

Архитектура ЭВМ

Лектор: к.т.н., доцент, Попов Алексей Юрьевич

Цель дисциплины:

- получить знания и навыки, необходимые для проектирования и эффективного использования современных аппаратных вычислительных средств.

Задачами дисциплины является изучение:

- принципов организации ЭВМ;
- методики проектирования ЭВМ и устройств, их составляющих.

ЛИТЕРАТУРА

1. Угрюмов Е. П. Цифровая схемотехника: Учеб. Пособие для вузов. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2004. – 800 с.: ил.
2. Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем: Учебник для вузов. – СПб.: Питер, 2004. – 668 с.: ил.
3. Каган Б.М. Электронные вычислительные машины и системы. - М.: Энергоатомиздат, 1991.

План проведения теоретических и практических занятий:

Семестр	Теоретические занятия	Лабораторные работы	Вид отчетности
	Попов Алексей Юрьевич	Шипилова Татьяна Дмитриевна Попов Алексей Юрьевич	
4	<ul style="list-style-type: none">• Вводная часть• Арифметические основы ЭВМ• Логические основы ЦВТ• Элементы и узлы ЭВМ• Организация памяти ЭВМ	<ul style="list-style-type: none">● Исследование работы триггеров● Исследование работы регистров● Исследование работы счетчиков.● Исследование работы мультиплексоров.	Зачет
5	<ul style="list-style-type: none">• Принципы построения и архитектура ЭВМ• Процессорные устройства• Организация ввода вывода• Вычислительные системы	<ul style="list-style-type: none">● Разработка радиоэлектронной аппаратуры на основе микроконтроллеров ARM7 TDMI● Синхронизация микроконтроллеров ARM7 TDMI и управление таймерами● Хакатон: Быстрое прототипирование решений Интернета вещей● Организация памяти конвейерных суперскалярных электронных вычислительных машин	Экзамен

Страница курса

e-learning.bmstu.ru/moodle/

- Поиск (google,yandex) по слову «ИУ6»
- Ресурсы,
- Курсы,
- Учебные дисциплины кафедры «Компьютерные системы и сети»
- Архитектура ЭВМ

The screenshot shows a Moodle course page. At the top, there is a dark header bar with the university logo and navigation links for 'РЕСУРСЫ', 'О КАФЕДРЕ', 'АБИТУРИЕНТАМ', 'ДИПЛОМНИКАМ', and a guest login message. Below the header is a breadcrumb navigation bar: 'В начало' > 'Курсы' > 'Учебные дисциплины кафедры "Компьютерные системы и сети"' > 'АрхЭВМ'. On the left, there is a vertical sidebar with icons for 'Вход' (Login), 'Градусы' (Degree), 'Дом' (Home), and 'Календарь' (Calendar). The main content area has a title 'ЛЕКЦИИ' (Lectures) and a list of 7 lecture topics, each with a small document icon:

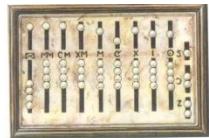
1. Арифметические основы ЭВМ
2. Элементы и узлы ЭВМ
3. Организация памяти ЭВМ
4. Принципы построения и архитектура ЭВМ
5. Процессорные устройства
6. Операционные устройства ЭВМ
7. Организация ввода-вывода

I. Введение

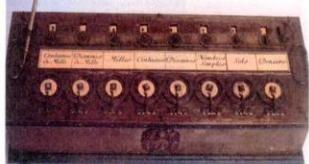
История развития вычислительной техники.

Механические вычислительные устройства.

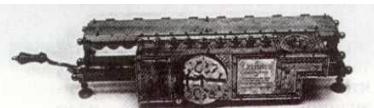
Абак



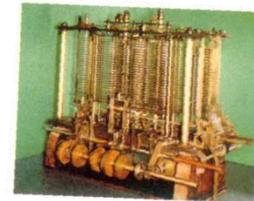
Машина
Паскаля



Машина
Лейбница



Машина
Бэбиджа



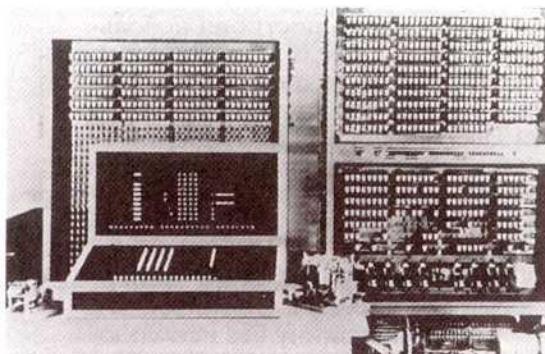
Современные
механические машины



Электромеханические счетные машины

Машины Конрада Цузе (Z1, Z2, Z3, Z4)

- Z1 – полностью механическая машина (1936);
- Z2 – использование реле в арифметическом устройстве (1939);
- Z3 и Z4 – электромеханические машины с механической памятью (1941 и 1945).



Машина Z3

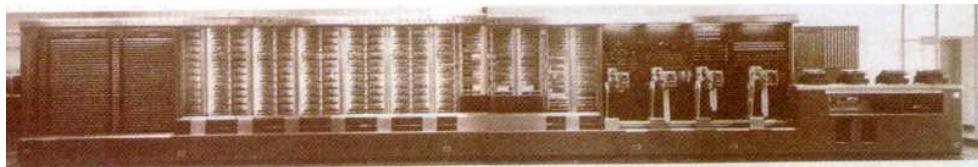


Машина Z4

Поколения электронных вычислительных машин

Первое поколение ЭВМ (с конца 30-х до середины 50-х)

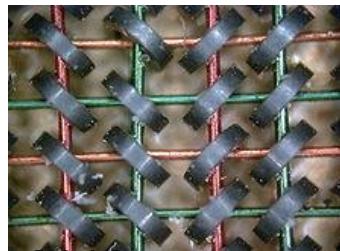
Поколение ЭВМ	Элементная база	Тип основного запоминающего устройства	Представители классов ЭВМ	Языки программирования	Программное обеспечение	Средства связи с пользователем
I (с конца 30-х до середины 50-х)	Электро-магнитные реле; электронные лампы	Линии задержки на электронные лучевые трубках, Ферритовые сердечники ($\sim 2^{12}$ - 2^{16})	Калькуляторы (ABC, ENIAC), Большие ЭВМ (MARK I, EDVAC, UNIVAC, БЭСМ, МЭСМ, Стрела, Минск, IAS)	Ручная коммутация, Машинные коды	Ассемблер	Индикаторы, Пульт управления, Перфокарты



ЭВМ MARK I



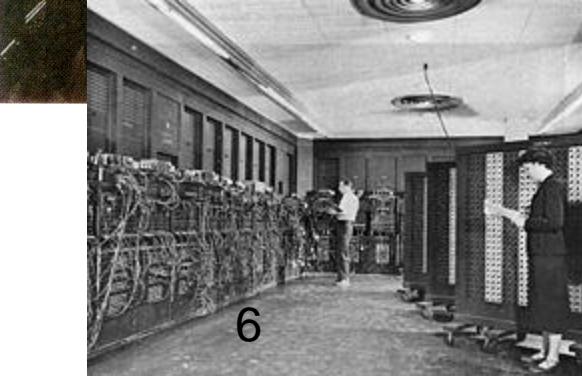
ЭВМ ENIAC



Ферритовые сердечники

2018

Архитектура ЭВМ



6

Второе поколение ЭВМ (с середины 50-х до середины 60-х)

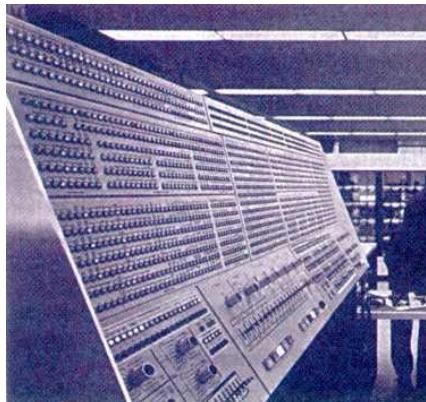
Поколение ЭВМ	Элементная база	Тип основного запоминающего устройства	Представители классов ЭВМ	Языки программирования	Программное обеспечение	Средства связи с пользователем
II (с середины 50-х до середины 60-х)	Транзисторы	Ферритовые сердечники (до 2 ¹⁹)	Малые и средние ЭВМ (БЭСМ-4, Урал-14, Минск-2, Днепр), Большие ЭВМ (TRADIAS, IBM 7030, IBM 7090, TX-O, БЭСМ-2,3)	Фортран, Алгол, Кобол	Компиляторы, автоматизированные системы управления, диспетчеры	Индикаторы, Пульт управления, Перфокарты, Перфоленты

ЭВМ БЭСМ-4



Третье поколение ЭВМ (с середины 60-х до середины 70-х)

Поколение ЭВМ	Элементная база	Тип основного запоминающего устройства	Представители классов ЭВМ	Языки программирования	Программное обеспечение	Средства связи с пользователем
III (с середины 60-х до середины 70-х)	Интегральные схемы малой и средней степени интеграции	Полупроводниковые ЗУ на интегральных схемах (до 2^{25})	Мини и микро-ЭВМ (Мир-1, М220), Средние и большие универсальные ЭВМ (ILLIAC IV, CDC6600, CDC7600, IBM 360, ЕС ЭВМ, СМ ЭВМ, БЭСМ-6)	Фортран, Алгол, В, С	ОС (UNIX, IBM), СУБД, САПР, Пакеты прикладных программ	Алфавитно-цифровые дисплеи

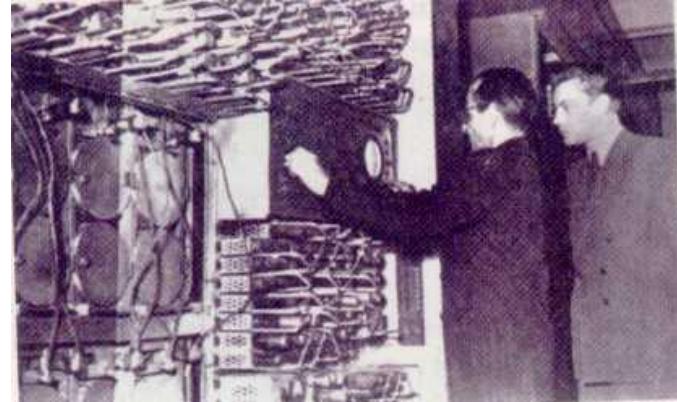


IBM 360



2018

Архитектура ЭВМ

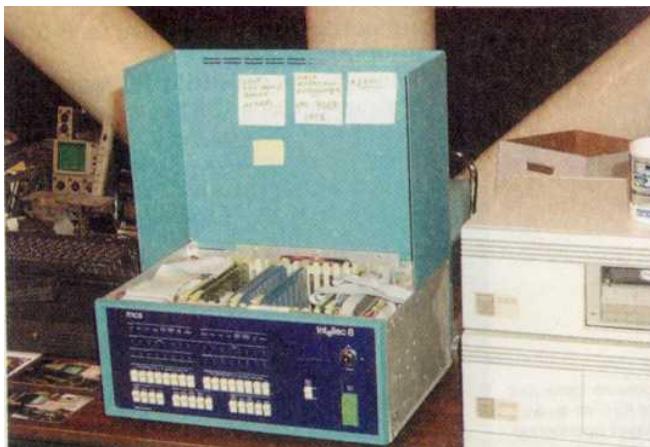


ЭВМ БЭСМ-6

8

Четвертое поколение ЭВМ (с середины 70-х до середины 80-х)

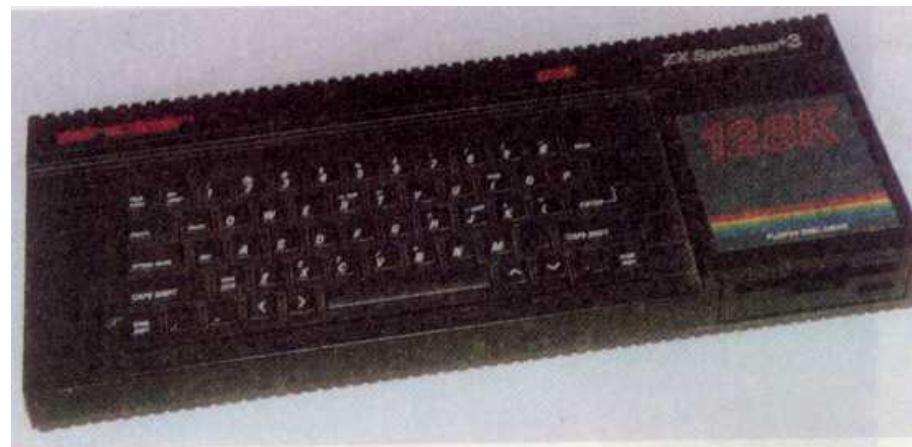
Поколение ЭВМ	Элементная база	Тип основного запоминающего устройства	Представители классов ЭВМ	Языки программирования	Программное обеспечение	Средства связи с пользователем
IV (с середины 70-х до середины 80-х)	Интегральные схемы большой и сверхбольшой степени интеграции	Полупроводниковые ЗУ на сверх больших интегральных схемах (до 2^{28})	Персональные компьютеры (Intellec8, IBM PC/XT/AT, Sinclair Spectrum), Средние и Большие ЭВМ (Cray, Эльбрус-1,2,3)	Пролог, Фортран, С, Паскаль	Графические ОС, Среды визуальной разработки, САПР, Системы программирования, Игры	Графические дисплеи, клавиатура, мышь



Intellec8 (Intel 8080)

2018

Архитектура ЭВМ



Sinclair Spectrum

9

Пятое поколение ЭВМ (с середины 80-х)

Поколение ЭВМ	Элементная база	Тип основного запоминающего устройства	Представители классов ЭВМ	Языки программирования	Программное обеспечение	Средства связи с пользователем
V (с середины 80-х)	Интегральные схемы сверхбольшой степени интеграции	Полупроводниковые ЗУ на сверх больших интегральных схемах (до $\sim 2^{32}$)	ПК на универсальных конвейерных МП (IA 32, PowerPC), Средние большие ЭВМ с массовым параллелизмом (серия IBM Mainframes, Cray, HP, DEC)	Языки с ООП, Языки параллельного программирования (MPI), Специализированные языки (VHDL, Perl, PHP, SQL и т.д.)	Мультимедиа, WWW	Графические дисплеи, клавиатура, мышь, звук

Классификация ЭВМ

Классификация ЭВМ по назначению:

Общего назначения

- Супер ЭВМ
- Минисупер ЭВМ
- Мэйнфреймы
- Серверы
- Рабочие станции
- Персональные компьютеры
- Ноутбуки
- Портативные компьютеры
- ...

Специализированные

...

Классификация ЭВМ по структуре:

- Однопроцессорные
- Многопроцессорные

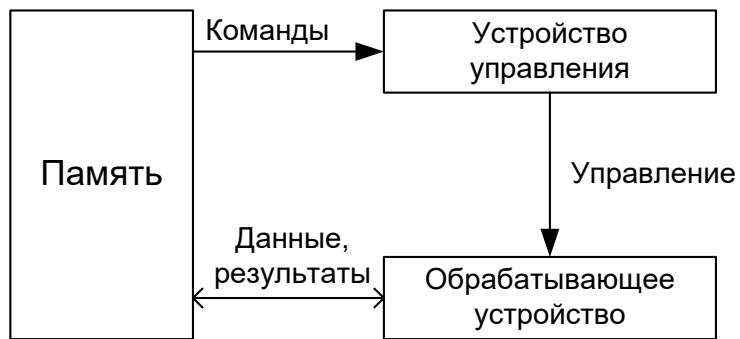
Классификация ЭВМ по режимам работы:

- Однопрограммные
- Мультипрограммные
- Мультипрограммные в составе систем
- ЭВМ в системах реального времени

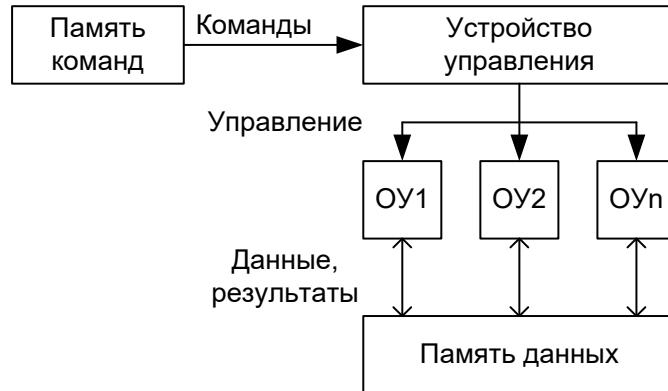
Классификация ЭВМ по количеству потоков команд и данных:

- ЭВМ с одним потоком команд и одним потоком данных (ОКОД, SISD);
- ЭВМ с одним потоком команд и многими потоками данных (ОКМД, SIMD);
- ЭВМ с многими потоками команд и одним потоком данных (МКОД, MISD);
- ЭВМ с многими потоками команд и многими потоками данных (МКМД, MIMD).

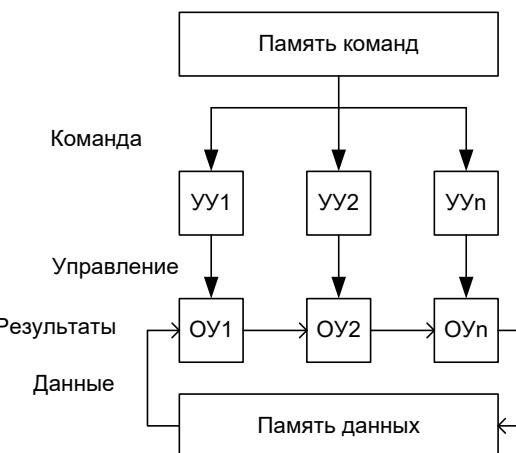
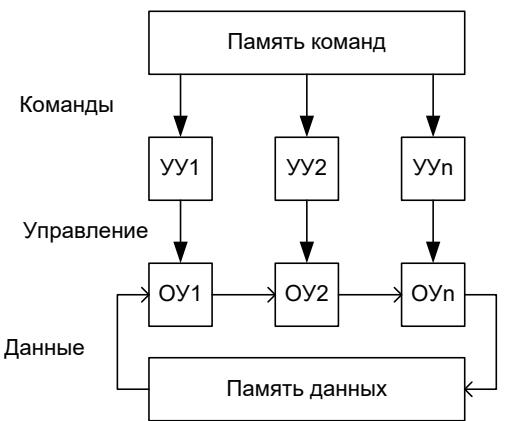
ОКОД, SISD



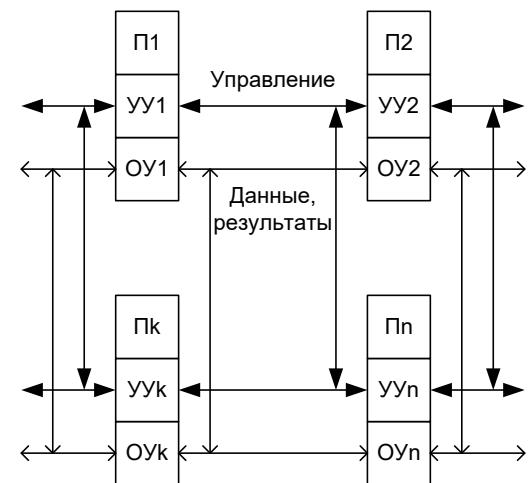
ОКМД, SIMD



МКОД, MISD



МКМД, MIMD



Основные характеристики ЭВМ

- Эффективность
- Производительность
- Надежность
- Стоимость
- Энергопотребление

Общий коэффициент эффективности

$$\Theta := \frac{P}{C_{\text{ЭВМ}} + C_{\text{эксплуатации}}}$$

Θ - Общий коэффициент эффективности,
 P - Производительность,
 $C_{\text{ЭВМ}}$ - Стоимость ЭВМ,
 $C_{\text{эксплуатации}}$ - Стоимость эксплуатации.

$$\Theta' := \frac{P}{C_{\text{ЭВМ}}}$$

$C_{\text{ЭВМ}} \gg C_{\text{эксплуатации}}$

$$\Theta := \frac{P \cdot K_{\text{и}}}{C_{\text{ЭВМ}} + C_{\text{эксплуатации}}}$$

Θ' - Эффективность без учета эксплуатационных издержек.
 $\Theta_{\text{н}}$ - Эффективность с учетом эксплуатационной надежности.

