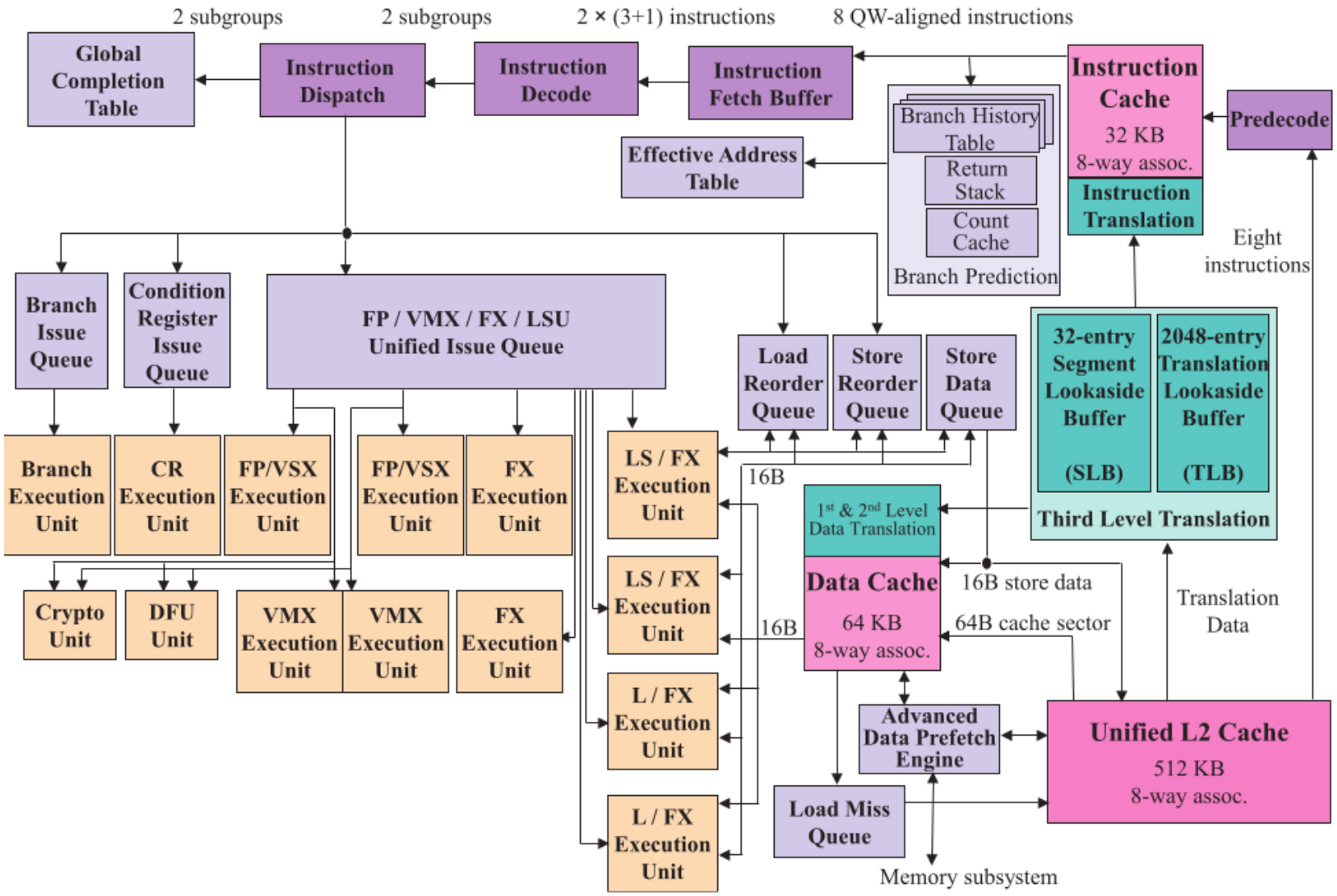


Микропроцессорное ядро

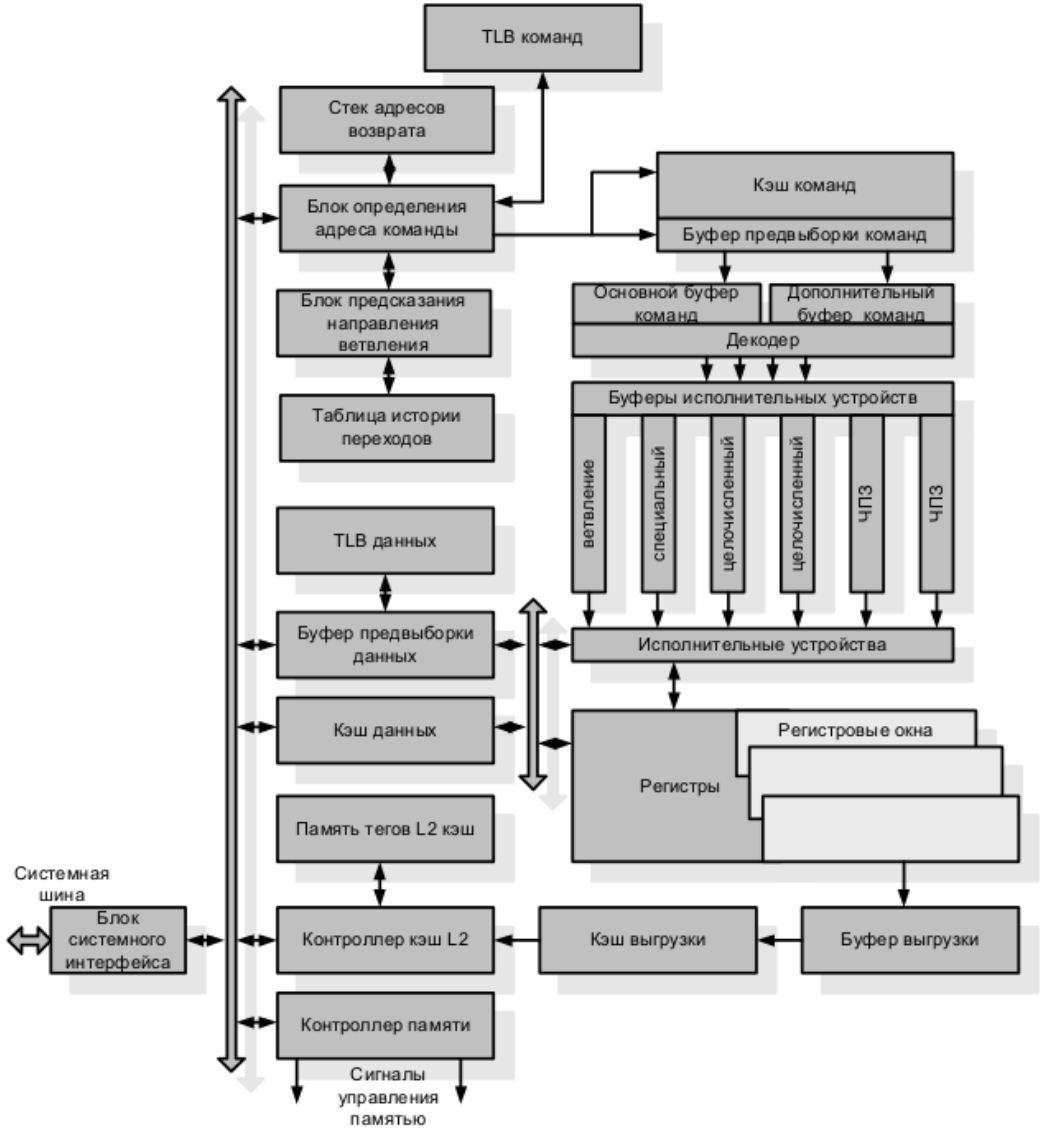


Структура процессорного ядра IBM POWER8

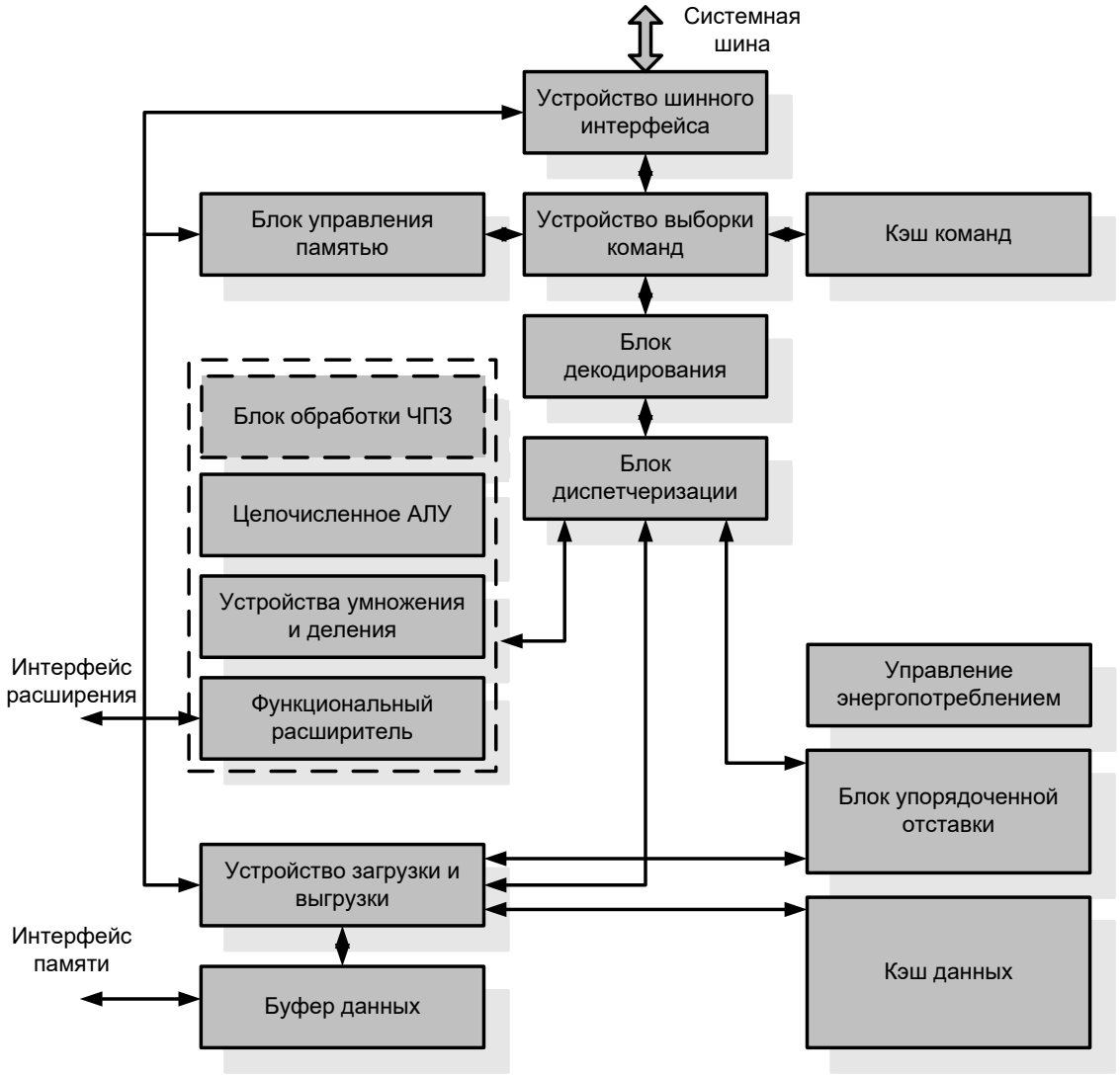
Конвейер



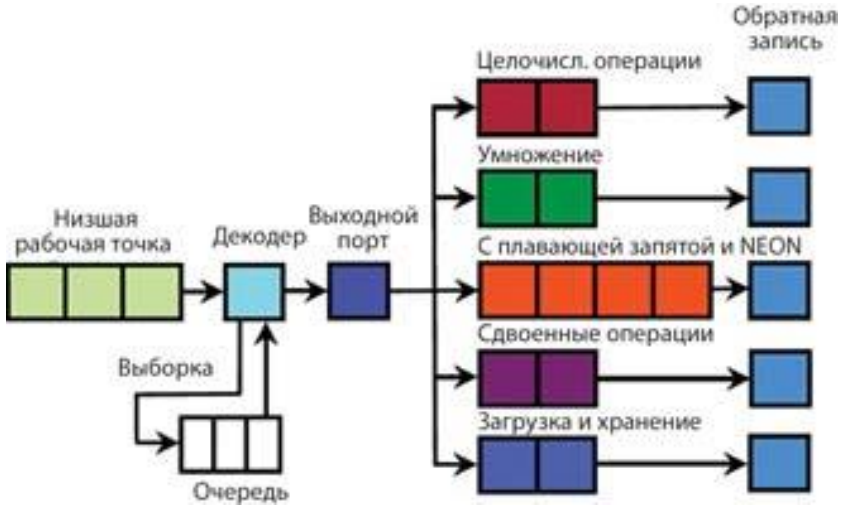
Микроархитектура суперскалярных процессоров Sun UltraSPARC III