Производительность ЭВМ

$$P := \frac{\sum_{s=1}^n K_s}{\sum_{s=1}^n K_s \cdot t_s}$$

K_s
 t_s

- Весовой коэффициент задачи S,
- Время выполнения задачи S.

Единицы измерения производительности:

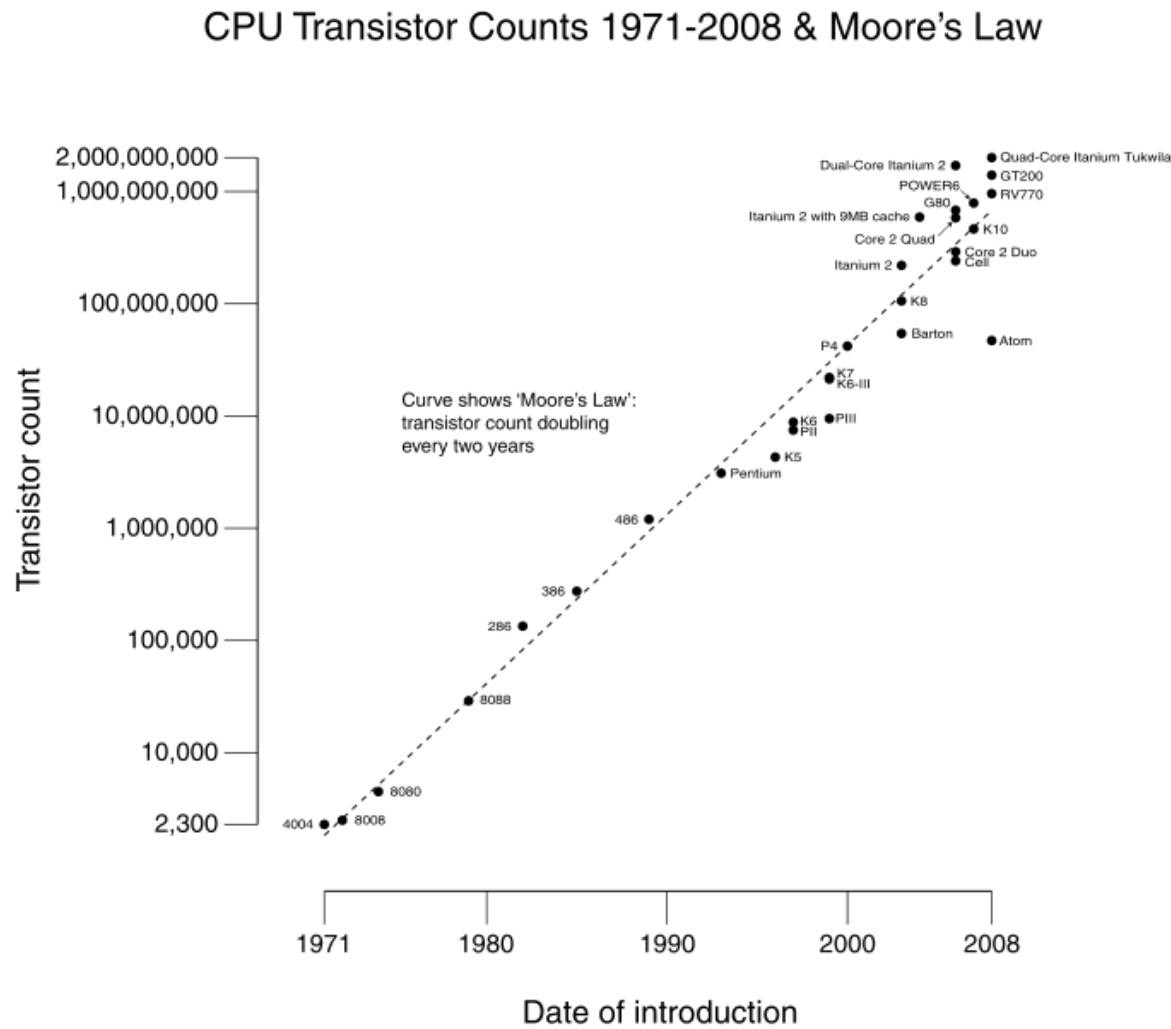
MIPs = 10^6 целочисленных операций в секунду.

MFlops = 10^6 операций с плавающей запятой в секунду.



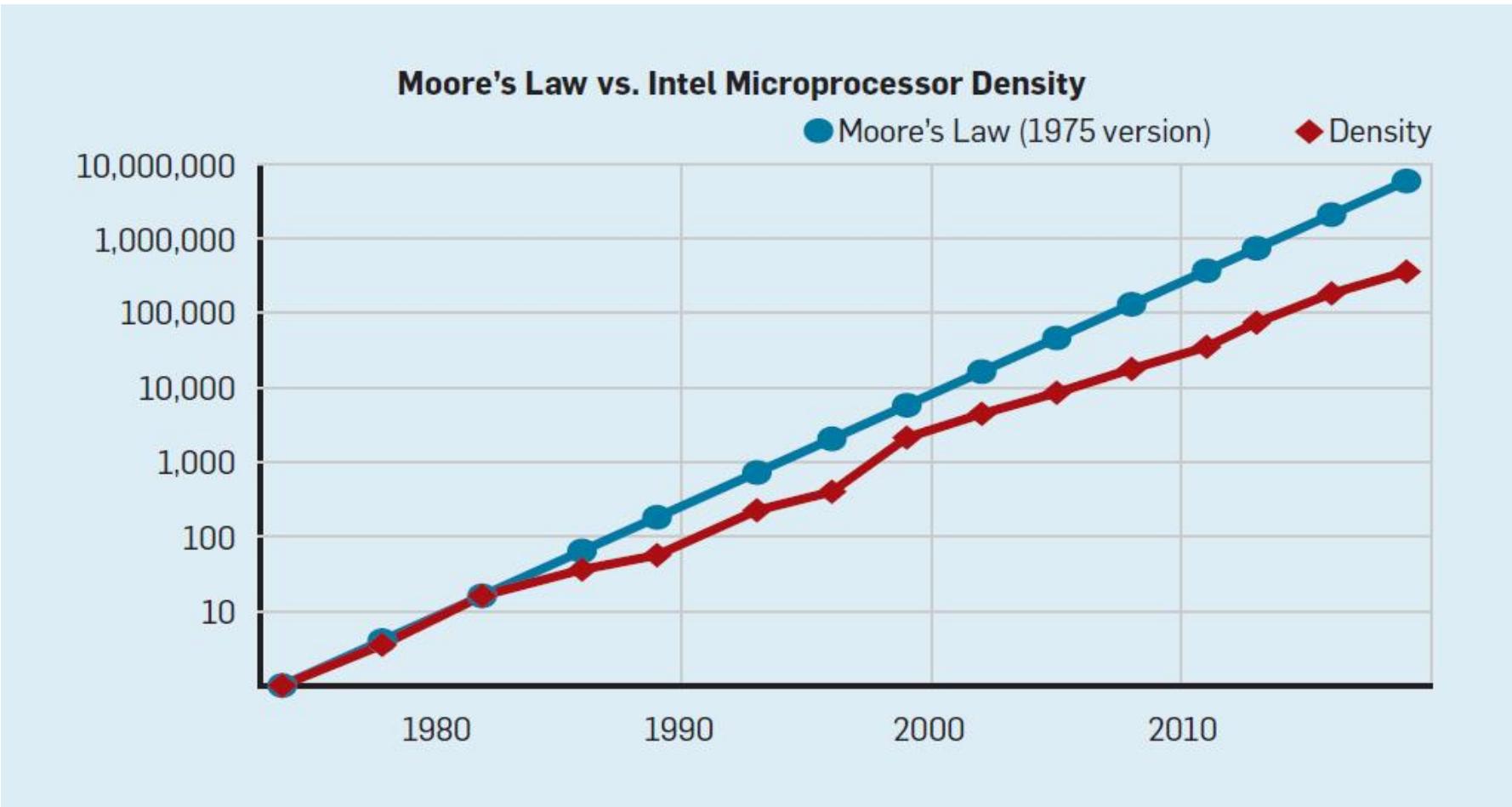
Закон Мура

Число транзисторов на кристалле будет удваиваться каждые 24 месяца



Закон Мура

Количество транзисторов на чипе Intel по сравнению с законом Мура

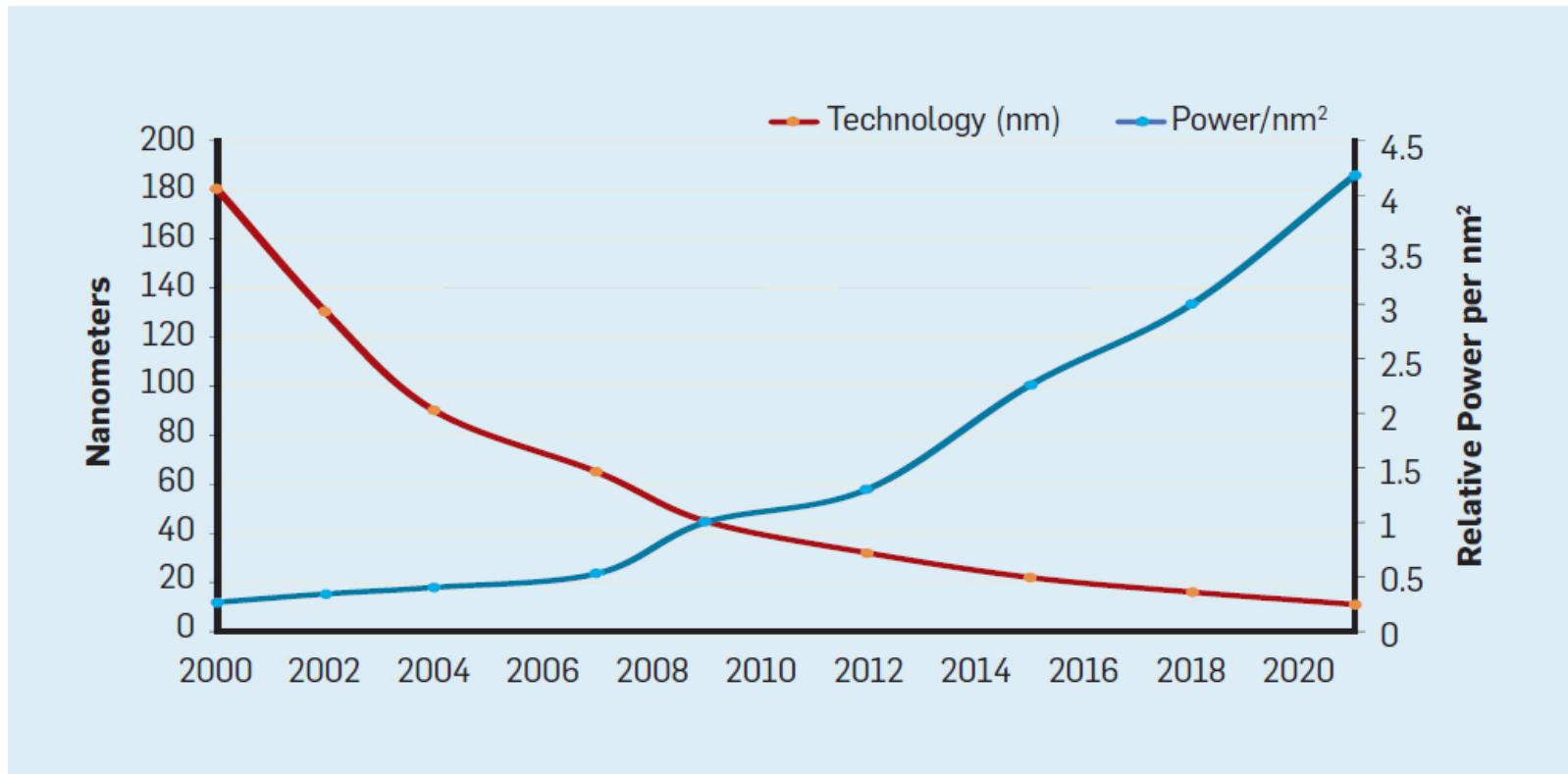


<https://m.habr.com/ru/post/440760/>

Масштабирование Деннарда

По мере увеличения плотности транзисторов потребление энергии на транзистор будет падать, поэтому потребление на мм^2 кремния будет почти постоянным

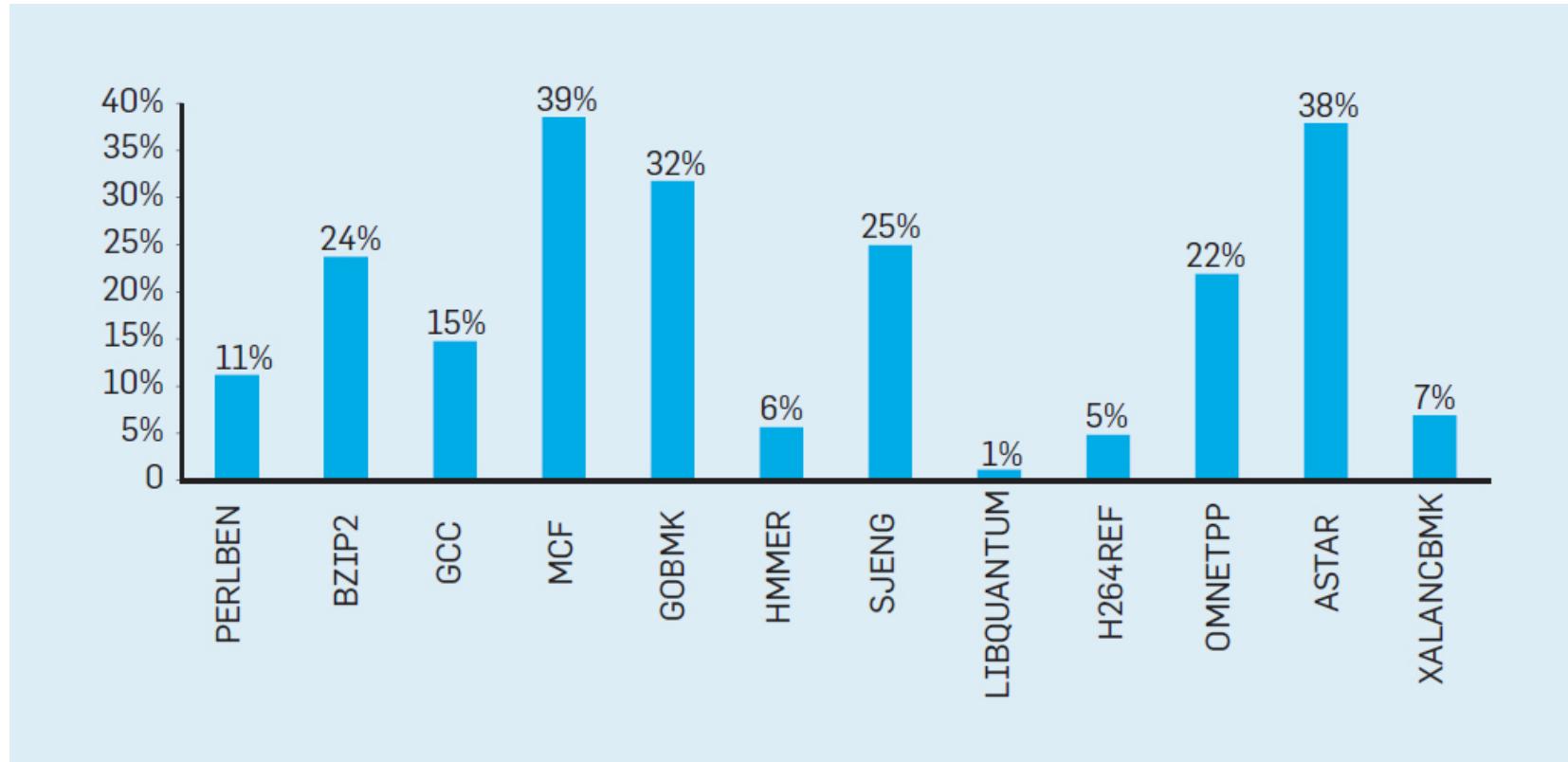
Количество транзисторов на чип и потребление энергии на мм^2



<https://m.habr.com/ru/post/440760/>

Эффективность современных микропроцессоров

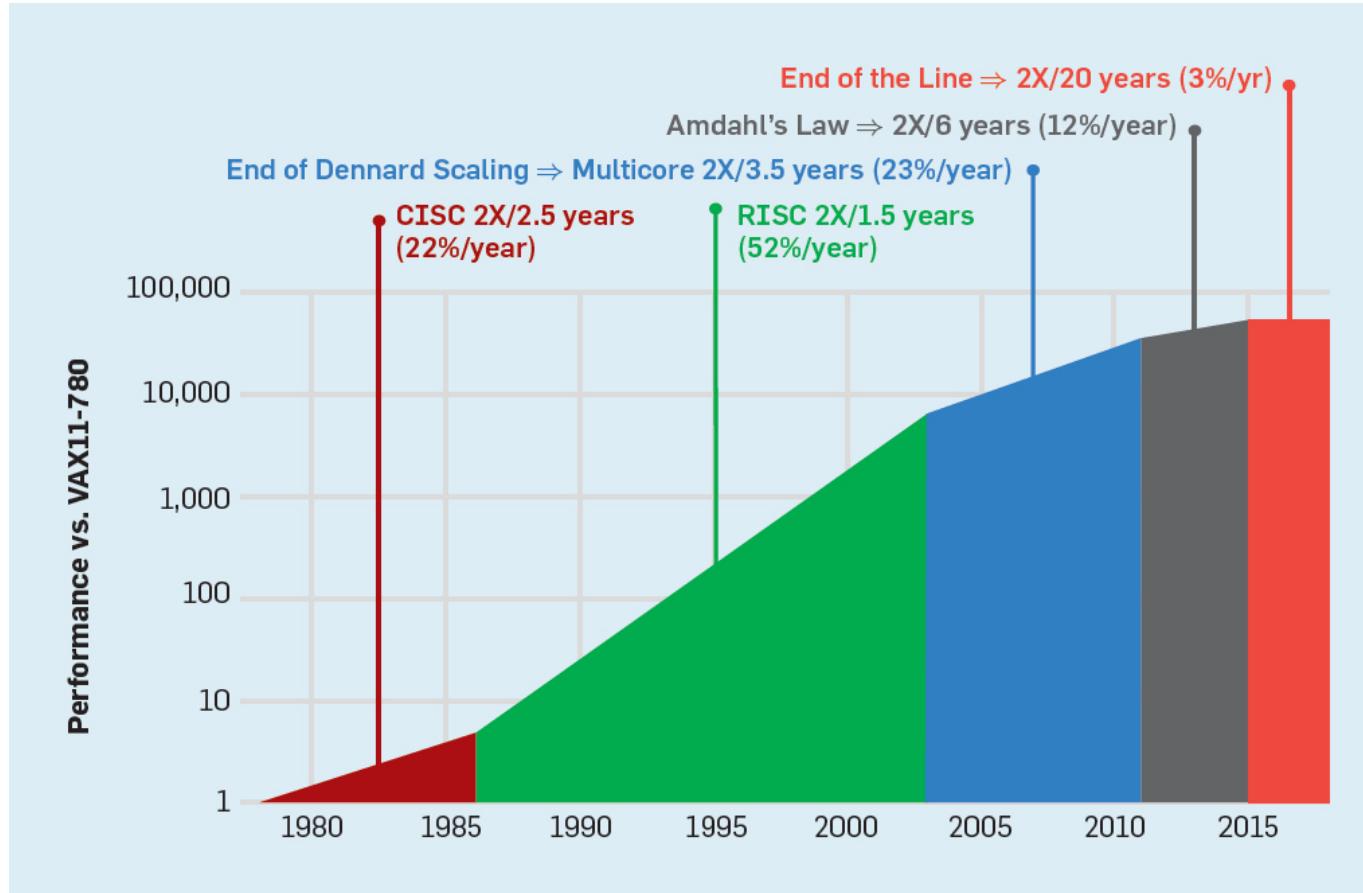
Потраченные впустую инструкции в процентах от всех инструкций, выполненных на Intel Core i7 для различных целочисленных тестов SPEC



<https://m.habr.com/ru/post/440760/>

Эффективность современных микропроцессоров

Рост компьютерной производительности по целочисленным тестам (SPECintCPU)

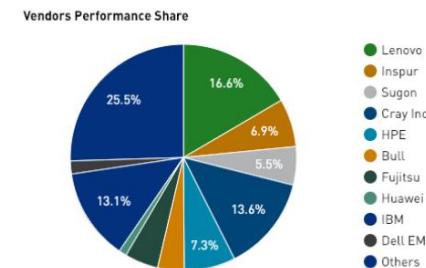
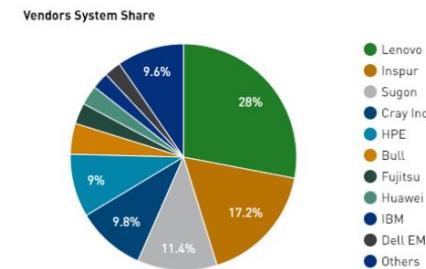


<https://m.habr.com/ru/post/440760/>

Список наиболее производительных ЭВМ (11.2018)

Параметры: Количество процессоров; Максимальная производительность Rmax (TFlops); Пиковая производительность Rpeak (TFlops); Рассекаемая мощность (KW).

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,397,824	143,500.0	200,794.9	9,783
2	Sierra - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
4	Tianhe-2A - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	27,154.3	2,384
6	Trinity - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect , Cray Inc. DOE/NNSA/LANL/SNL United States	979,072	20,158.7	41,461.2	7,578
7	AI Bridging Cloud Infrastructure (ABCI) - PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	32,576.6	1,649
8	SuperMUC-NG - ThinkSystem SD530, Xeon Platinum 8174 24C 3.1GHz, Intel Omni-Path , Lenovo Leibniz Rechenzentrum Germany	305,856	19,476.6	26,873.9	
9	Titan - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x , Cray Inc. DOE/SC/Oak Ridge National Laboratory United States	560,640	17,590.0	27,112.5	8,209
10	Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom , IBM DOE/NNSA/LLNL United States	1,572,864	17,173.2	20,132.7	7,890

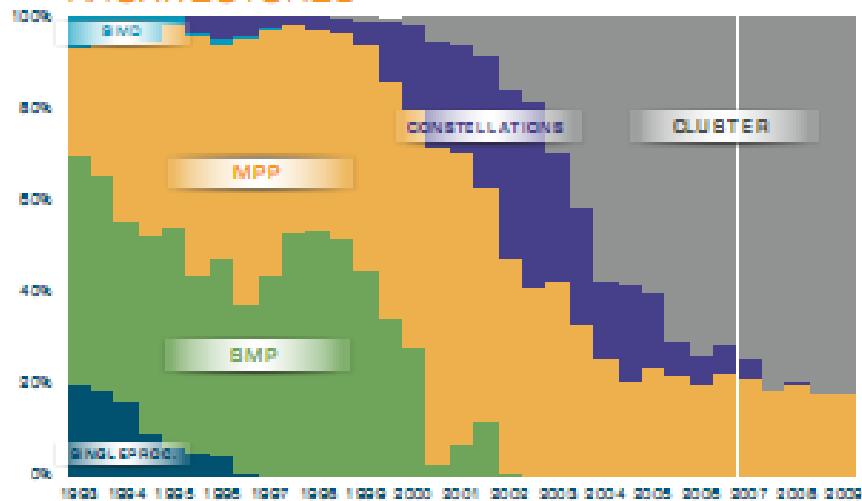


Список наиболее производительных ЭВМ (11.2018, продолжение)

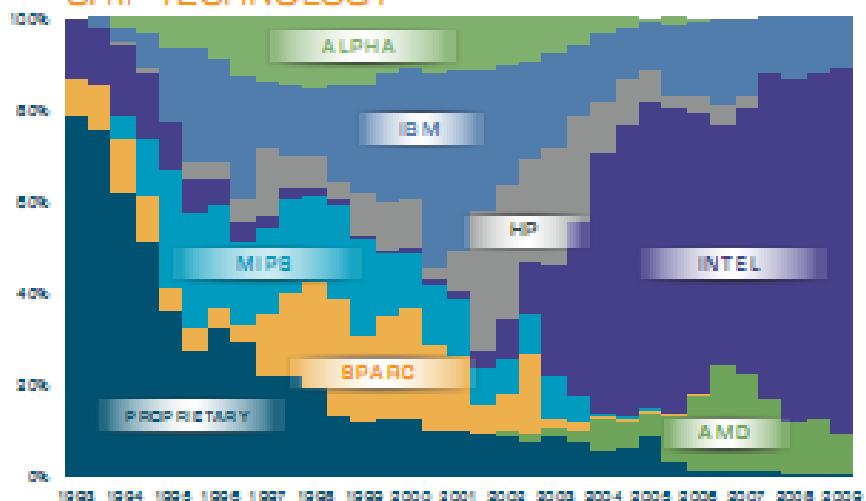
Rank	System		Rmax Cores	Rpeak (TFlop/s)	Power (TFlop/s)	(kW)
63	Lomonosov 2 - T-Platform A-Class Cluster, Xeon E5-2697v3 14C 2.6GHz, Infiniband FDR, Nvidia K40m , T-Platforms	Moscow State University - Research Computing Center Russia	42,688	2,102.0	2,962.3	1,079
227	Lomonosov - T-Platforms T-Blade2/1.1, Xeon X5570/X5670/E5630 2.93/2.53 GHz, Nvidia 2070 GPU, PowerXCell 8i Infiniband QDR , T-Platforms	Moscow State University - Research Computing Center Russia	78,660	901.9	1,700.2	2,800
412	Polytechnic RSC Tornado - RSC Tornado, Xeon E5-2697v3 14C 2.6GHz, Infiniband FDR , RSC Group	St. Petersburg Polytechnic University Russia	19,936	658.1	829.3	320

Top500 статистика

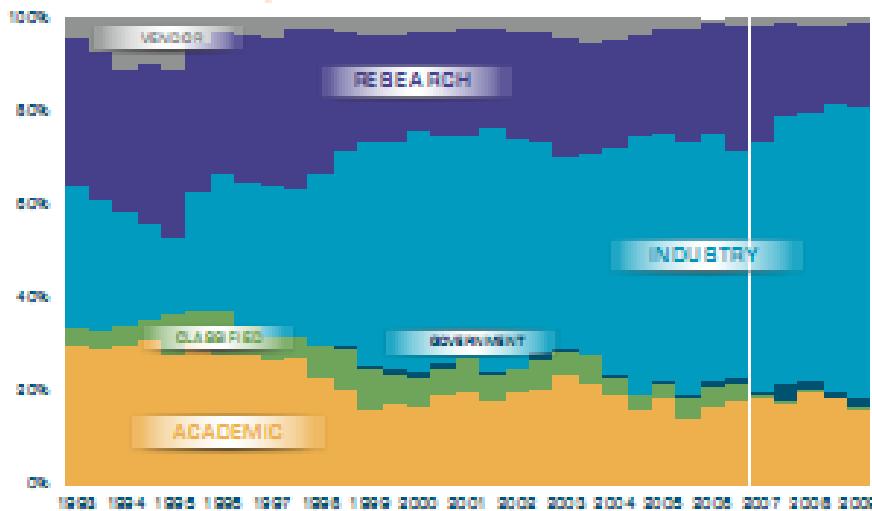
ARCHITECTURES



CHIP TECHNOLOGY



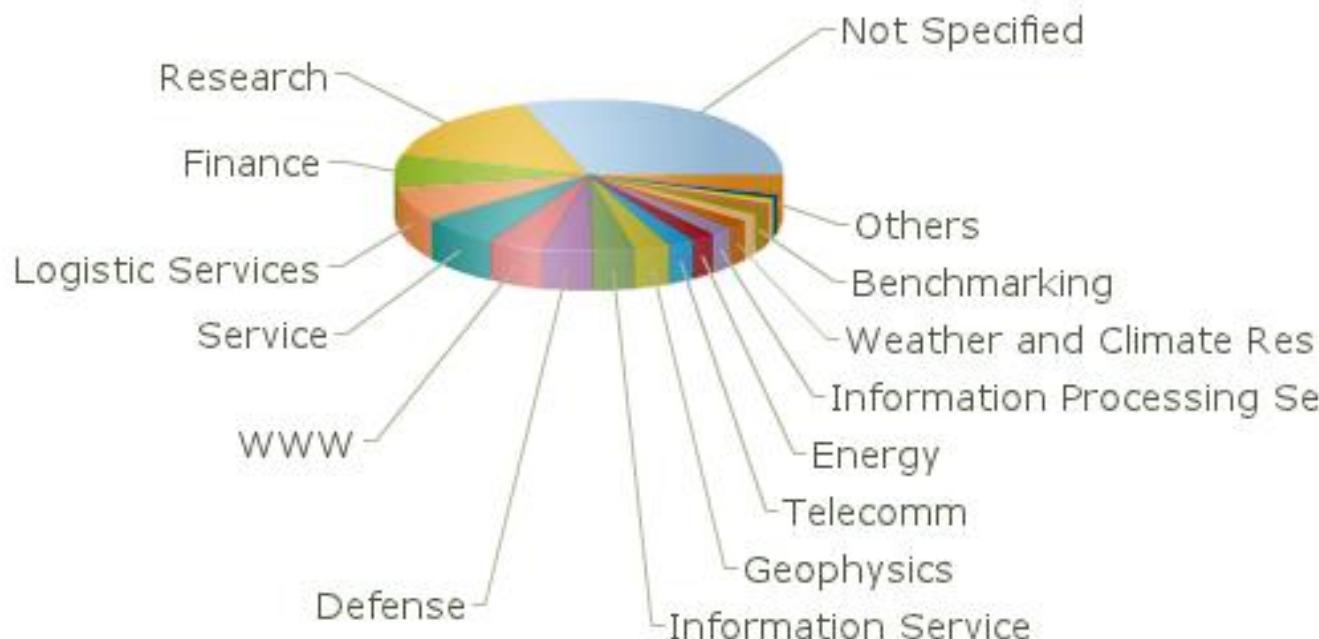
INSTALLATION TYPE



Top500 статистика

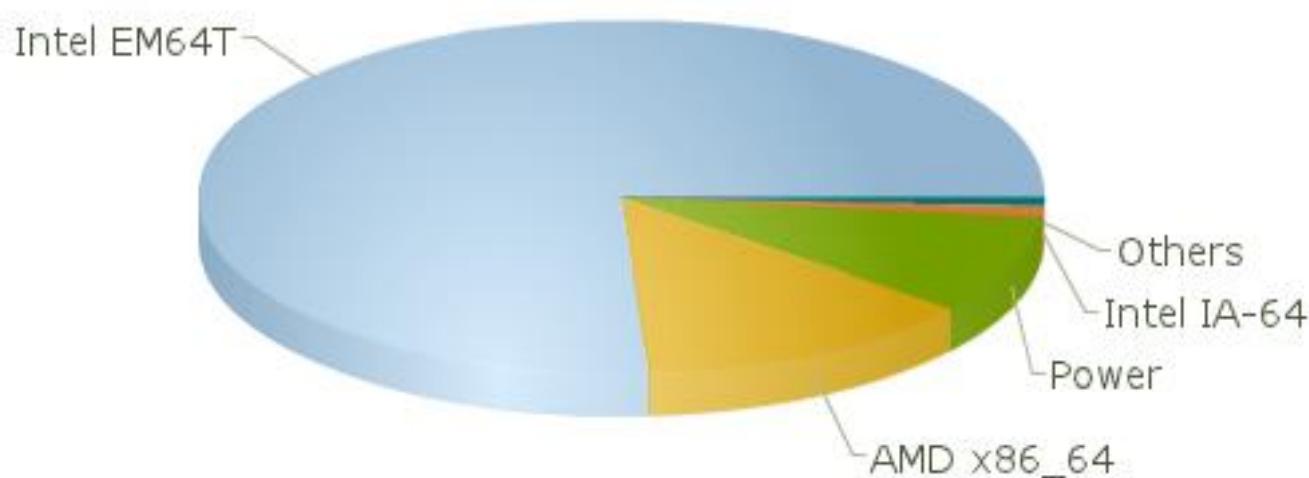
Application Area / Systems

June 2011



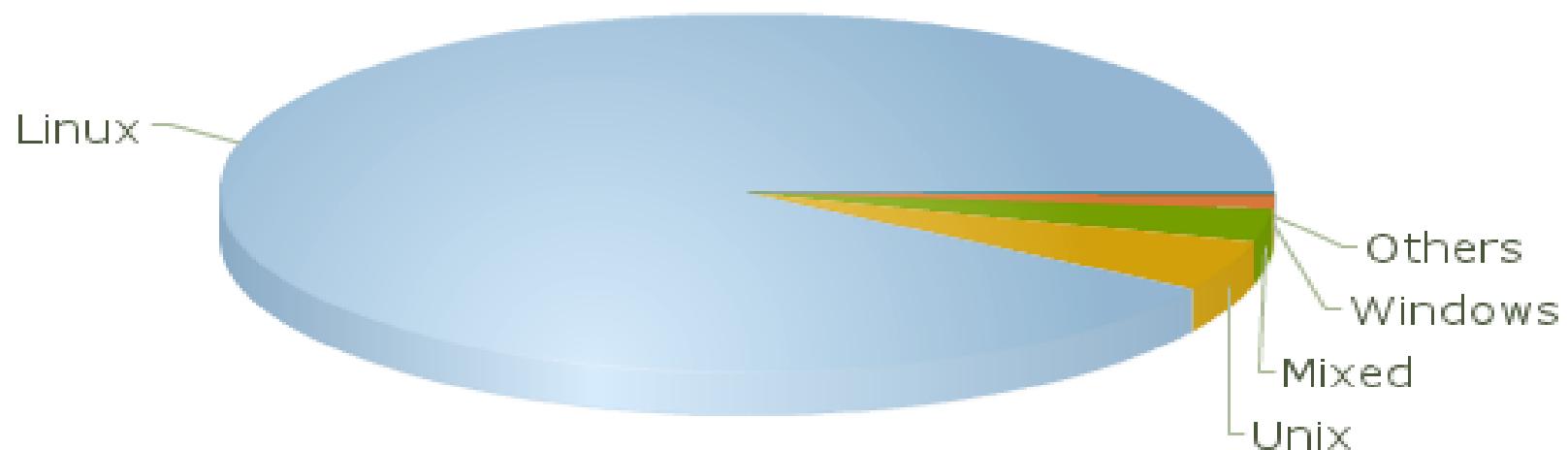
Top500 статистика

Processor Family / Systems
June 2011

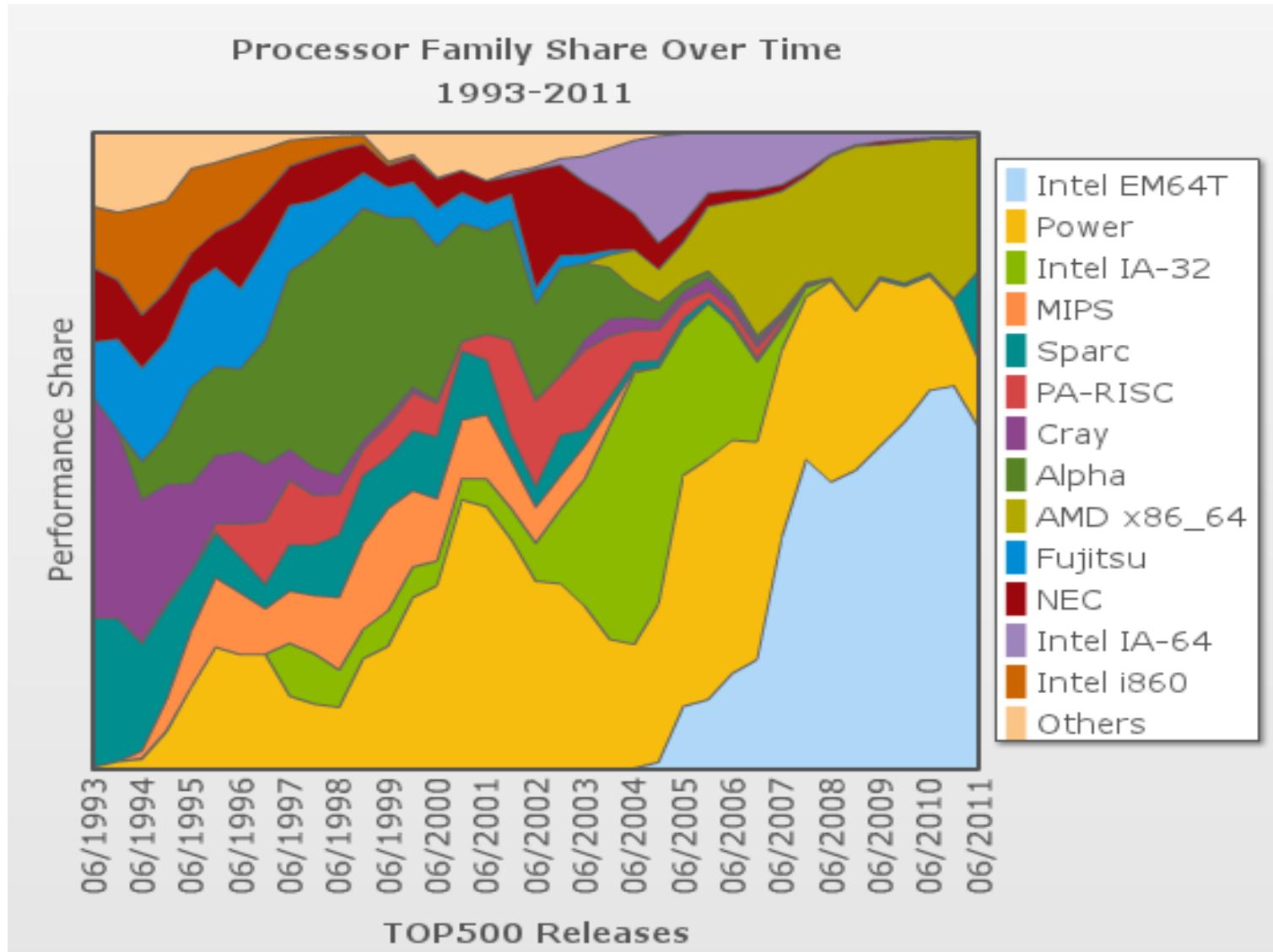


Top500 статистика

Operating system Family / Systems
June 2011



Top500 статистика



Device democracy

Интернет вещей - IoT (Internet of Things)

концепция вычислительной сети физических объектов («вещей»), оснащённых встроенными технологиями для взаимодействия друг с другом или с внешней средой, рассматривающая организацию таких сетей как явление, способное перестроить экономические и общественные процессы, исключающее из части действий и операций необходимость участия человека.