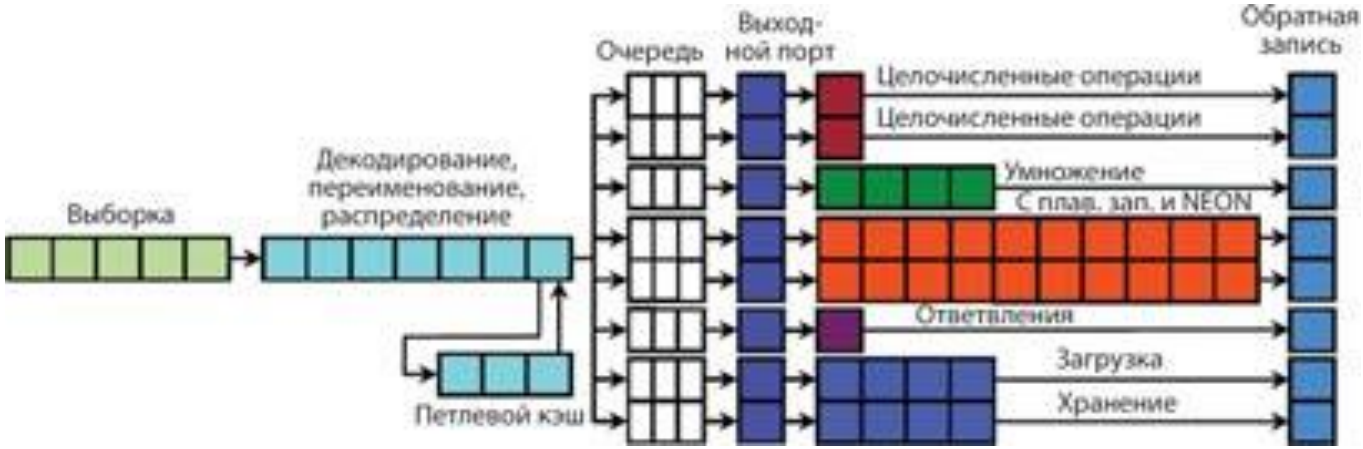
Архитектура синтезируемых суперскалярных процессорных ядер MIPS32 74K



Микроархитектура Cortex-A7

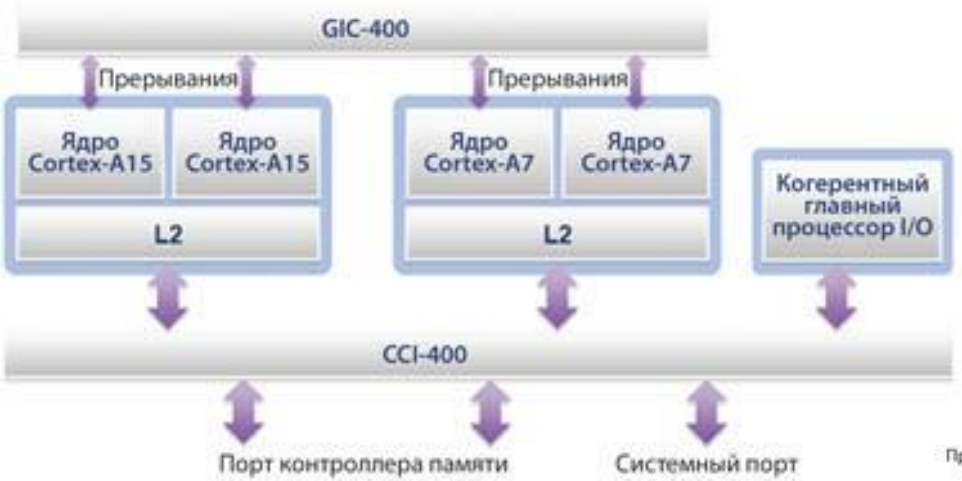


Микроархитектура Cortex-A15

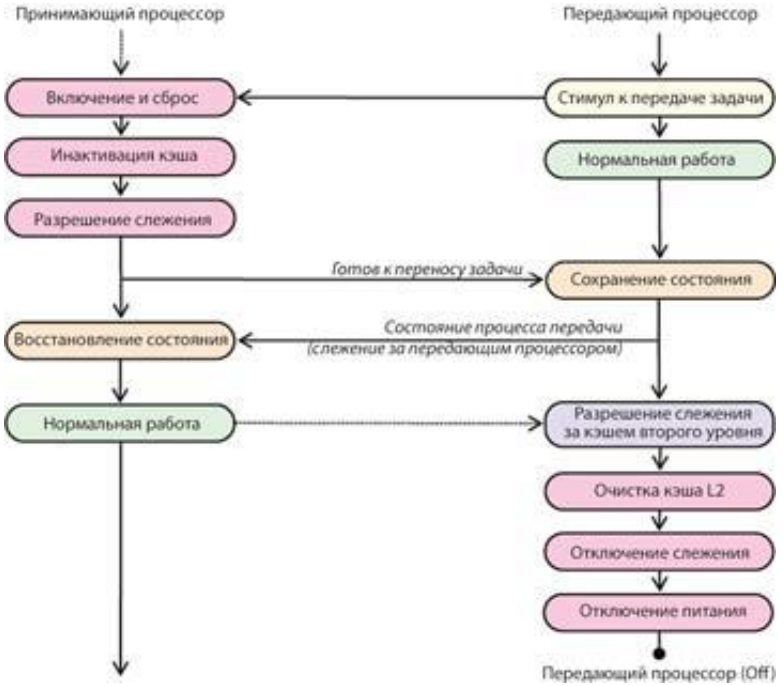


Greenhalgh P. Big.LITTLE processing with ARM Cortex-A15 & Cortex-A7//www.eetimes.com.