Демократия устройств - Device Democracy

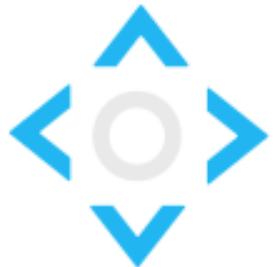
Концепция построения сети физических объектов, которым предоставлены полномочия самостоятельного и коллегиального принятия решений о дальнейшем поведении.

IBM Institute for Business Value

Device democracy

Saving the future of the Internet of Things





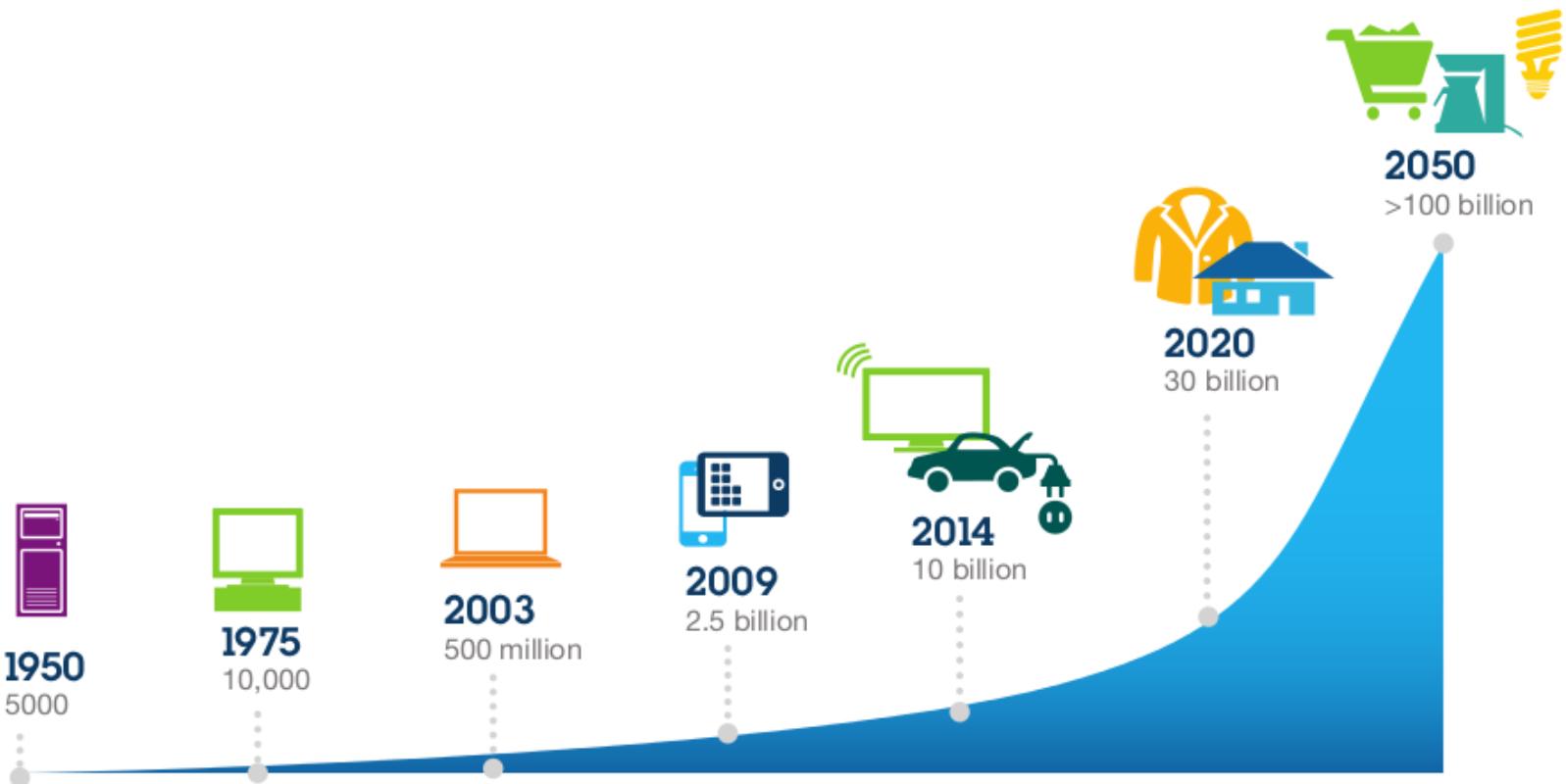
In the emerging device-driven democracy, power in the IoT will shift from the center to the edge.

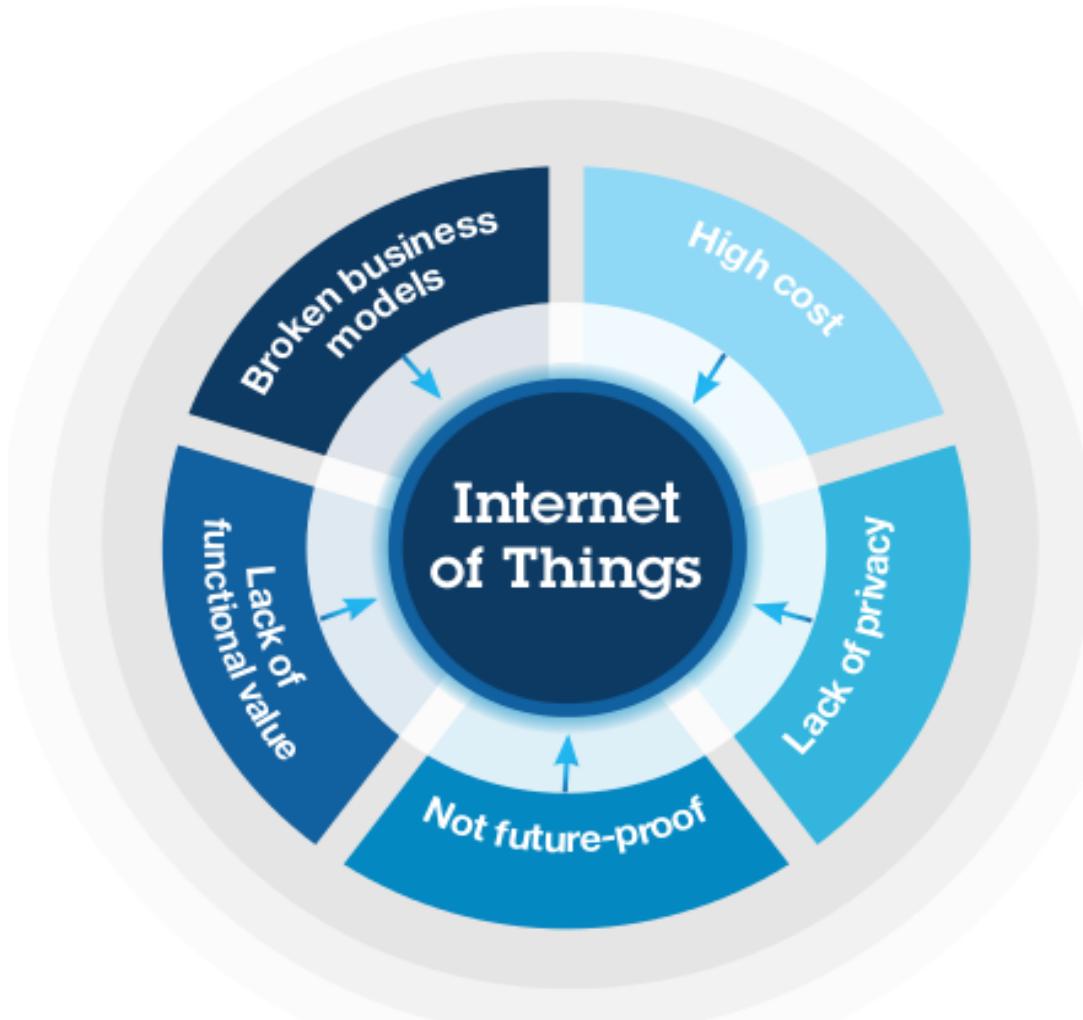


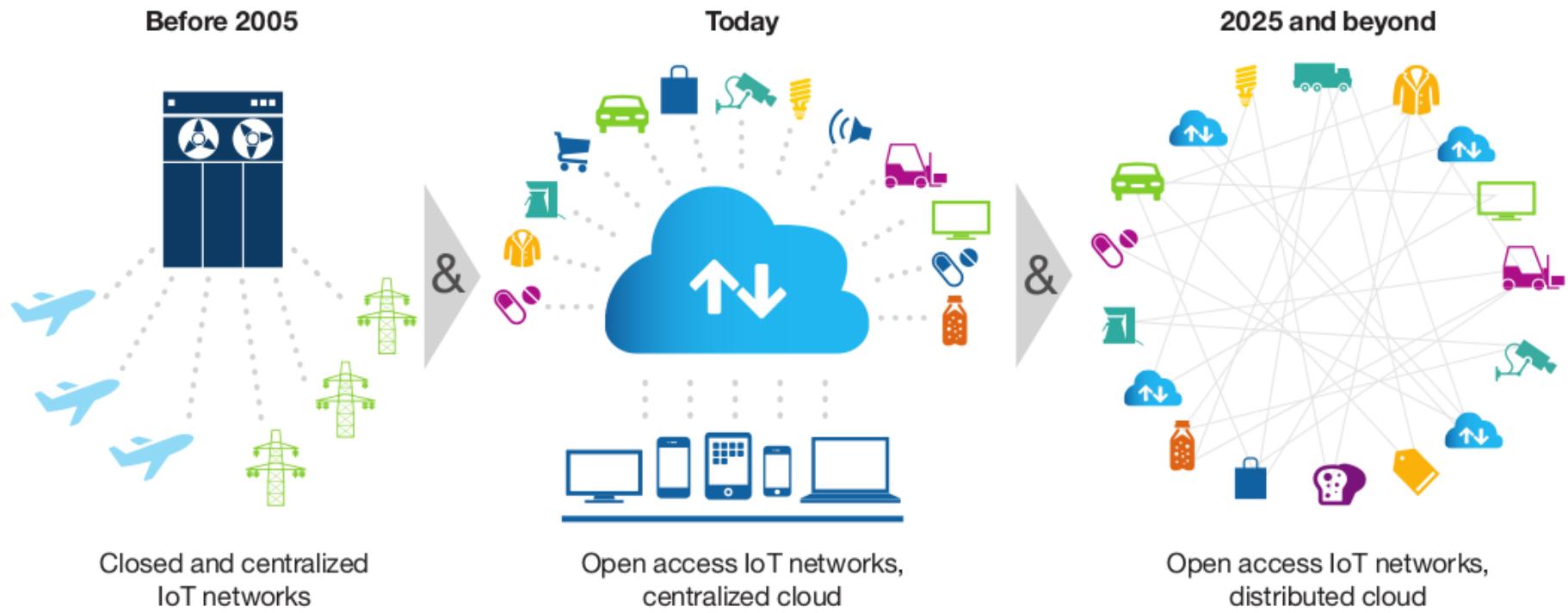
As devices compete and trade in real-time, they will create liquid markets out of the physical world.



In the IoT of hundreds of billions of devices, connectivity and intelligence will be a means to better products and experiences, not an end.

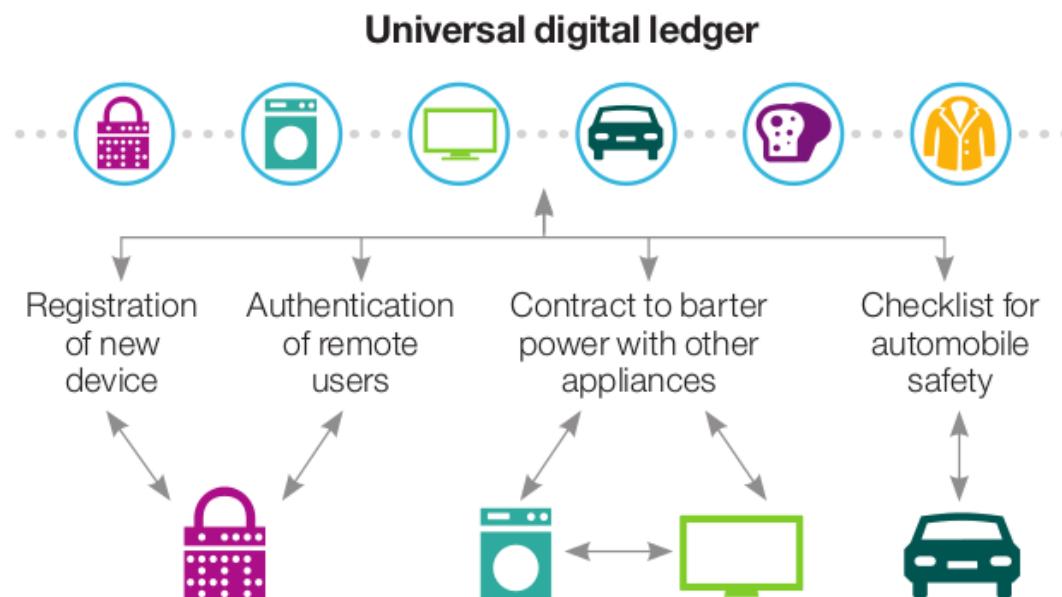






Цепочка блоков (**blockchain**) является непрерывным рядом блоков, которая содержит в себе полную историю операций. Каждый из этих блоков может содержать любой тип данных, которые разработчик счёл необходимыми в него включить.

Система платежей в Bitcoin организована таким образом, что транзакции не подтверждаются, пока не будут коллективными усилиями сети упакованы в последовательность блоков. Блок представляет собой запись последних транзакций, которые ещё не были записаны в предыдущие блоки. Он делится на заголовок и список транзакций. Заголовок блока включает в себя свой хеш SHA-256, хеш предыдущего блока из цепочки, список хешей транзакций, время создания блока и другую служебную информацию.



II. Арифметические основы ЭВМ

Системой счисления называется совокупность правил для представления чисел с помощью символов (цифр).

Позиционная система счисления:

$$(\dots a_3 a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3} \dots) = \dots + a_3 b^3 + a_2 b^2 + a_1 b^1 + a_0 + a_{-1} b^{-1} + a_{-2} b^{-2} + a_{-3} b^{-3}$$

Системы счисления, используемые в ЭВМ:

- Двоичная (0,1)
- Десятичная (0,...,9)
- Восьмеричная (0,...,7)
- Шестнадцатирична (0,...,9,A,B,C,D,E,F)
- Двоично-десятичная (0000,...,1001)
- Шестидесятирична (0,...,59)
- Троичная (-1,0,1)

Преобразование из двоичной системы счисления в десятичную:

$$1011.01_2 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 + 0 * 2^{-1} + 1 * 2^{-2} = (8 + 2 + 1 + 0.25)_{10} = 11.25_{10}$$

Преобразование из двоичной системы счисления в восьмеричную:

$$10111101_2 = 010 \quad 111 \quad 101 = 275_8$$

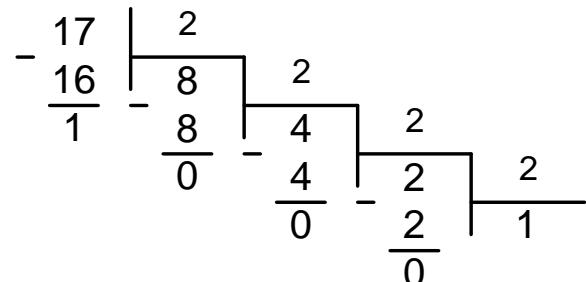
Преобразование из двоичной системы счисления в шестнадцатиричную:

$$10111101_2 = 10 \ 11 \quad 1101 = BD_{16}$$

Преобразование из десятичной системы счисления в двоичную:

Целая часть

$$17,95_{10} = 10001,11110\ldots_2$$



Младший
разряд

Старший
разряд

Дробная часть

$$\begin{array}{r}
 \times .95 \\
 \times 2 \\
 \hline
 \text{Старший} \\
 \text{разряд} \\
 \times 1 .90 \\
 \hline
 2 \\
 \times 1 .80 \\
 \hline
 2 \\
 \times 1 .60 \\
 \hline
 2 \\
 \times 1 .20 \\
 \hline
 \text{Младший} \\
 \text{разряд} \\
 \times 2 \\
 \hline
 0 .40
 \end{array}$$

...

Двоичная арифметика

$$0+0=0$$

$$0*0=0$$

$$0+1=1$$

$$0*1=0$$

$$1+0=1$$

$$1*0=0$$

$$1+1=0$$

$$1*1=1$$

Пример сложения и умножения

$$\begin{array}{r}
 10010111 \\
 + 10011010 \\
 \hline
 100110001
 \end{array}$$

$$\begin{array}{r}
 \times 1011 \\
 \times 1101 \\
 \hline
 1011 \\
 0000 \\
 + 1011 \\
 \hline
 1011 \\
 \hline
 10001111
 \end{array}$$

Прямой, обратный и дополнительный коды

Прямой код

$$G_{\text{пр}} = \begin{cases} G, & \text{при } G \geq 0 \\ A+|G|, & \text{при } G \leq 0 \end{cases}$$

$A-B=A+(-B)$
 G – n -разрядное число;
 A – вес старшего разряда
 $A = 2^{n-1}$ для целых и $A=1$ для дробей

Положительные
числа

$$10_{10} = 01010_2$$

$$0.75_{10} = 0.110_2$$

Отрицательные
числа

$$-10_{10} = 11010_2 = 10000+01010$$

$$-0.75_{10} = 1.110_2 = 1.000+0.110$$

Обратный код

$$G_{\text{обр}} = \begin{cases} G, & \text{при } G \geq 0 \\ B-|G|, & \text{при } G \leq 0 \end{cases}$$

G – n -разрядное число;
 B – наибольшее число без знака
 $B = 2^n-1$ для целых и $B=2-2^{-(n-1)}$ для дробей

Положительные
числа

$$10_{10} = 01010_2$$

$$0.75_{10} = 0.110_2$$

Отрицательные
числа

$$-10_{10} = 10101_2 = 11111-01010$$

$$-0.75_{10} = 1.001_2 = 1.111-0.110$$

Дополнительный

код

$$G_{\text{доп}} = \begin{cases} G, & \text{при } G \geq 0 \\ C - |G|, & \text{при } G < 0 \end{cases}$$

Положительные
числа

$$10_{10} = 01010_2$$

$$0.75_{10} = 0.110_2$$

G – n -разрядное число;

C – наибольшее число без знака + 1

$C = 2^n$ для целых и $C=2$ для дробей

Отрицательные
числа

$$-10_{10} = 10110_2 = 100000-01010$$

$$-0.75_{10} = 1.010_2 = 10.000-0.110$$

Переполнение при сложении чисел в дополнительном коде определяется, если перенос в знаковый разряд не вызывает перенос из знакового разряда, и перенос из знакового разряда не вызван переносом в знаковый

$$\begin{array}{r} 00,1111 \\ + 00,0001 \\ \hline 01,0000 \end{array}$$

Переполнение

$$\begin{array}{r} 01,0010 \\ + 01,1100 \\ \hline 10,1110 \end{array}$$

Переполнение

$$\begin{array}{r} 01,1111 \\ + 00,0001 \\ \hline 10,0000 \end{array}$$

Нет переполнения

- Числа в ЭВМ:

Числа с фиксированной запятой (позиция разделителя дробной и целой части заранее определена)

Числа с плавающей запятой (позиция разделителя определяется с помощью порядка числа)

Числа с плавающей запятой:

Пример:

$$\begin{aligned} X &= S^P * q \\ q &- мантисса\ числа\ X; \\ P &- порядок\ числа \\ S &- основание\ характеристики \\ (\text{для}\ \text{двоичной}\ \text{системы}\ S=2); \\ S^P &- характеристика \end{aligned}$$
$$\begin{aligned} 0,0110000 * 10^{011}_2 &= 0,375 * 2^3_{10} = \\ &= 0.0011000 * 10^{100}_2 = 0.1100000 * 10^{010}_2 = \\ &= 0.75 * 2^2_{10} \end{aligned}$$

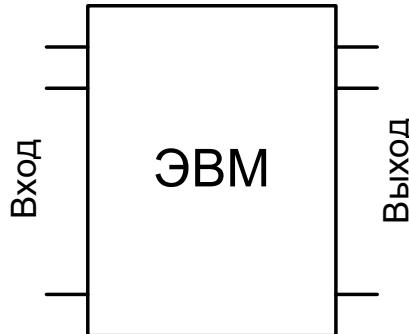
Для представления порядка используется смещенный код, в котором знаковый разряд инвертирован. Это позволяет легко сравнивать порядки чисел

- Сравнение числе с Ф.З и с П.З.:

У Ч.П.З. Большой диапазон представления

Арифметика над Ч.П.З. более сложная

III. Логические основы цифровой вычислительной техники



Любую ЭВМ можно рассматривать как сложное устройство, на вход которого подается входная информация в определенной последовательности. При этом на выходе должна формироваться ожидаемая выходная информация

- ЭВМ состоит из взаимодействующих устройств, задачей которых является преобразование входной информации в выходную.

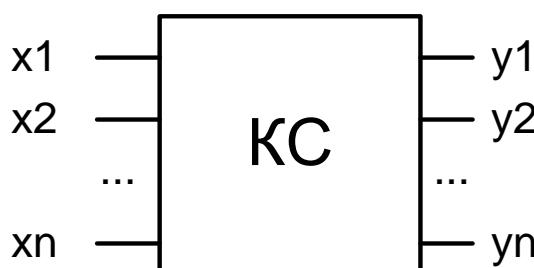
Такие устройства бывают двух типов:

Комбинационные схемы

Цифровые автоматы

Комбинационные схемы

$$y=f(x)$$



Цифровые

автоматы

Цифровые автоматы представляют собой комбинационные схемы и устройства хранения (память).

Работа цифровых автоматов происходит в соответствии с частотой поступления входного слова. Для того, чтобы сигналы поступали одновременно, срабатывание ЦА происходит по синхросигналу

Цифровые автоматы

Для задания ЦА необходимо определить:

- Входной алфавит: множество значений $x(t)$.
- Выходной алфавит: множество значений $y(t)$.
- Алфавит состояний: Q .
- Начальное состояния Q_0 .
- Функция переходов $A(Q, x)$.
- Функция выходов $B(Q, x)$.

Автомат Мили

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

Автомат Мура

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

Схема автомата Мили

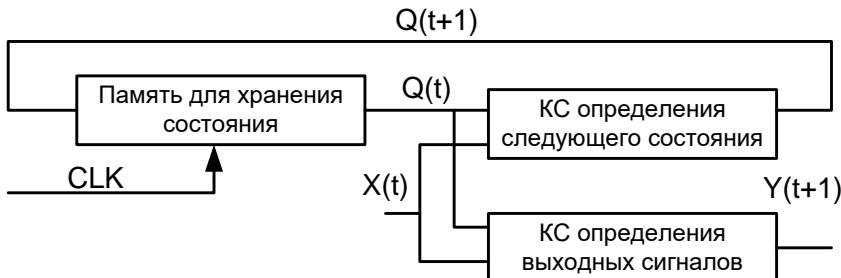
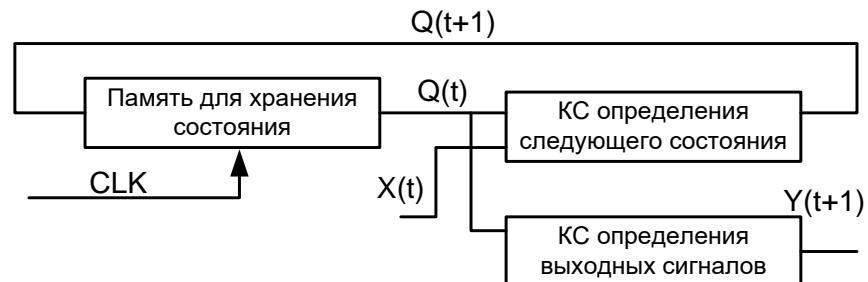


Схема автомата Мура



Проектирование комбинационных схем

Проектирование комбинационных схем заключается в определении выходного слова в виде функции алгебры логики от входного слова

Дизъюнктивной (конъюнктивной) нормальной формой называется равносильная ей формула, представляющая собой дизъюнкцию (конъюнкцию) элементарных конъюнкций (дизъюнкций).

N=3		B	\bar{B}
A	\bar{A}	0 1 0 0	0 0 0 0
\bar{C}	C	1 0 1 0	1 0 0 0
$Y = ABC + \bar{A}\bar{B}C = BC$			

Любую функцию можно образовать посредством базисных операций: Отрицания, дизъюнкции и конъюнкции.

ДНФ и КНФ не являются самым простым способом задания ФАЛ. Для минимизации нормальных форм применяют карты Карно

		B	\bar{B}		
		A	\bar{A}	\bar{D}	D
		\bar{C}	C	\bar{D}	D
1	0	0	1	0	0
0	1	1	1	0	0
0	1	1	1	0	0
1	0	0	1	0	0

$Y = DC + \bar{D}\bar{C}$

Логические функции

A	0	0	1	1	Обозначение функции	Название функции
B	0	1	0	1		
	0	1	1	1	$A \cup B$	Дизъюнкция
	0	0	0	1	$A \cap B$	Конъюнкция
	1	1	0	0	$\neg A$	Отрицание A
	0	0	1	0	$\neg A \rightarrow B$	Запрет $\neg A \rightarrow B$
	0	1	0	0	$\neg B \rightarrow A$	Запрет $\neg B \rightarrow A$
	0	1	1	0	$A \neg B$	Исключающее ИЛИ
	1	0	0	0	$A \downarrow B$	Стрелка Пирса ИЛИ-НЕ
	1	0	0	1	$A \sim B$	Равнозначность
	1	0	1	1	$B \rightarrow A$	Импликация от B к A
	1	1	0	1	$A \rightarrow B$	Импликация от A к B
	1	1	1	0	A / B	Штрих Шеффера И-НЕ

IV. Элементы и узлы ЭВМ

Триггеры

Триггер – логический элемент, который может находиться в одном из двух устойчивых состояний.

S, J – входы установки триггера в «1».

R, K – входы установки триггера в «0».

T – счетный вход триггера.

D – информационный вход триггера D

C – вход синхронизации

Q – прямой выход триггера

\bar{Q} – инверсный выход триггера

по логике:

- RS, D, T, JK

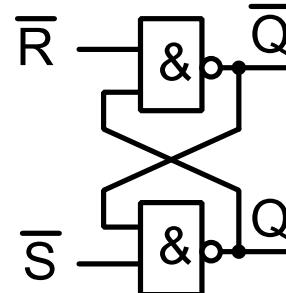
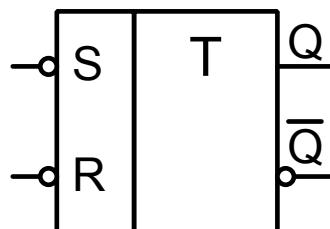
по способу приема:

- Асинхронные,
- Синхронные,
- Одноступенчатые,
- Двухступенчатые

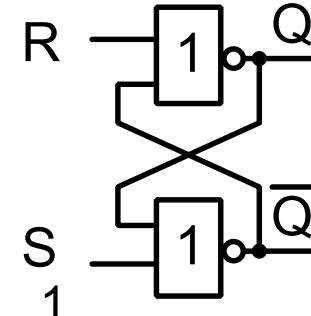
по способу синхронизации

- управляемые фронтом/спадом (динамические).
- управляемые уровнем (защелки)

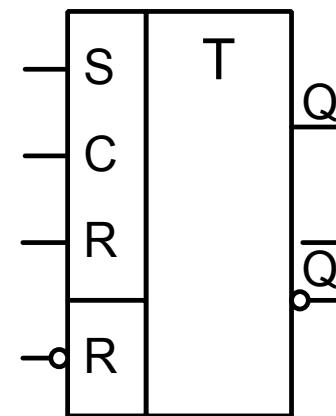
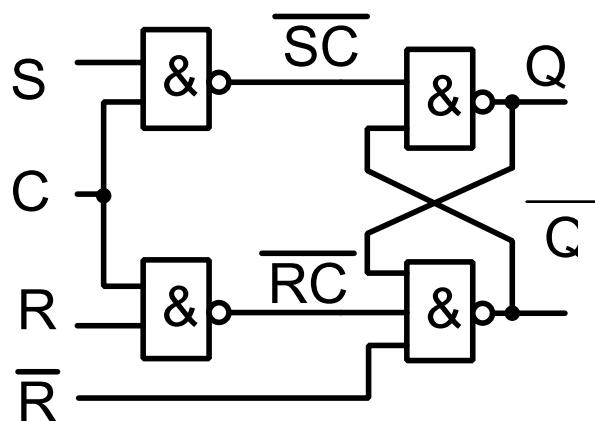
Одноступенчатый асинхронный RS-триггер



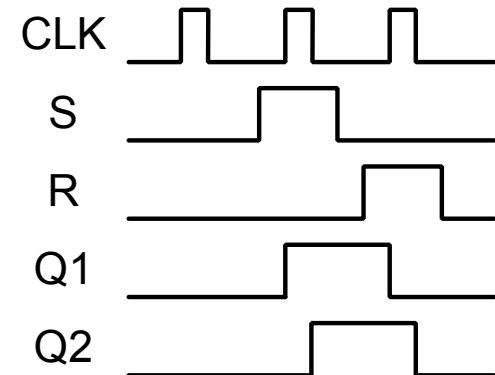
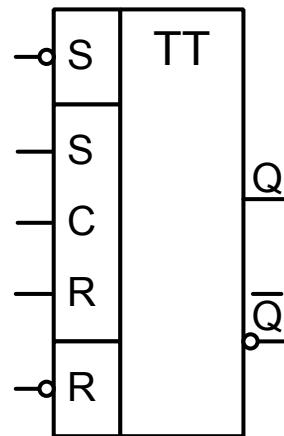
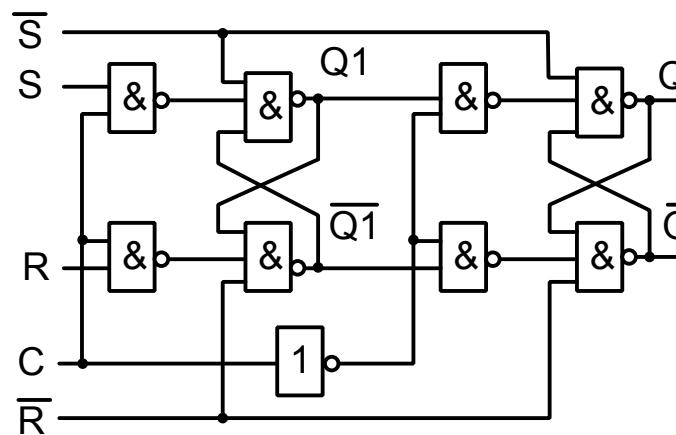
Архитектура ЭВМ



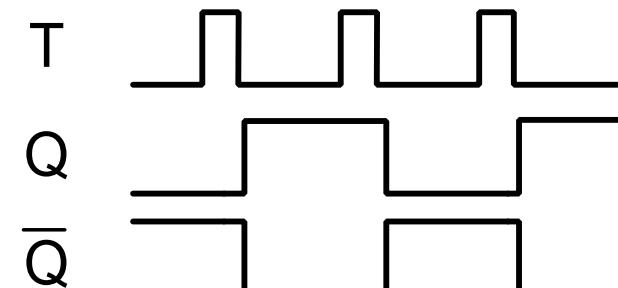
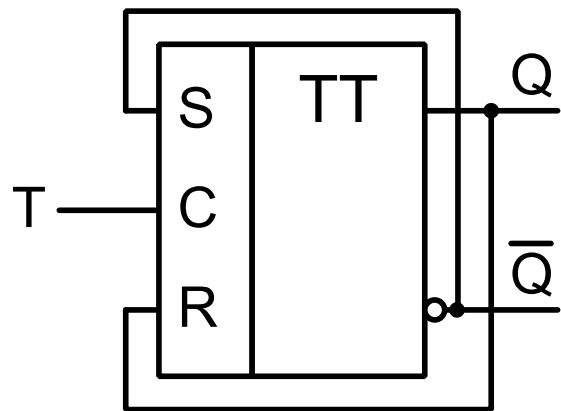
Одноступенчатый синхронный RS-триггер



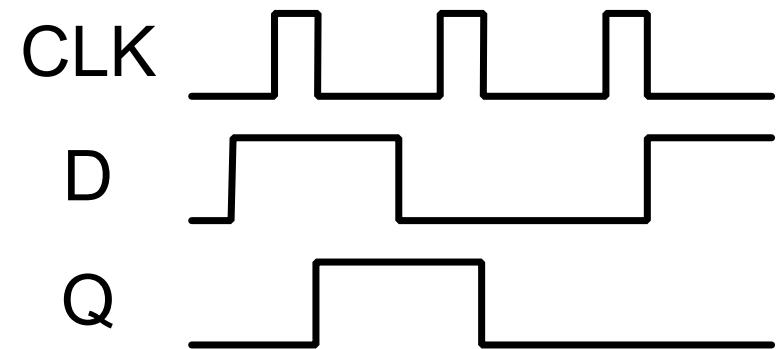
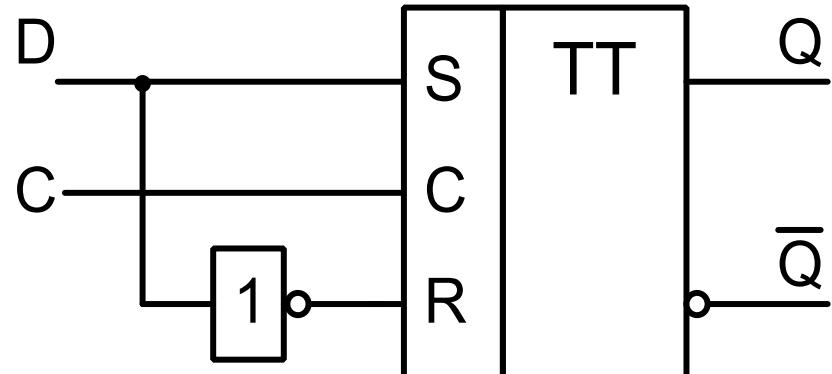
Двухступенчатый синхронный RS-триггер