big.LITTLE

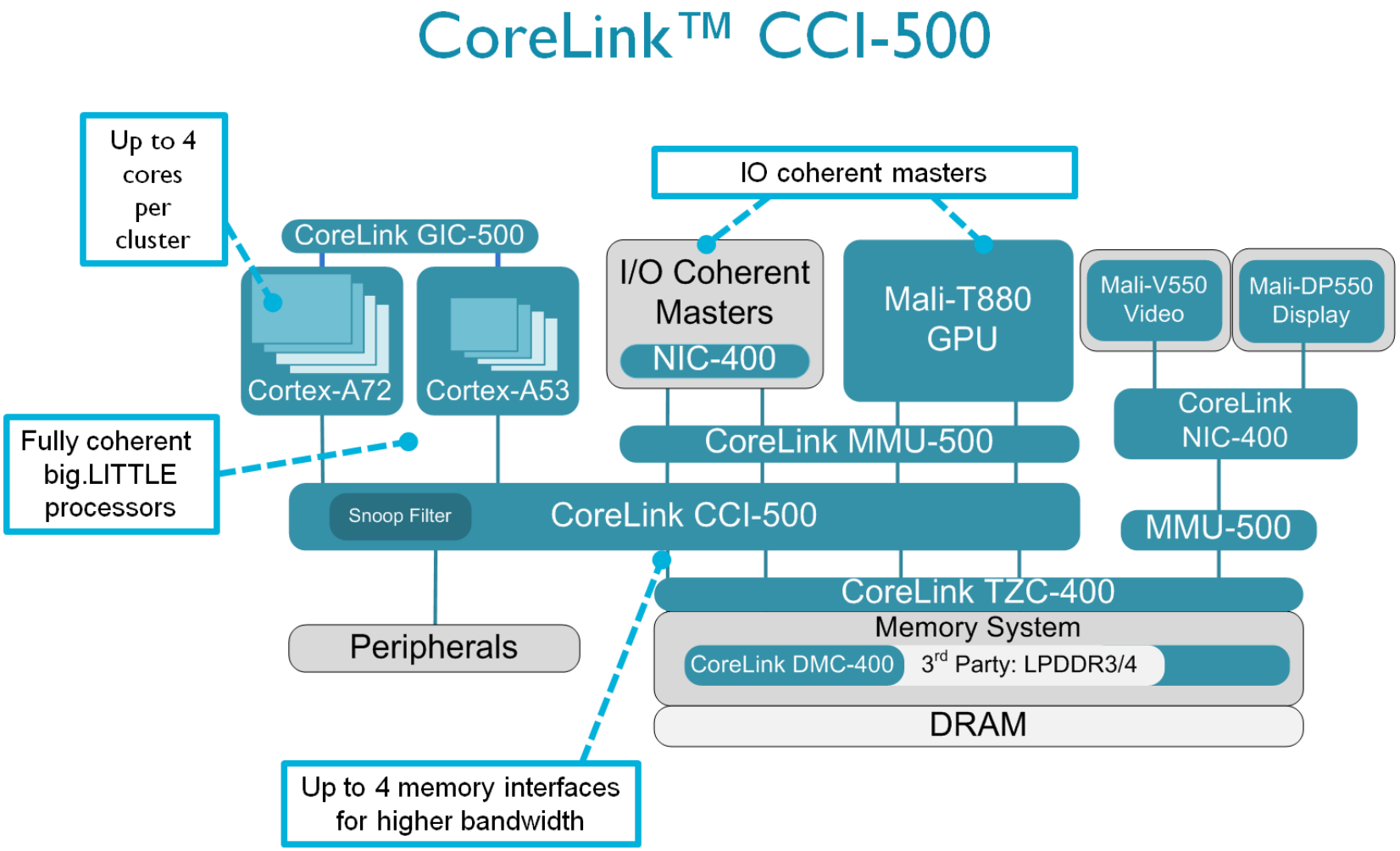


Одним из ключевых элементов является система межсоединений CCI-400, которая обеспечивает полную согласованность между Cortex-A15 и Cortex-A7, а также устройствами ввода-вывода. Контроллер прерываний GIC-400 способен распределить до 480 прерываний, причем прерывания могут направляться в любое ядро в кластере Cortex-A15 или Cortex-A7. big.LITTLE распределяет задачи операционной системы (ОС) и приложений так, чтобы они выполнялись только на одном процессоре в каждый момент времени

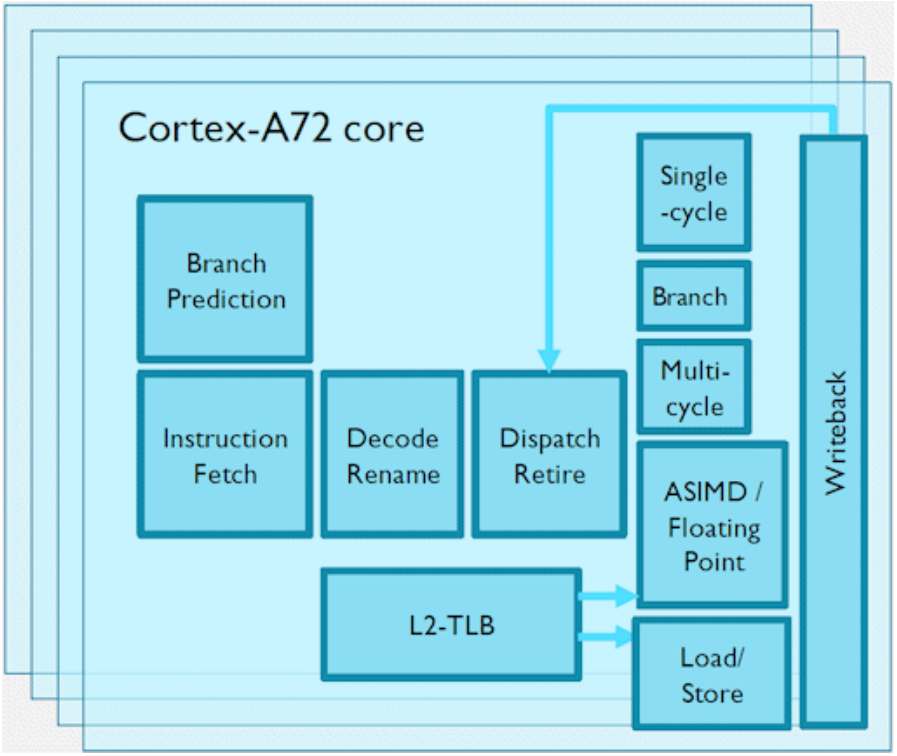


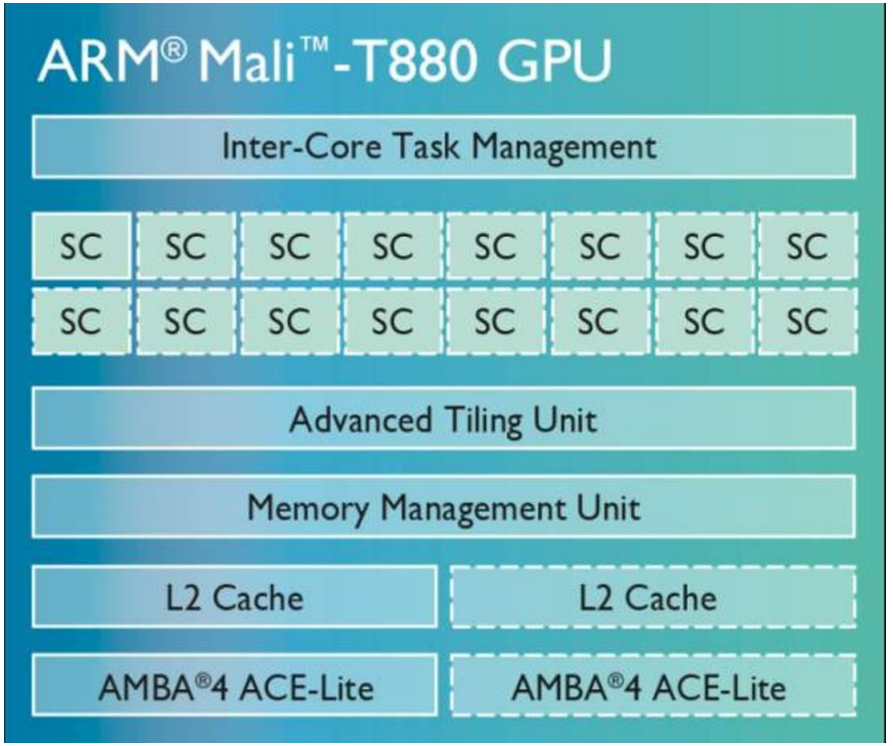
http://www.russianelectronics.ru/leader-r/review/2192/doc/58808/