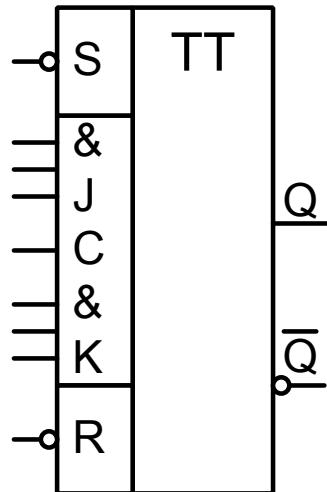
T-триггер



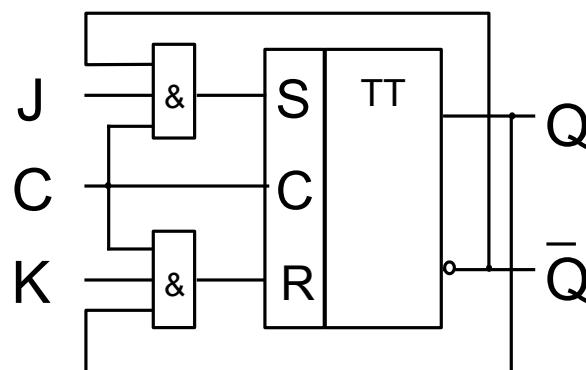
D-триггер



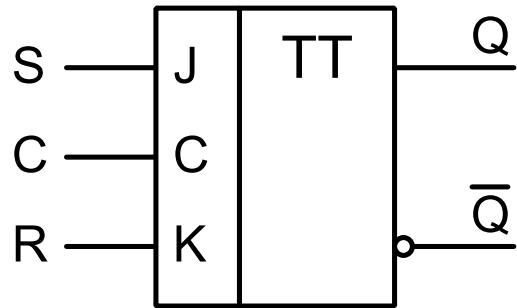
JK-триггер



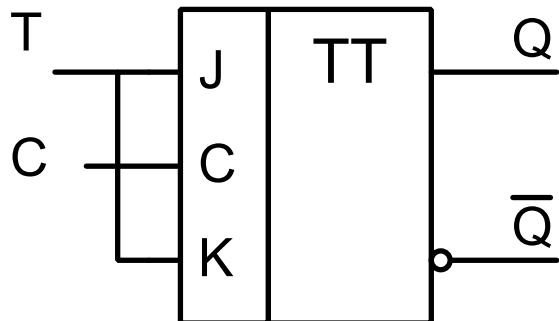
J(t)	K(t)	Q(t+1)	Режим
0	0	Q(t)	Хранение
0	1	0	Установка «0»
1	0	1	Установка «1»
1	1	Q(t)	Инверсия



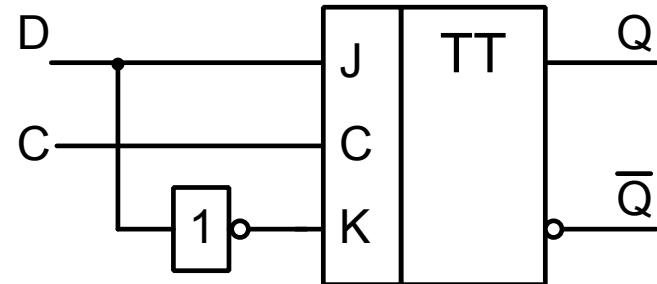
RS-триггер на основе JK-триггера



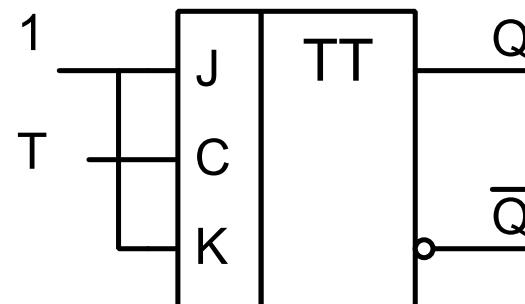
Синхронный Т-триггер на основе JK-триггера



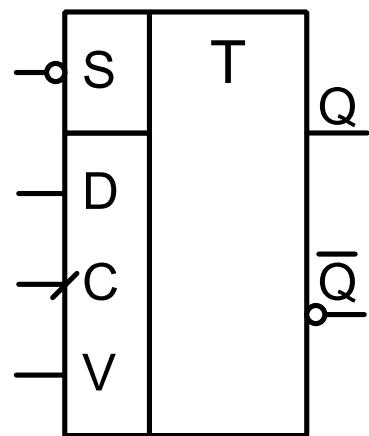
D-триггер на основе JK-триггера



Асинхронный Т-триггер на основе JK-триггера



DV-триггер



Динамические триггеры

Схема DV-триггера

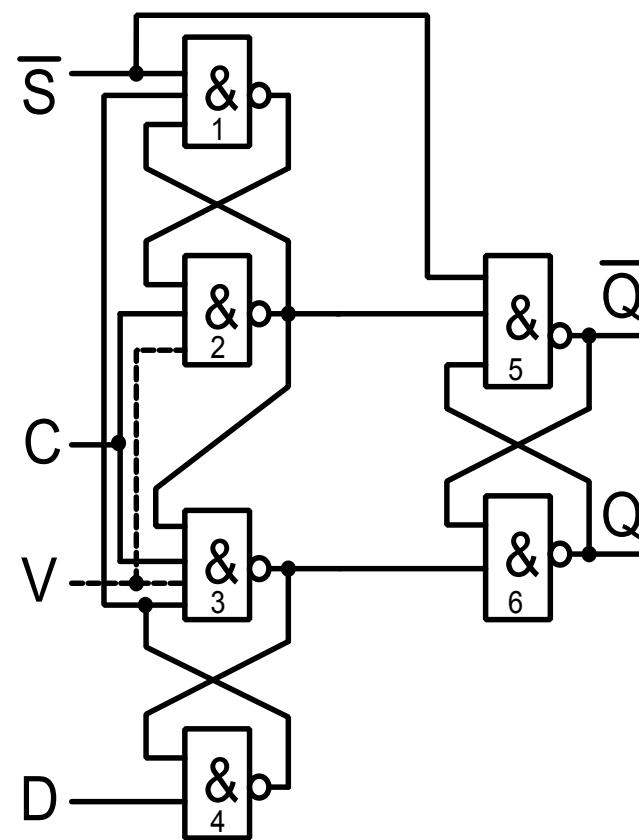


Диаграмма работы DV-триггера

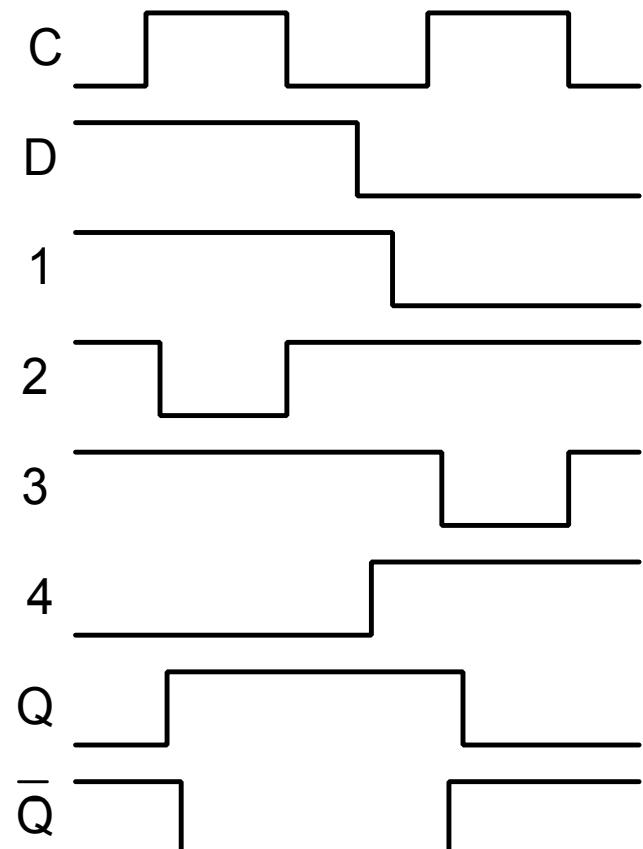
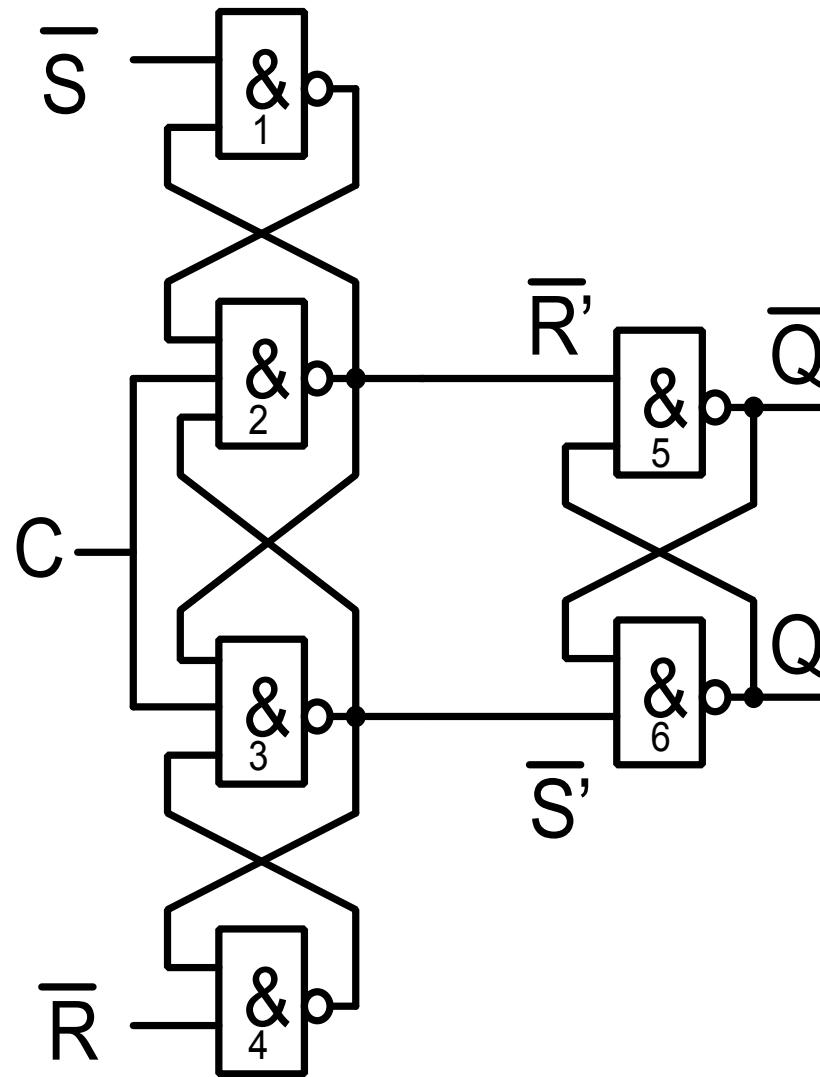
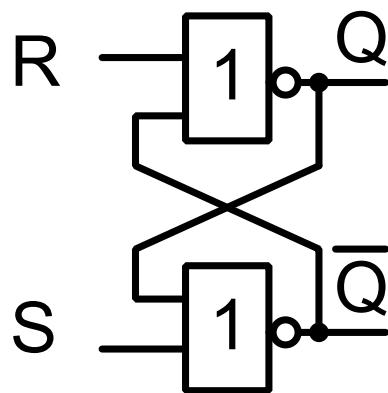


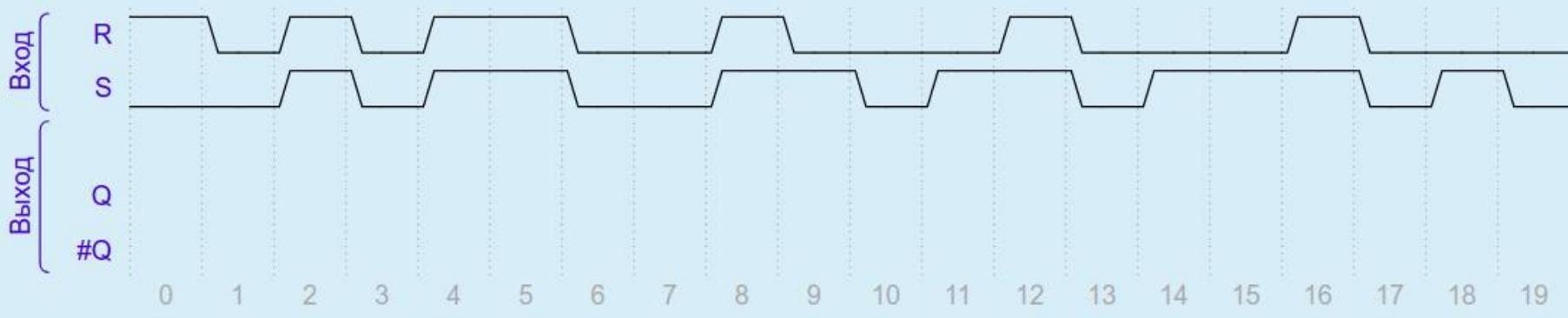
Схема трех триггеров



Одноступенчатый асинхронный RS-триггер на элементах ИЛИ-НЕ



Асинхронный RS триггер на элементах ИЛИ-НЕ.



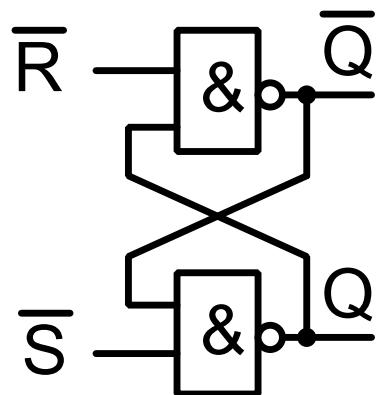
Нарисовать диаграмму выходных сигналов. "#" - инверсный сигнал.

В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

Q: 000X00XX01110X110X11 ✓

#Q: 110X00XX00000X000X00 ✓

Одноступенчатый асинхронный RS-триггер на элементах И-НЕ



Асинхронный RS триггер на элементах И-НЕ.



Нарисовать диаграмму выходных сигналов. "#" - инверсный сигнал.

В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

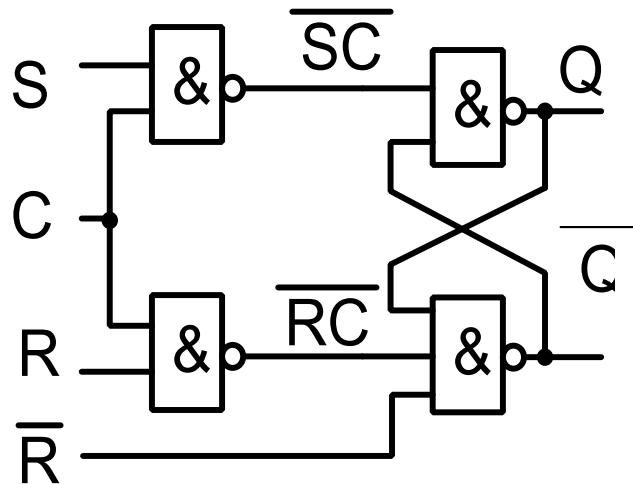
Q: 00100011011X11011100



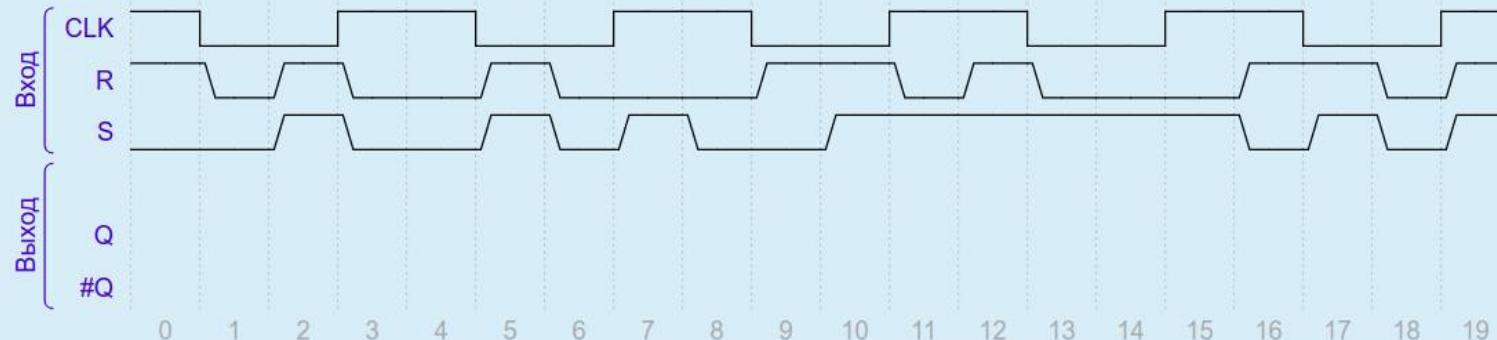
#Q: 11011110111X10101011



Одноступенчатый синхронный RS-триггер на элементах И-НЕ



Синхронный одноступенчатый RS триггер на элементах И-НЕ.



Нарисовать диаграмму выходных сигналов. "#" - инверсный сигнал.

В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).

Например: 0X11100X100000111101

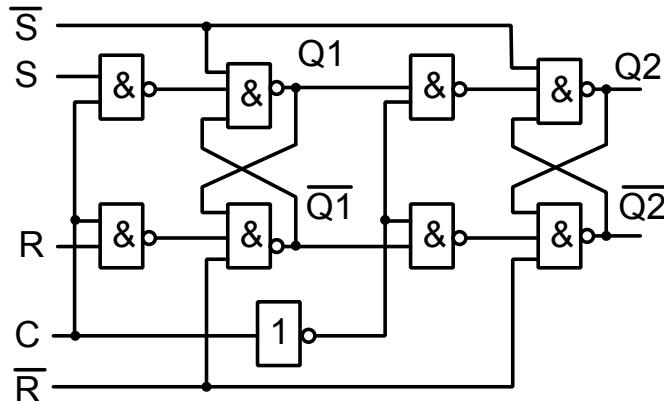
Q: 0000000111111XX10001



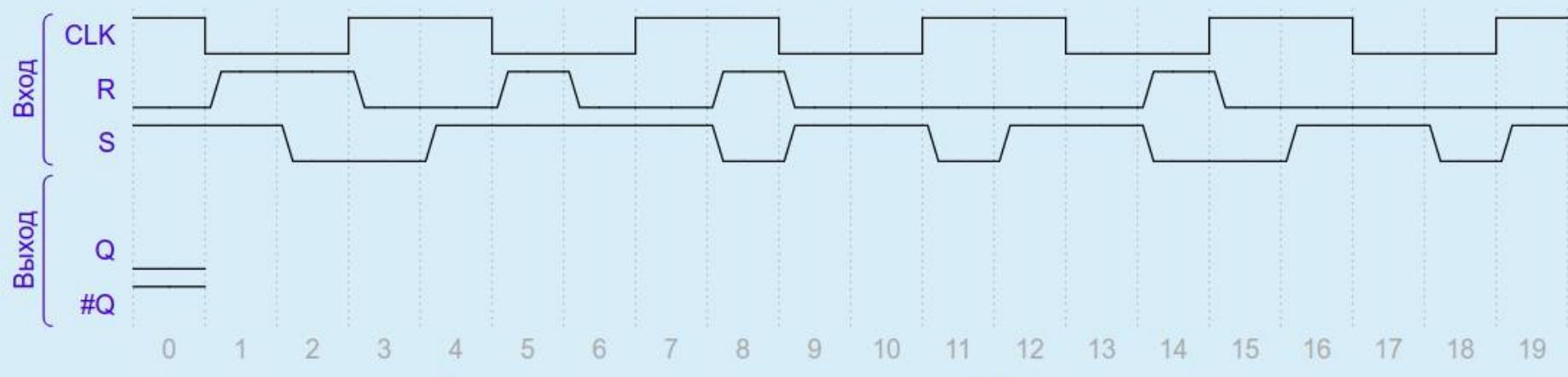
#Q: 1111111000001XX01111



Двухступенчатый синхронный RS-триггер на элементах И-НЕ



Синхронный двухступенчатый RS триггер на элементах И-НЕ.