Архитектура системы на кристалле Cortex-A72

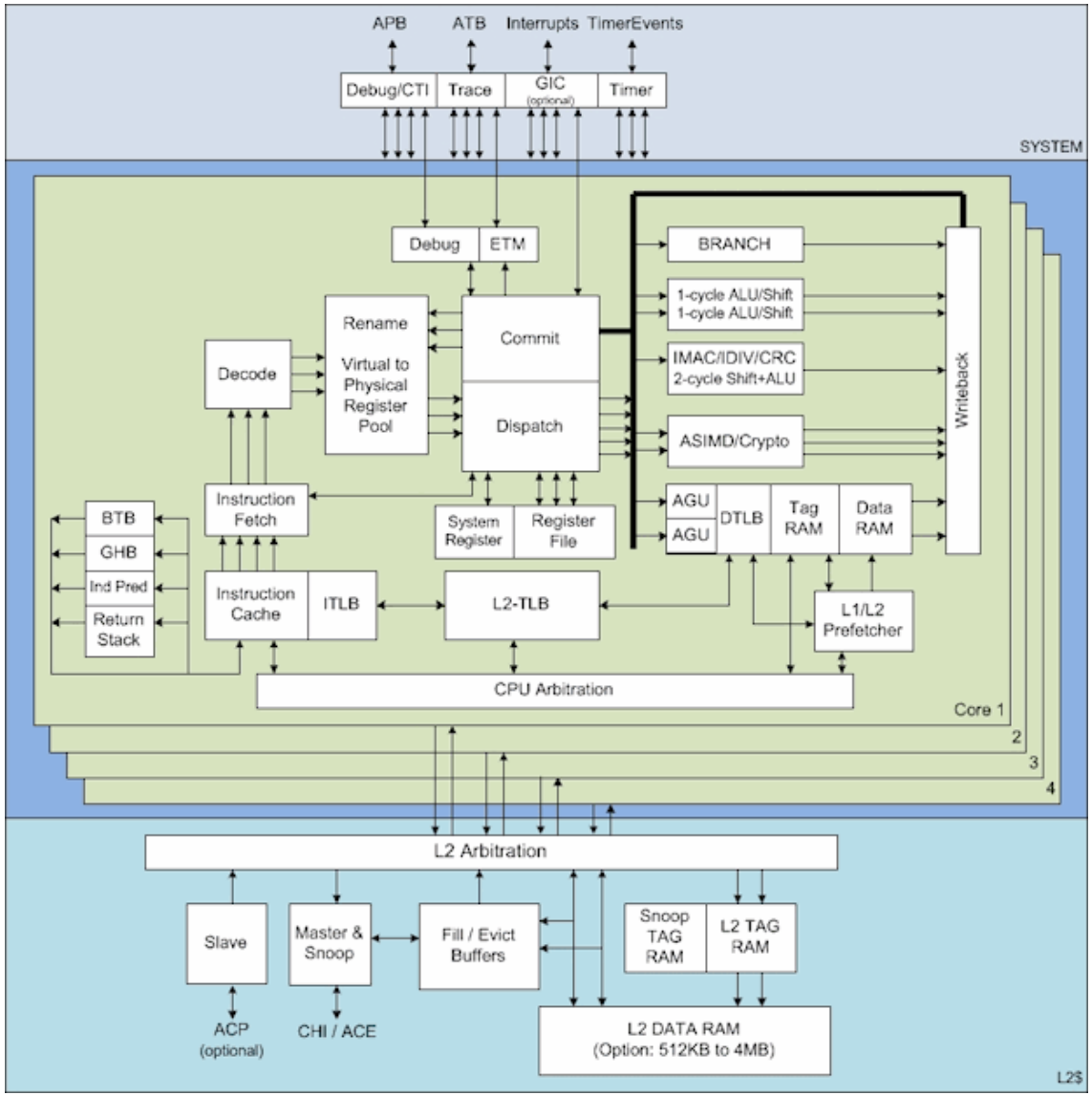


Микропроцессор Cortex-A72



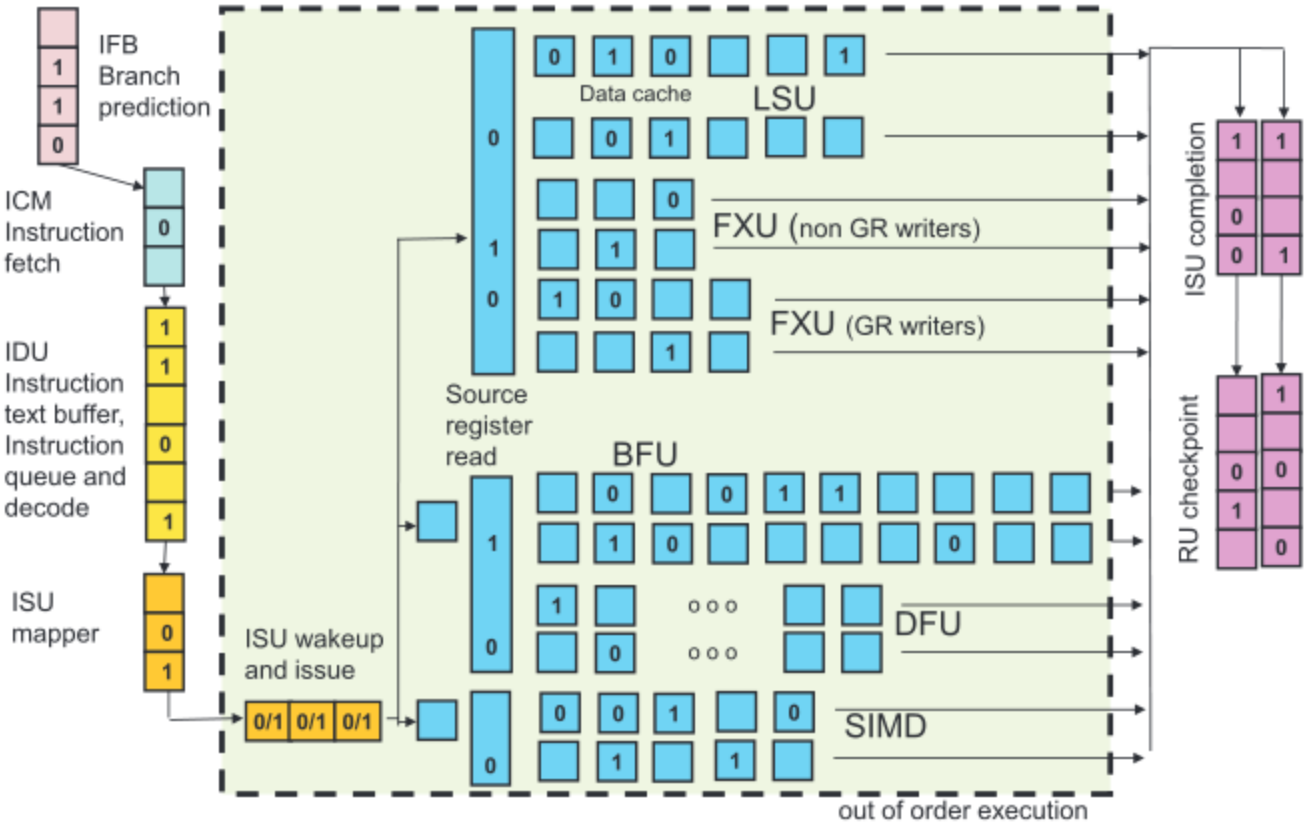


Микроархитектура Cortex-A72



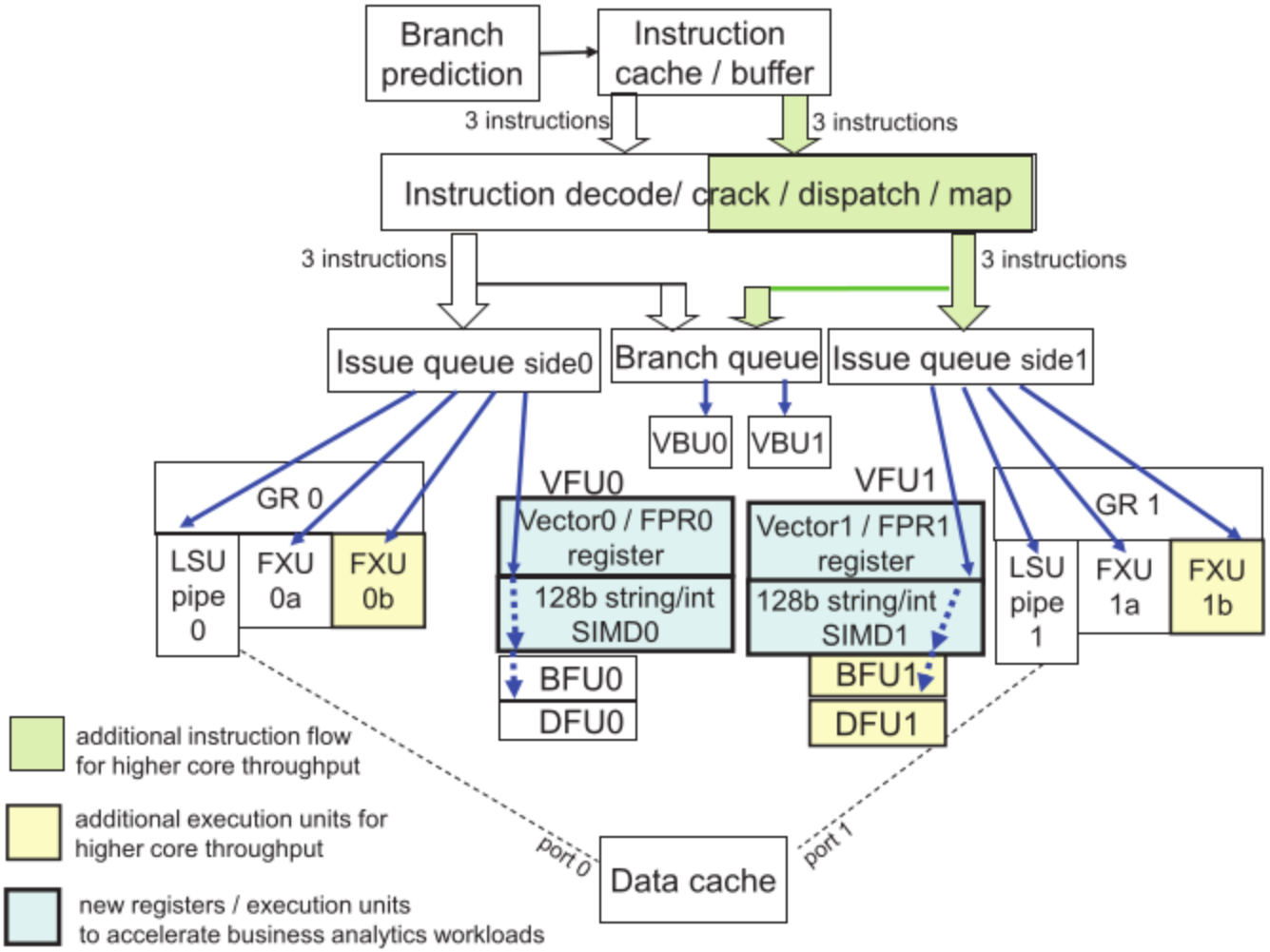
Микроархитектура IBM System z13

Конвейер



Микроархитектура IBM System z13

Стадии обработки команд



Устройства управления: классификация

По типу автомата:

- Автомат Мили.
- Автомат Мура.