Нарисовать диаграмму выходных сигналов. "#" - инверсный сигнал.

В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

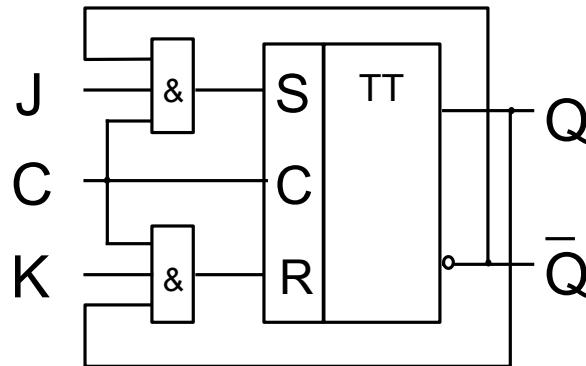
Q: 011111110000111111



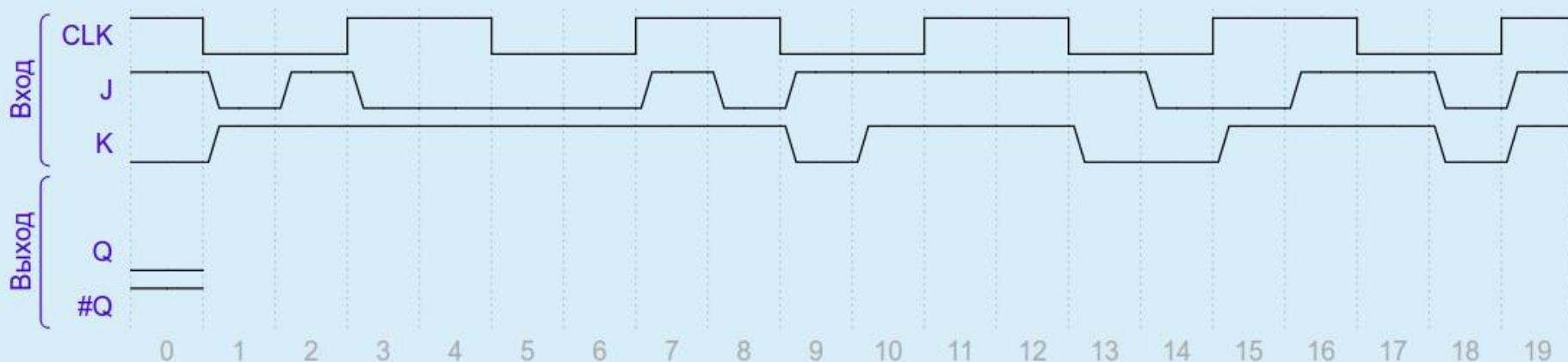
#Q: 10000000011110000000



Двухступенчатый синхронный JK-триггер на элементах И-НЕ



Синхронный двухступенчатый JK триггер на элементах И-НЕ.



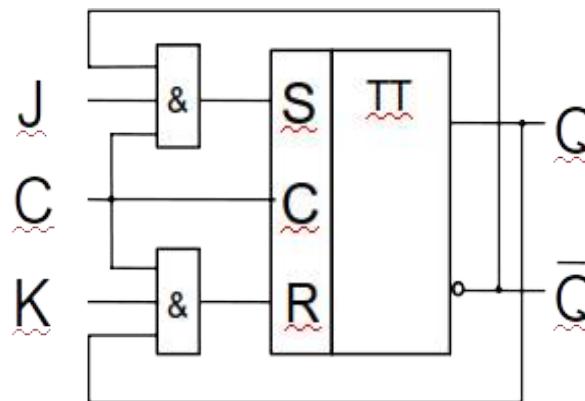
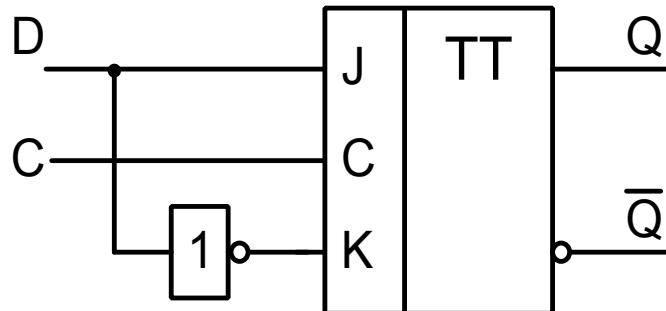
Нарисовать диаграмму выходных сигналов. "#" - инверсный сигнал.

В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

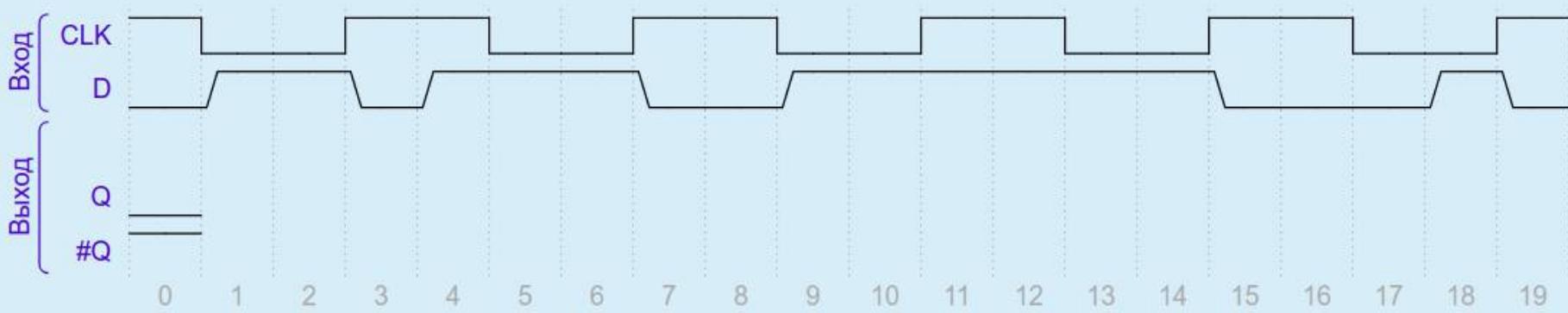
Q: 01111000000001111000 ✓

#Q: 1000011111110000111 ✓

Двухступенчатый синхронный D-триггер на элементах И-НЕ



Синхронный двухступенчатый D триггер на элементах И-НЕ.



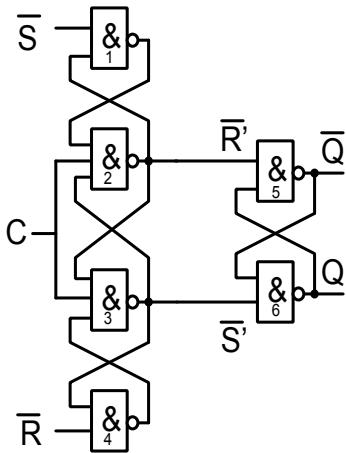
Нарисовать диаграмму выходных сигналов. "#" - инверсный сигнал.

В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

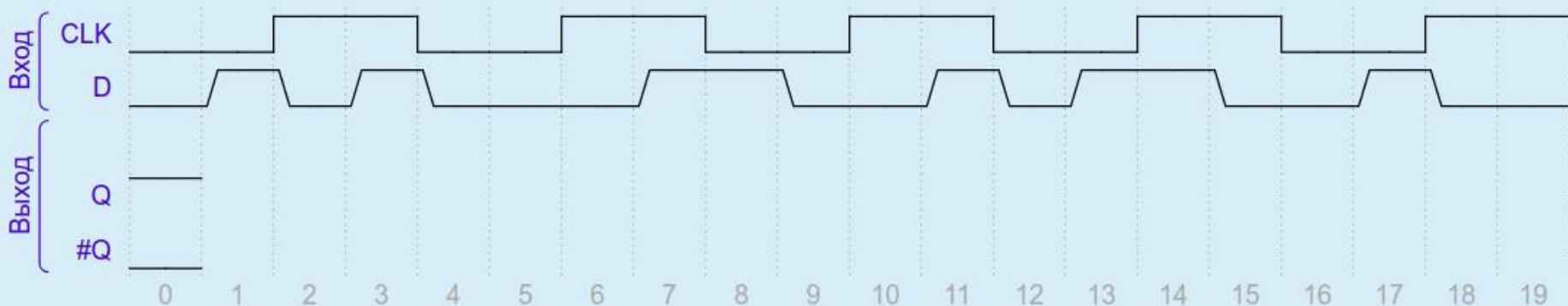
Q: 00000111100001111000 ✓

#Q: 11111000011110000111 ✓

Динамический D-триггер на элементах И-НЕ



Динамический D триггер.



Нарисовать диаграмму выходных сигналов. "#" - инверсный сигнал.

В ответе необходимо указать значения Q и $\#Q$ (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

Q: 11111100000000111111 ✓

#Q: 00000011111111000000 ✓

Регистры

Регистром называется устройство, предназначенное для запоминания слова, а также для выполнения над словом некоторых логических преобразований.

Операции, выполняемые регистром:

- Сброс (установка в 0);
- Прием слова (запись);
- Выдача слова (чтение);
- Сдвиг слова (сдвиг вправо, влево, циклический сдвиг);
- Преобразование параллельного кода в последовательный;
- Поразрядные логические операции.

Условное обозначение универсального регистра

<u>Сигнал</u>	<u>Тип</u>	<u>Описание</u>
CLK	Вход	Синхросигнал
RESET	Вход	Сброс
LOAD	Вход	Разр. загрузки
HOLD	Вход	Запр. сдвига
UP	Вход	Направление
DL	Вход	Младший бит
DU	Вход	Старший бит
D#	Вход	Входное слово
OE	Вход	Разрешение выдачи
Q#	Выход	Выходное слово
QL	Выход	Младший бит
QU	Выход	Старший бит

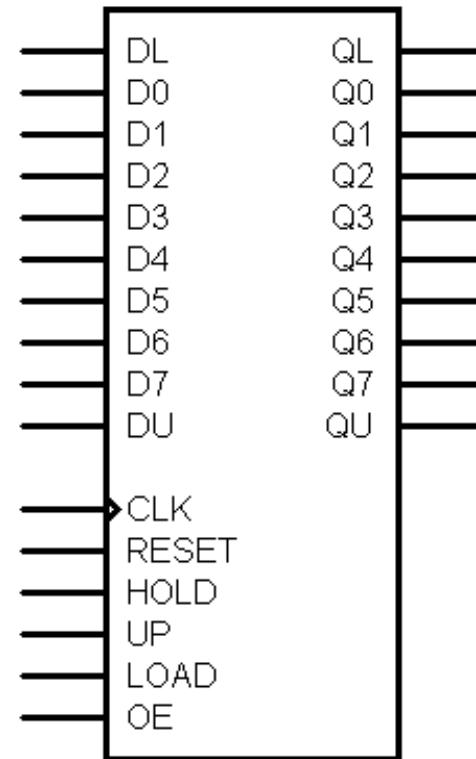
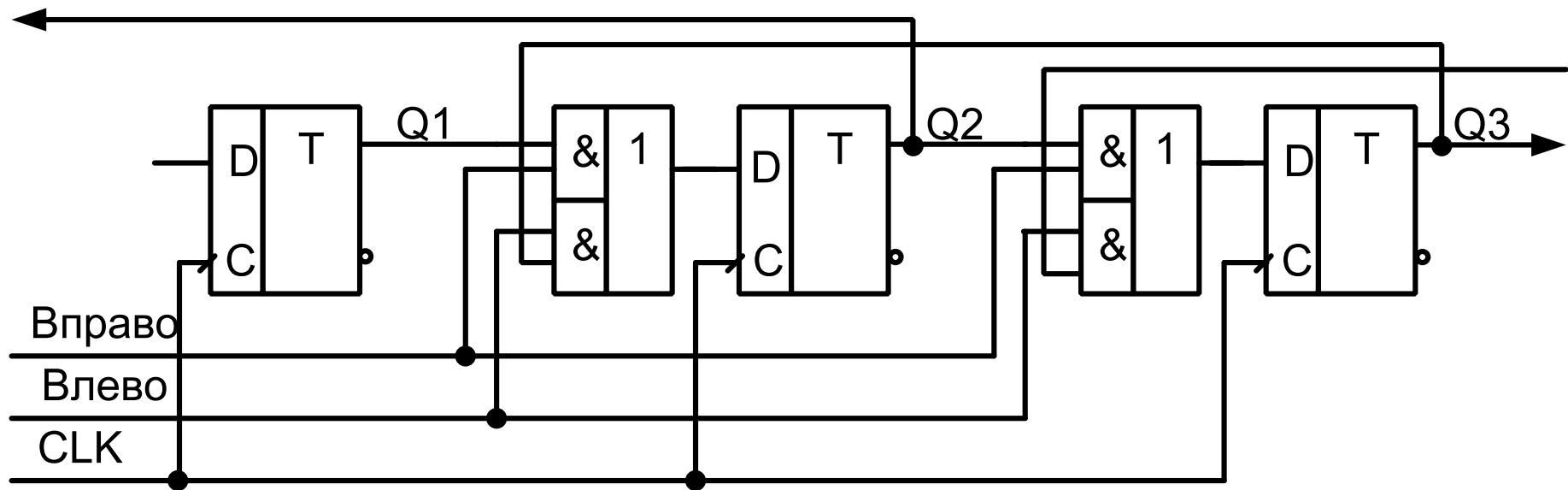


Схема сдвигового регистра



Счетчики

Счетчиком называется узел ЭВМ, предназначенный для подсчета входных сигналов.

Модуль счета: число возможных состояний счетчика.

Классификация счетчиков.

По способу счета: суммирующие, вычитающие, реверсивные.

По модулю счета: двоичные, десятичные,

По способу распространения переноса: с параллельным переносом, с последовательным переносом, с групповой структурой.

По способу синхронизации: асинхронные, синхронные.

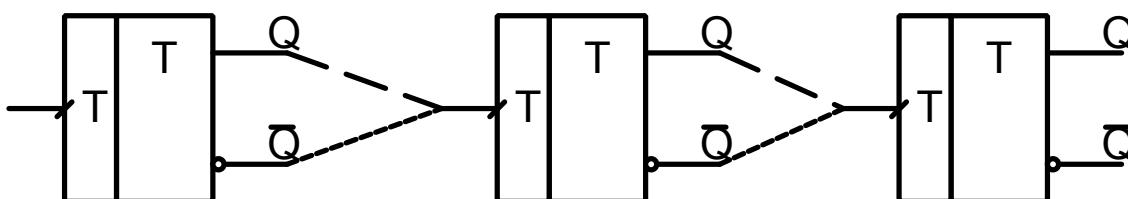
По режиму работы: для подсчета входных сигналов, для деления частоты.

Таблица состояний

$X_{\text{сч}}$	Q4	Q3	Q2	Q1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
...				
15	1	1	1	1

перенос

— — Обратный счет



— — Прямой счет

Счетчик с последовательным переносом

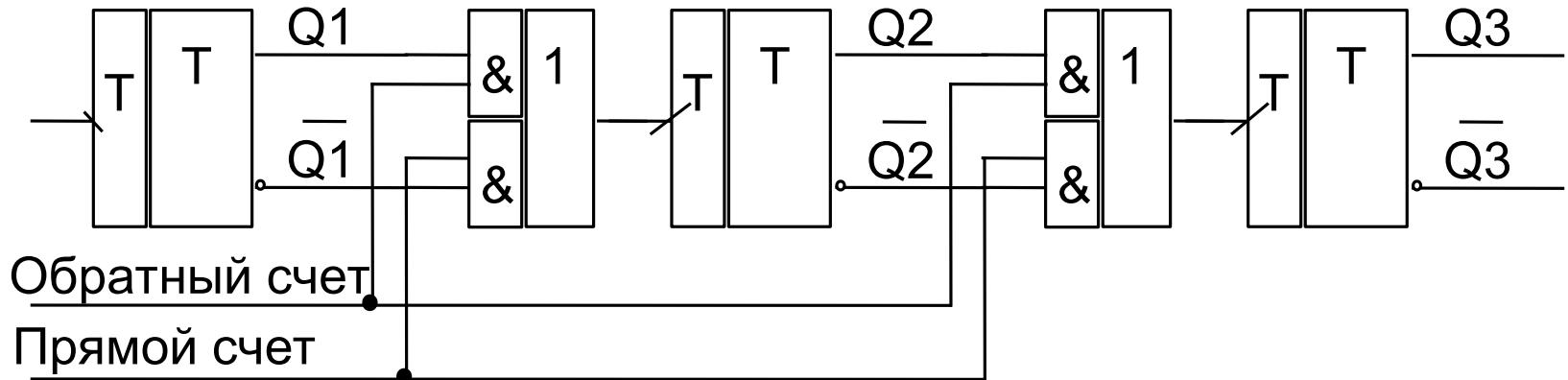
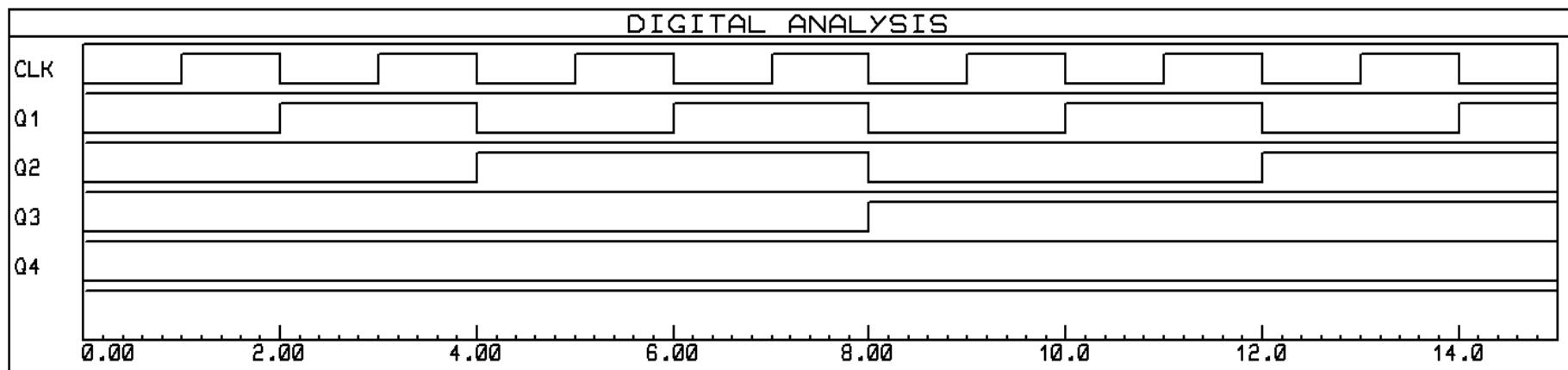


Диаграмма работы (прямой счет)