По способу реализации:

- Устройство управления с жесткой логикой.

  Функции выдачи сигналов управления и разделения во времени сигналов управления реализуются с помощью комбинационных схем и триггерной памяти.

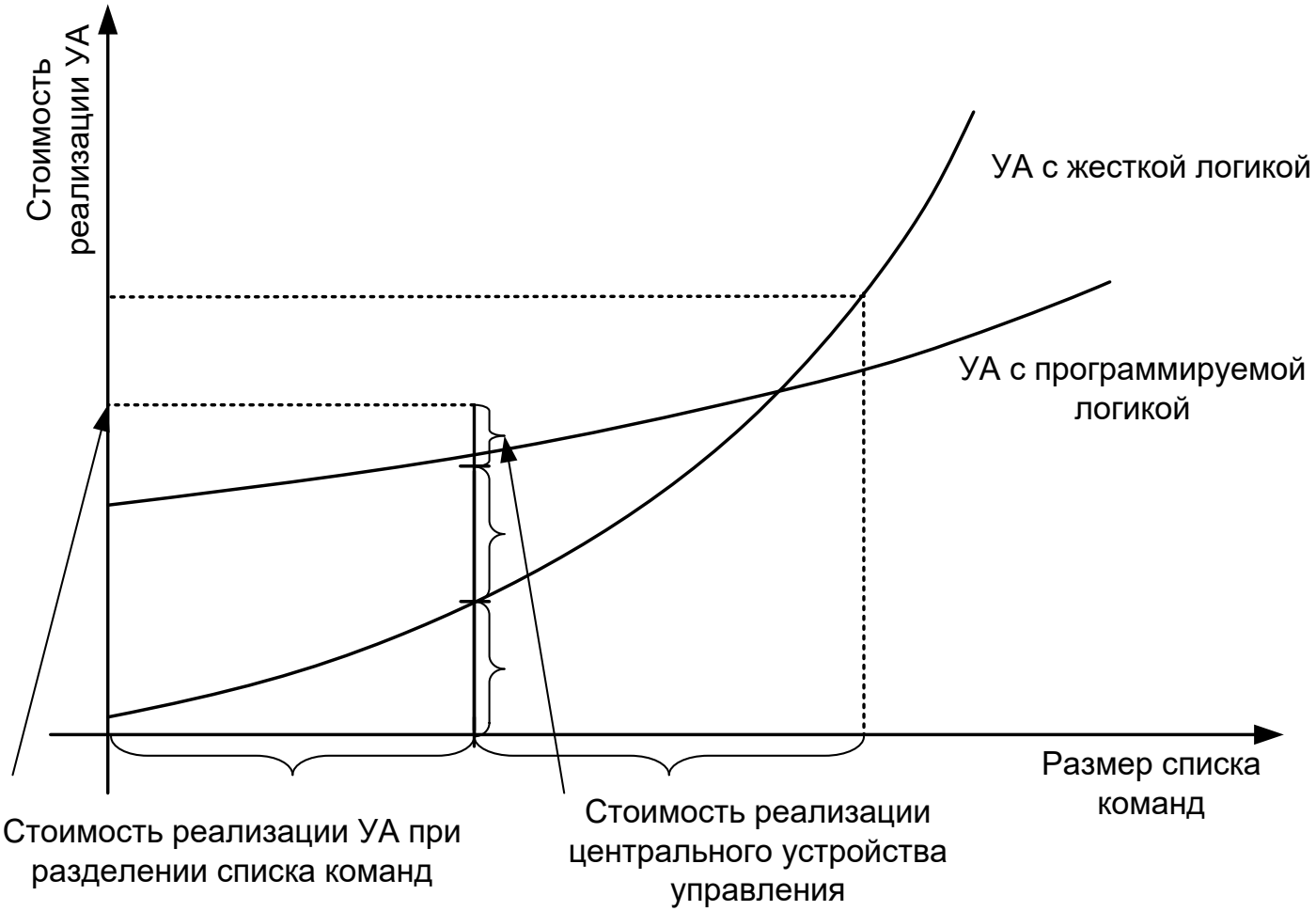
- Устройство управления с программируемой логикой.

  Каждой выполняемой операции ставится в соответствие совокупность хранимых в памяти слов (микрокоманд), каждая из которых содержит информацию о микрооперациях, подлежащих исполнению в текущем такте.

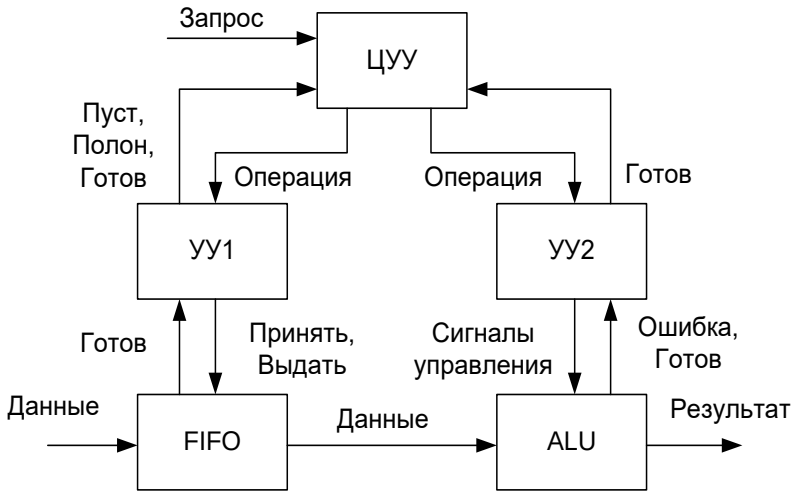
  (+) Простота модификации и наращивания.

  (-) Невысокое быстродействие для простых устройств.

Сравнение способов реализации УУ



Пример декомпозиции УУ



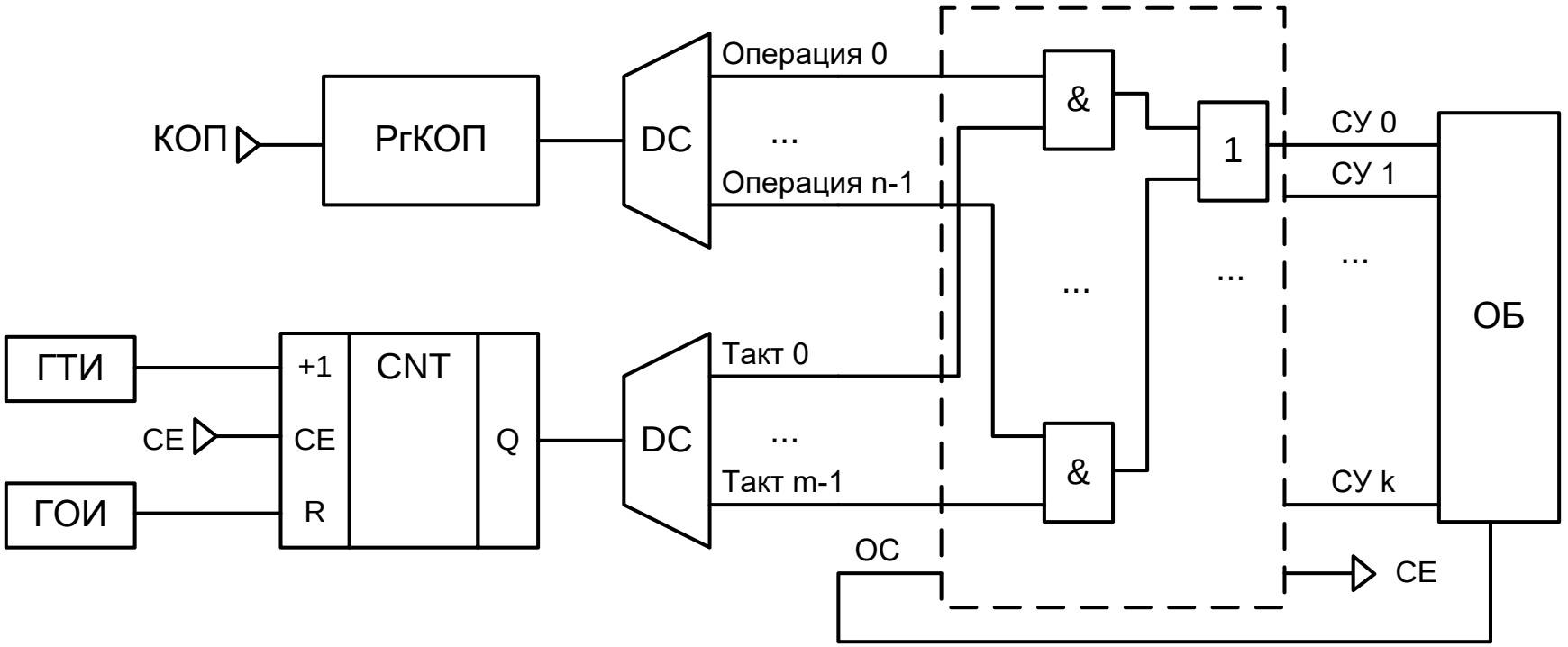
По способу кодирования микрокоманд:

- Минимальное кодирование (горизонтальное).
- Максимальное кодирование (вертикальное).
- Горизонтально-вертикальное кодирование.
- Вертикально-горизонтальное кодирование.
- Кодирование с помощью памяти нанокоманд.

По способу исполнения команд:

- Последовательные.
- Конвейерные.

Пример реализации управляющего автомата с жесткой логикой



Синтез управляющих автоматов с жесткой логикой

Исходные данные:

-Описание логики функционирования операционного блока (временные диаграммы, словесное описание, таблицы истинности, графы и т.д.).
-Сигналы управления.
-Осведомительные сигналы.
-Временные параметры.

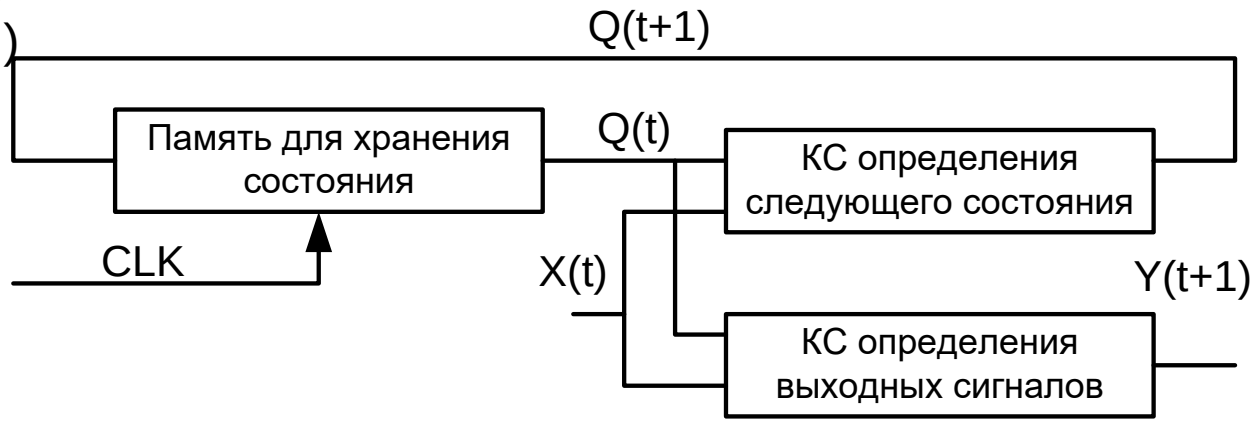
Процедура синтеза:

-Определение алгоритма функционирования управляющего автомата.
-Определение состояний с учетом выбранного типа автомата (Мили или Мура).
-Кодирование состояний автомата.
-Определение логических функций для сигналов управления и их минимизация.
-Определение логических функций перехода

Автомат Мили

$$\begin{cases} Q(t+1) = A(Q(t), x(t)) \\ Y(t+1) = B(Q(t), x(t)) \end{cases}$$

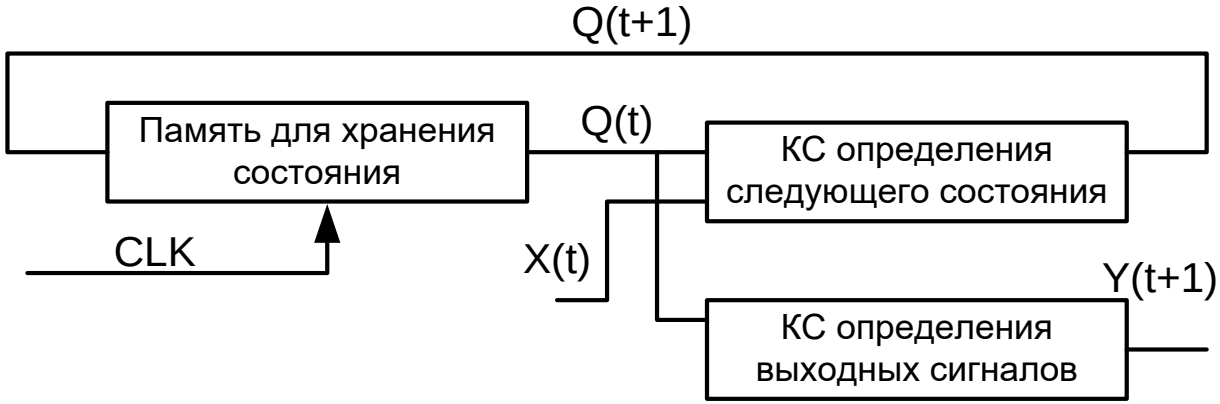
Схема автомата Мили



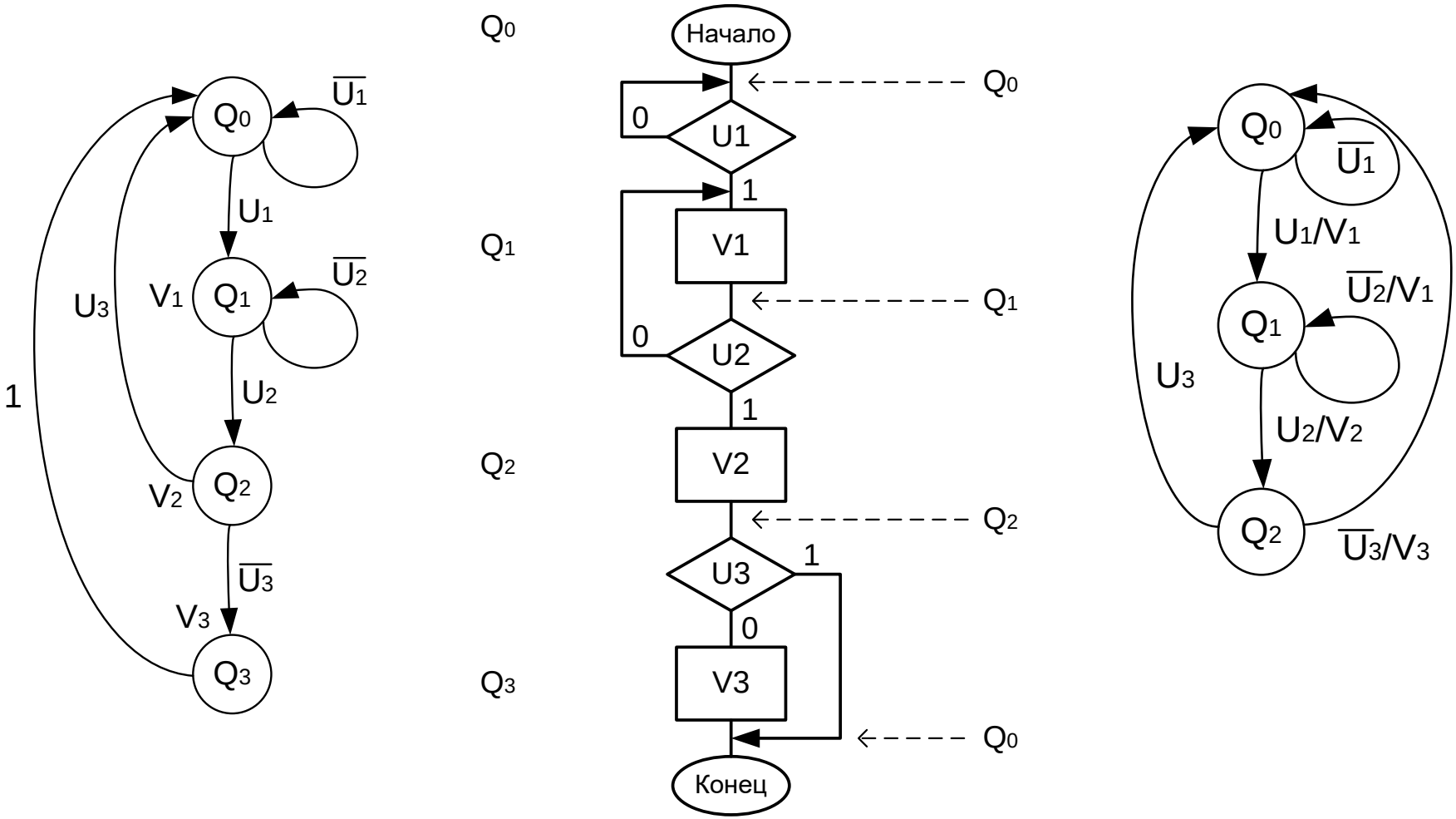
Автомат Мура

$$\begin{cases} Q(t+1) = A(Q(t), x(t)) \\ Y(t+1) = B(Q(t)) \end{cases}$$

Схема автомата Мура



Пример синтеза УУ с ЖЛ на основе автоматов Мили и Мура



Автомат Мура

$V_i = U\ Q_{Vi}$, где Q_{Vi} – состояния автомата, в которых сигнал V_i активен.

$V_1 = Q_1$; $V_2 = Q_2$; $V_3 = Q_3$;

$Q_0 = Q_0!U_1\ U\ Q_2U_3\ U\ Q_3$;

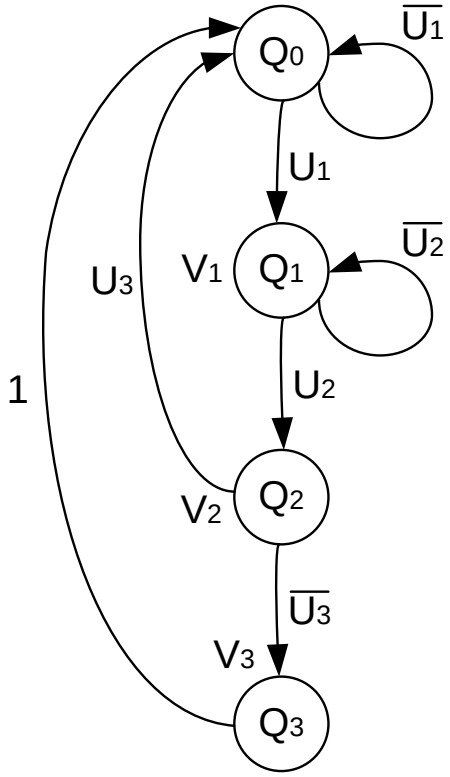
$Q_1 = Q_1!U_2\ U\ Q_0U_1$

$Q_2 = Q_1U_2$

$Q_3 = Q_2!U_3$

Таблица кодирования состояний

	D1	D0
Q0	0	0
Q1	0	1
Q2	1	0
Q3	1	1

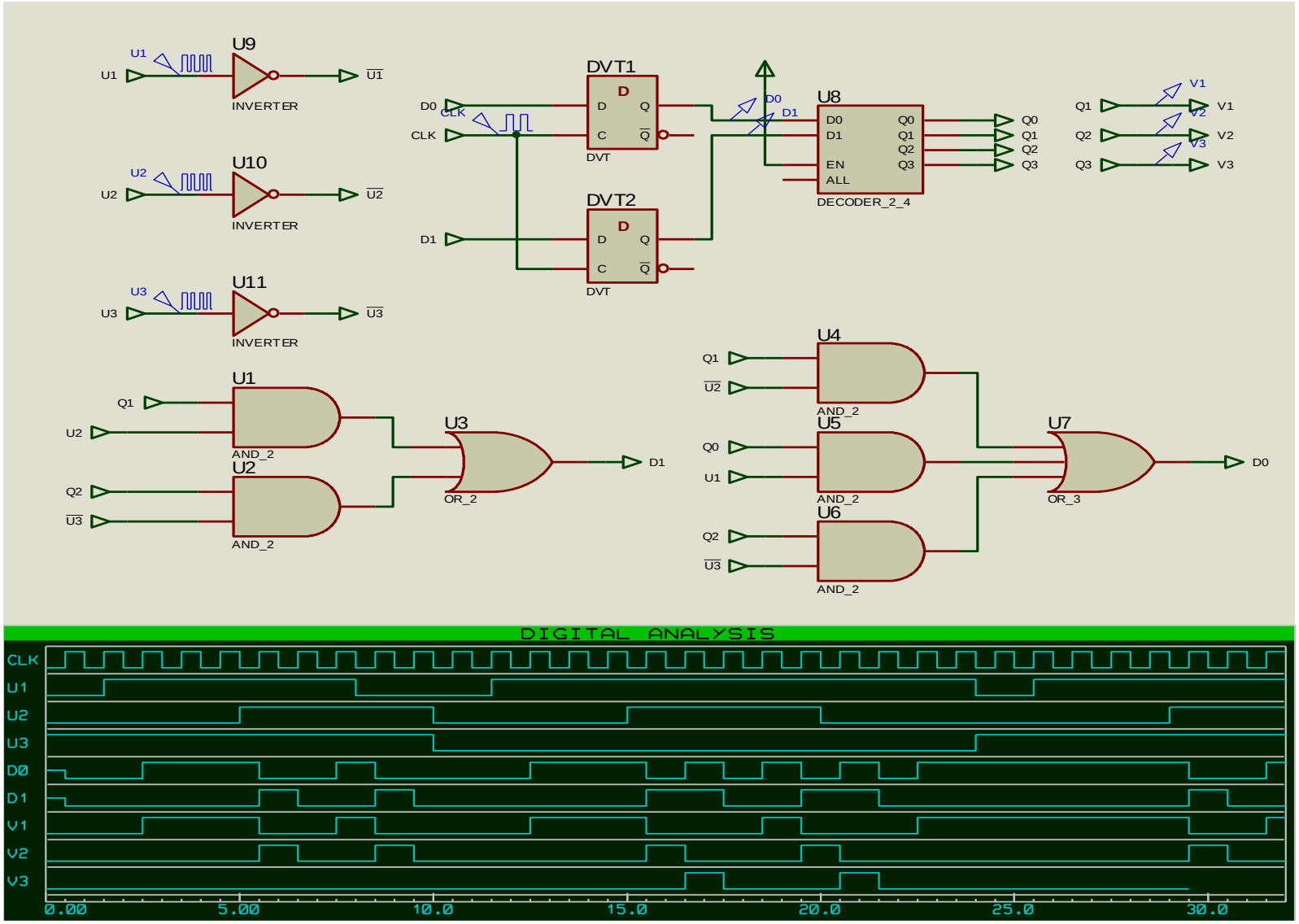


Наиболее используемые способы кодирования состояний: двоичное, One-Hot, Код Грея

$D_0(t+1) = Q_1(t)\ U\ Q_3(t) = Q_1!U_2\ U\ Q_0U_1\ U\ Q_2!U_3$;

$D_1(t+1) = Q_2(t)\ U\ Q_3(t) = Q_1U_2\ U\ Q_2!U_3$;

Автомат Мура



Автомат Мили

$V_i = U(Q_{Vi}U_j)$, где Q_{Vi} – состояния автомата, в которых сигнал V_i активен, U_j - условие перехода.

$V_1 = Q_0U_1 \cup Q_1!U_2$; $V_2 = Q_1U_2$; $V_3 = Q_2!U_3$;

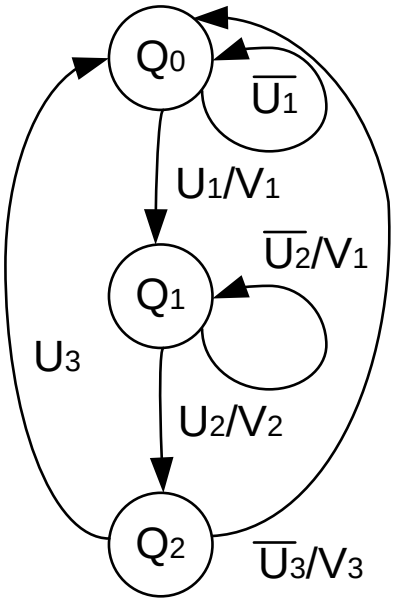
$Q_0 = Q_0!U_1 \cup Q_2U_3 \cup Q_2!U_3$;

$Q_1 = Q_1!U_2 \cup Q_0U_1$

$Q_2 = Q_1U_2$

Таблица кодирования состояний

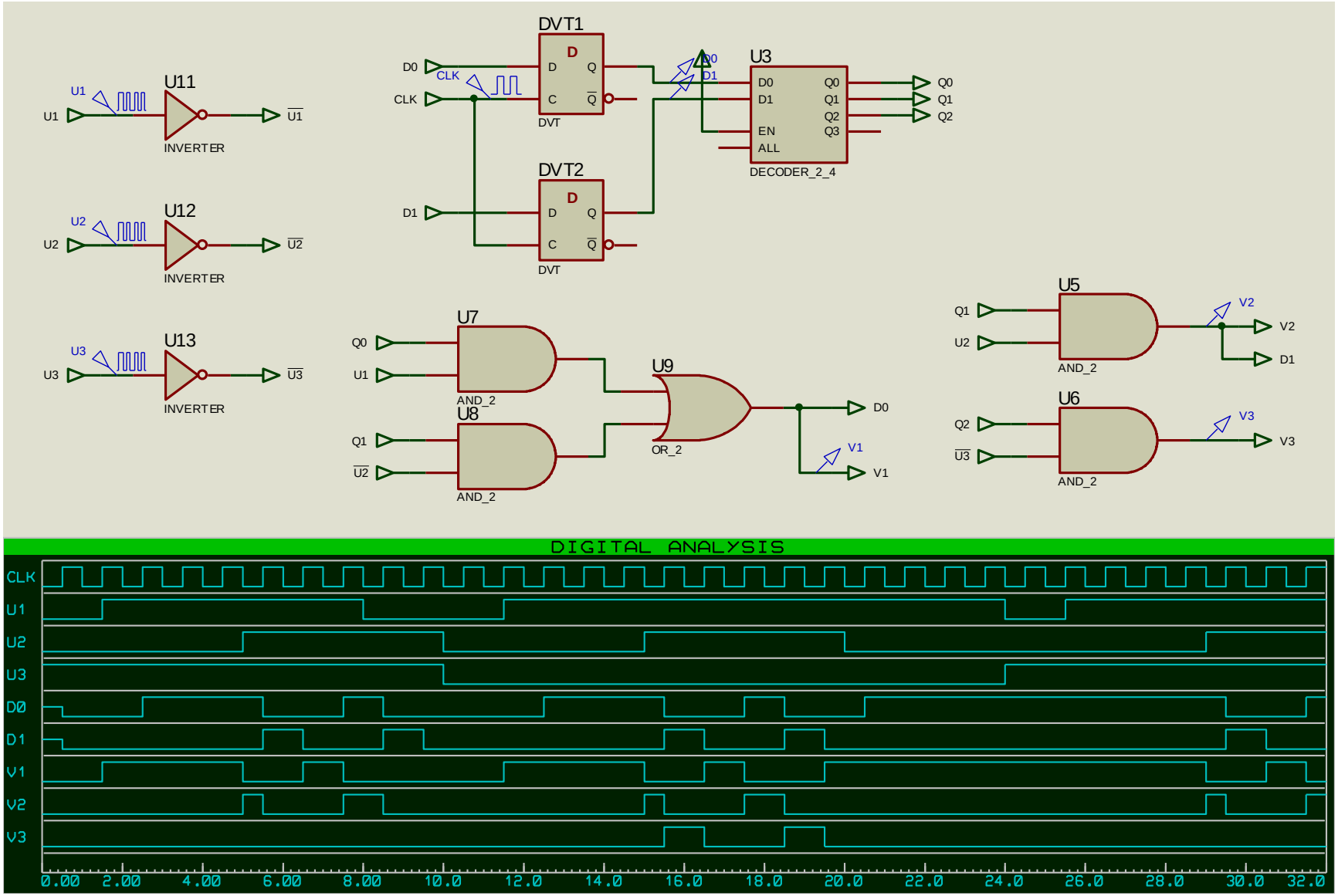
	D1	D0
Q0	0	0
Q1	0	1
Q2	1	0



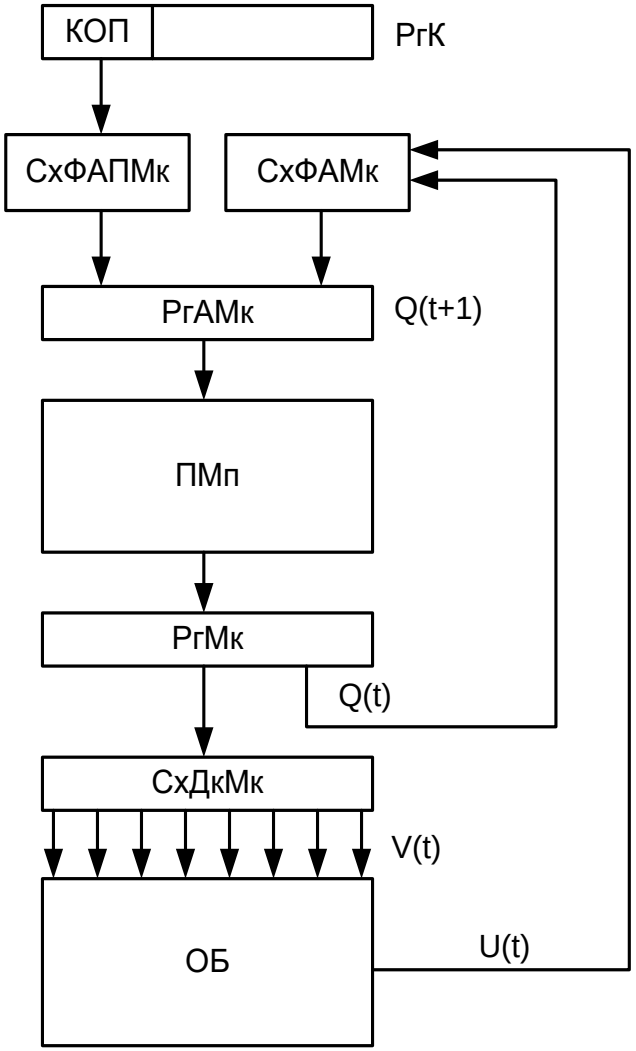
$D_0(t+1) = Q_1(t) = Q_1!U_2 \cup Q_0U_1$;

$D_1(t+1) = Q_2(t) = Q_1U_2$;

Автомат Мили



Управляющие устройства с программируемой логикой



РгК – регистр команды;
КОП – код операции;
СхФАПМк – схема формирования адреса первой микрокоманды;
СхФАМк – схема формирования адреса микрокоманды;
РгАМк – регистр адреса микрокоманды;
ПМп – память микропрограмм;
РгМк – регистр микрокоманды;
СхДкМк – схема декодирования микрокоманд;
ОБ – операционный блок.

<u>В зависимости от типа ПМп:</u>

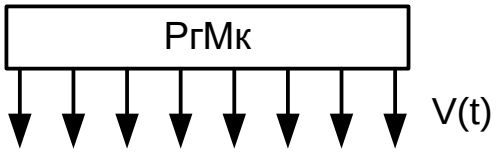
- -Статическое микропрограммирование (ROM);
- -Динамическое микропрограммирование (RAM).

<u>По способу адресации микрокоманд:</u>

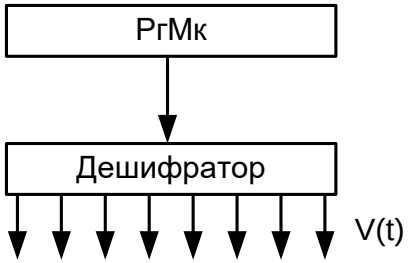
- -Принудительная адресация;
- -Естественная адресация.

Способы кодирования микрокоманд

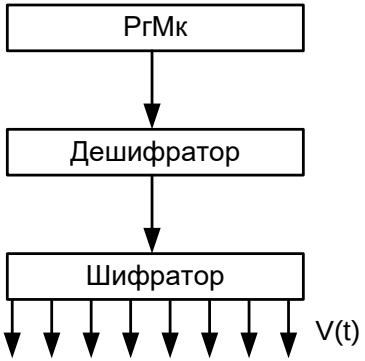
Минимальное кодирование (горизонтальное).



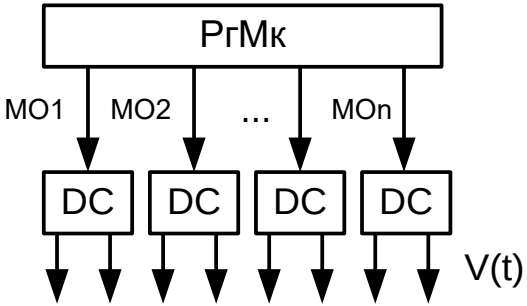
Максимальное кодирование (вертикальное).



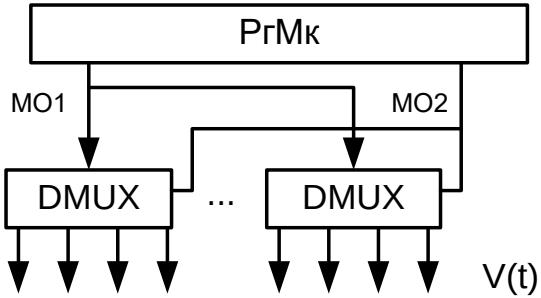
Максимальное кодирование с шифратором.



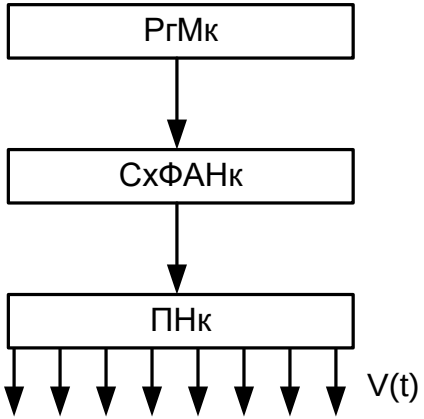
Вертикально-горизонтальное кодирование



Горизонтально-вертикальное кодирование.



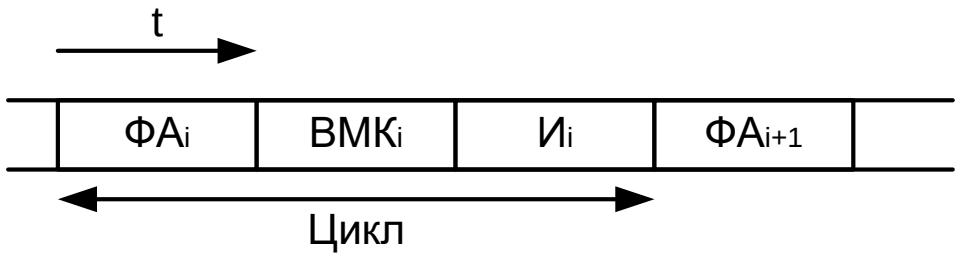
Кодирование с помощью памяти нанокоманд



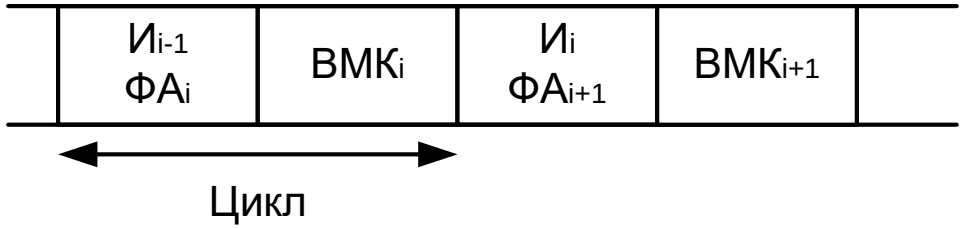
Цикл микрокоманды

Цикл исполнения микрокоманды;
-Формирование адреса микрокоманды.
-Выборка микрокоманды.
-Исполнение микрокоманды.

При последовательном исполнении цикла:
$T_{МК} = T_{ФА} + T_{ВМК} + T_{И}$.



При совмещении исполнения и формирования адреса следующей микрокоманды:
$T_{МК} = MAX((T_{ФА} + T_{ВМК}), T_{И})$.



При совмещении всех действий:
$T_{МК} = MAX(T_{ФА}, T_{ВМК}, T_{И})$.