Счетчик с параллельным переносом

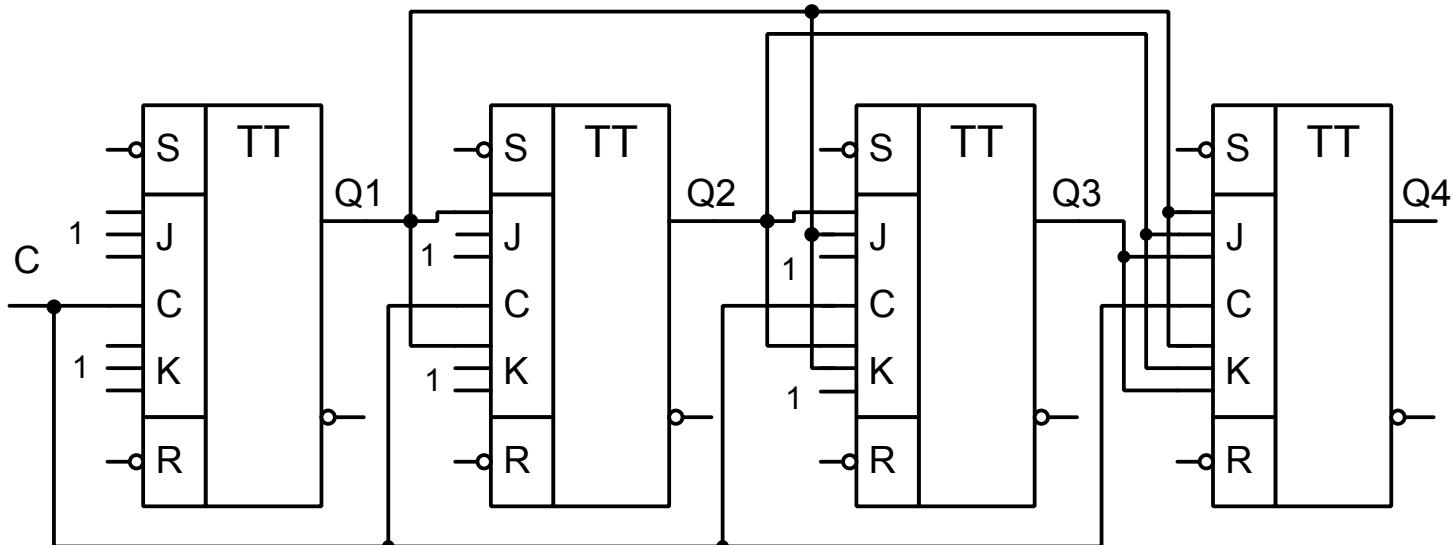
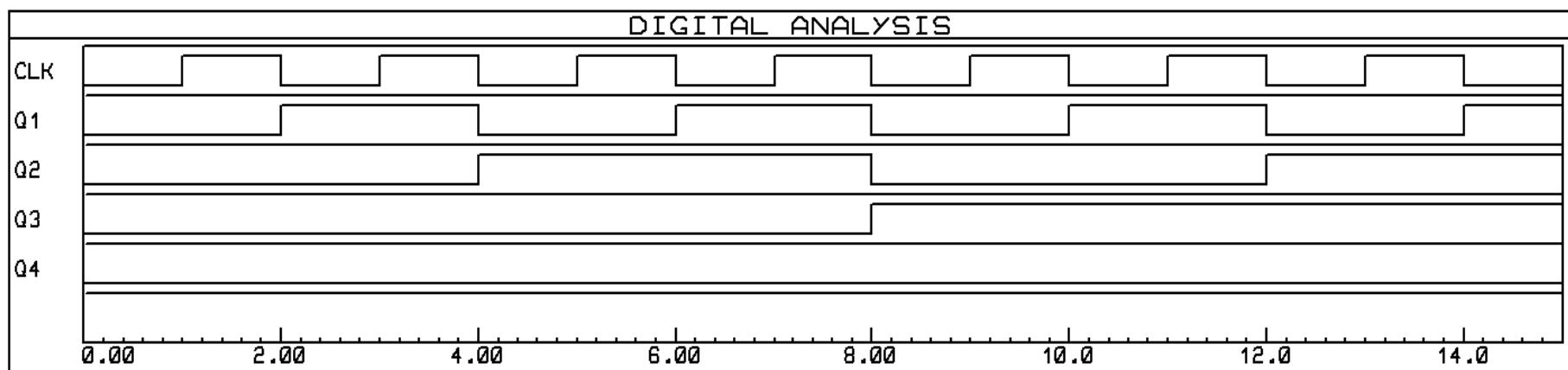


Диаграмма работы

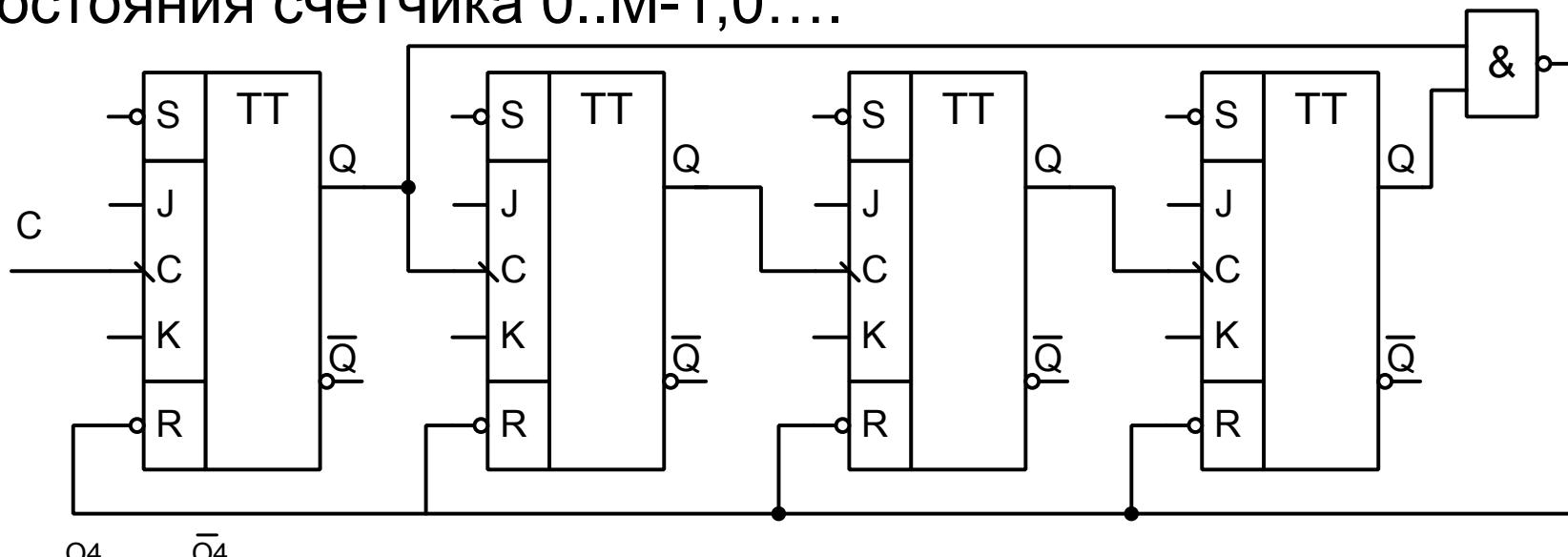


Построение счетчиков с произвольным модулем счета

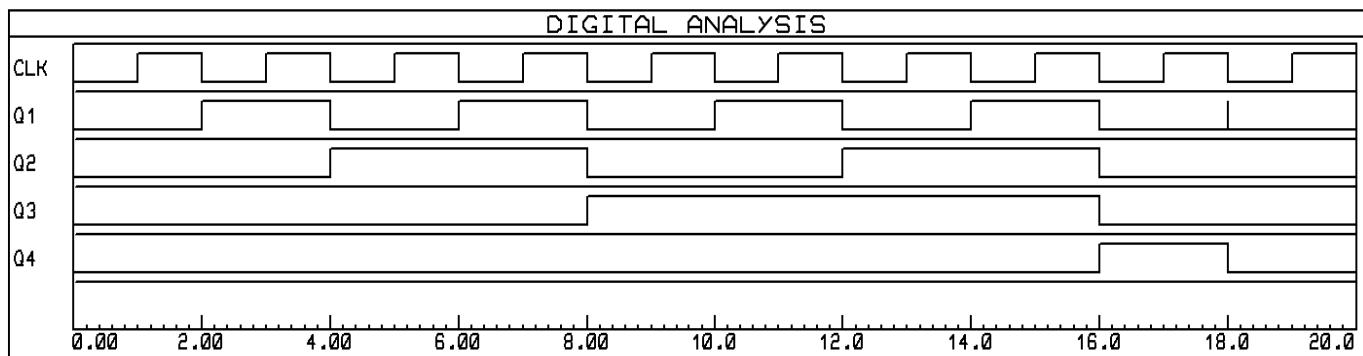
Метод управляемого сброса

Построить счетчик с модулем счета $M=9$.

Состояния счетчика $0..M-1, 0....$



	Q4	\bar{Q}_4		
Q3	x	x	0	0
\bar{Q}_3	1	x	0	0
	\bar{Q}_2	Q2	\bar{Q}_2	
	Q1		\bar{Q}_1	



Метод модификации связей

Построить счетчик с модулем счета $M=7$.

Состояния счетчика $0, 2, 3..7, 0, 2\dots$ (состояние «1» пропущено).

Таблица функционирования счетчика

		Функции возбуждения						
$Q(t)$	$Q(t+1)$	J_2	K_2	J_1	K_1	J_0	K_0	
000	010	0	x	1	x	0	x	
010	011	0	x	x	0	1	x	
011	100	1	x	x	1	x	1	
100	101	x	0	0	x	1	x	
101	110	x	0	1	x	x	1	
110	111	x	0	x	0	1	x	
111	000	x	1	x	1	x	1	

Минимизация функций возбуждения

J2	Q0	\bar{Q}_0	
Q1	1 x	x 0	
\bar{Q}_1	x x	x 0	
	\bar{Q}_2	Q2	\bar{Q}_2

$J2 = Q0$

J1	Q0	\bar{Q}_0	
Q1	x x	x x	
\bar{Q}_1	x 1	0 1	
	\bar{Q}_2	Q2	\bar{Q}_2

$J1 = Q0 + \bar{Q}_2$

J0	Q0	\bar{Q}_0	
Q1	x x	1 1	
\bar{Q}_1	x x	1 0	
	\bar{Q}_2	Q2	\bar{Q}_2

$J0 = Q1 + Q2$

K2	Q0	\bar{Q}_0	
Q1	x 1	0 x	
\bar{Q}_1	x 0	0 x	
	\bar{Q}_2	Q2	\bar{Q}_2

$K2 = Q0 \cdot Q1$

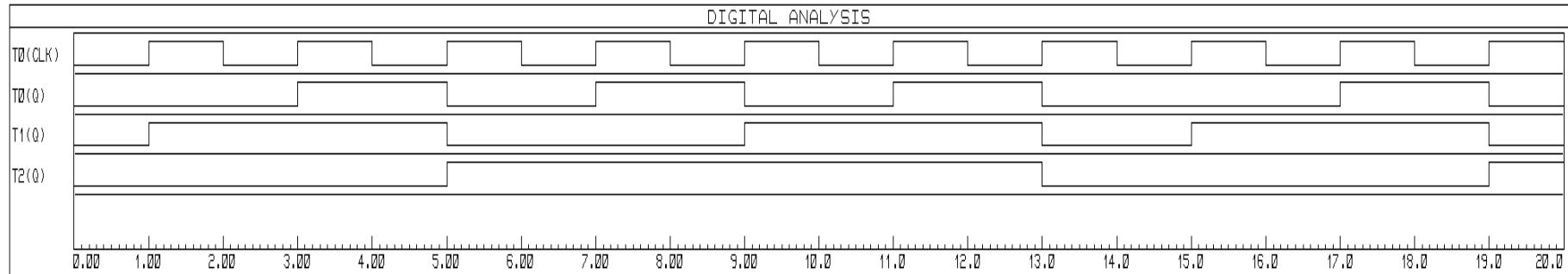
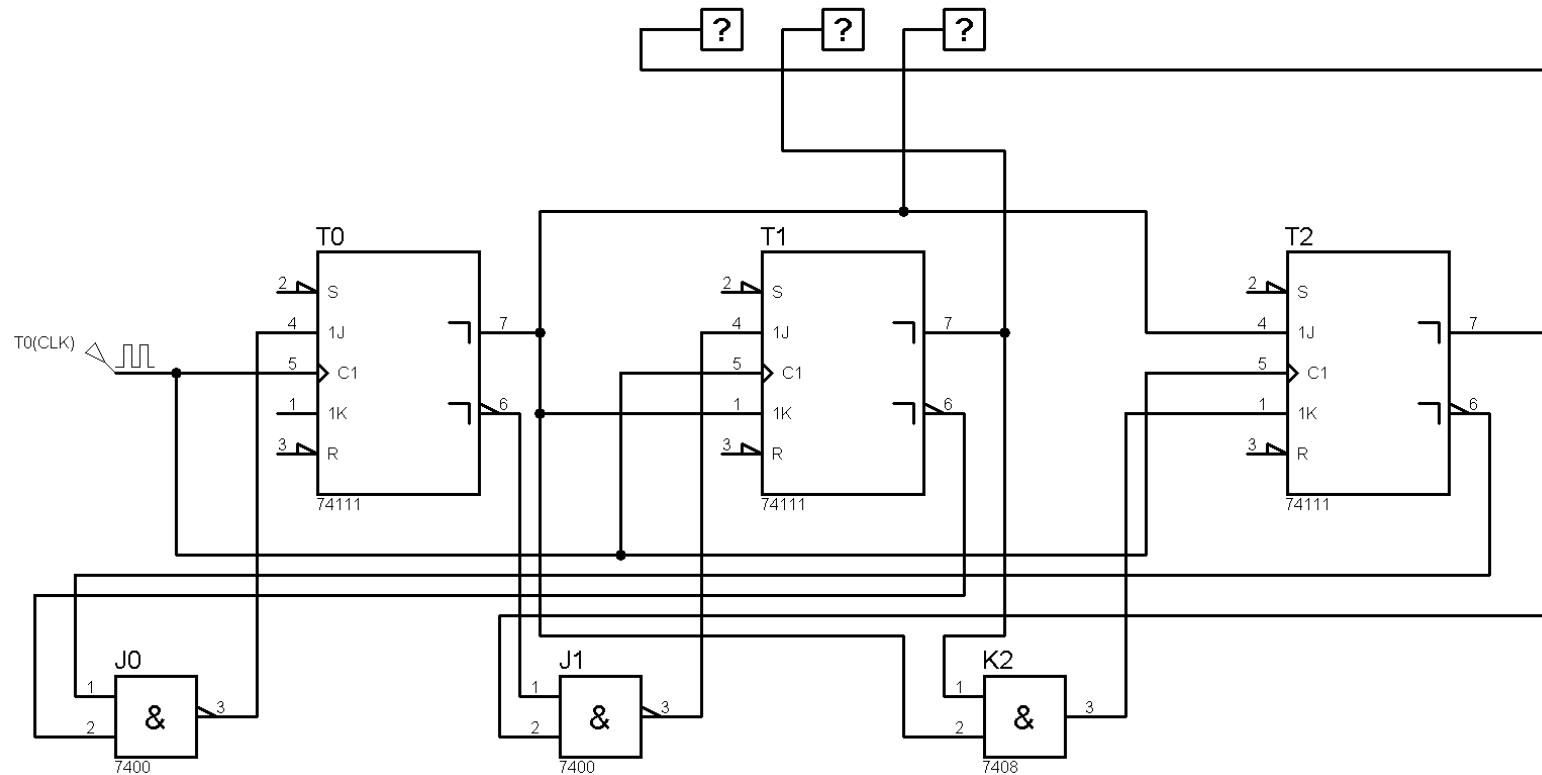
K1	Q0	\bar{Q}_0	
Q1	1 1	0 0	
\bar{Q}_1	x x	x x	
	\bar{Q}_2	Q2	\bar{Q}_2

$K1 = Q0$

K0	Q0	\bar{Q}_0	
Q1	1 1	x x	
\bar{Q}_1	x 1	x x	
	\bar{Q}_2	Q2	\bar{Q}_2

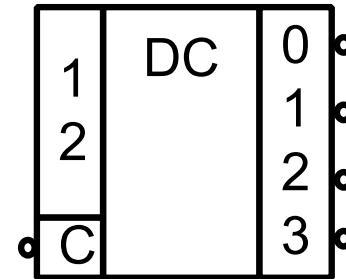
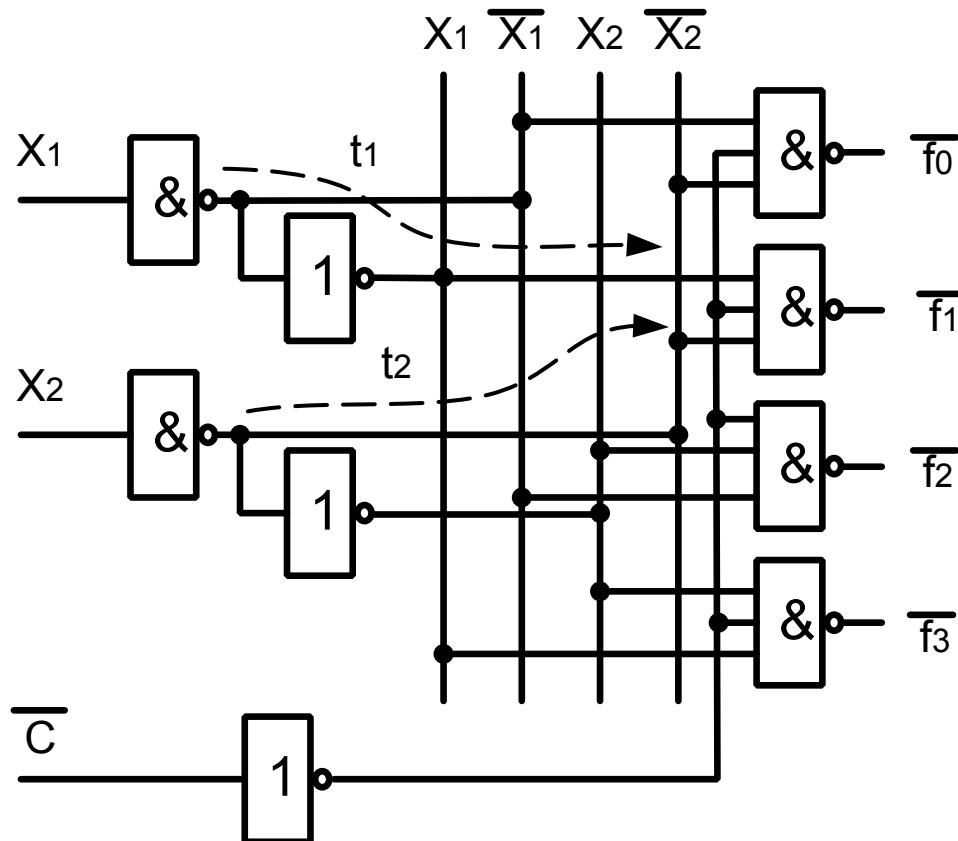
$K0 = 1$

Схема счетчика



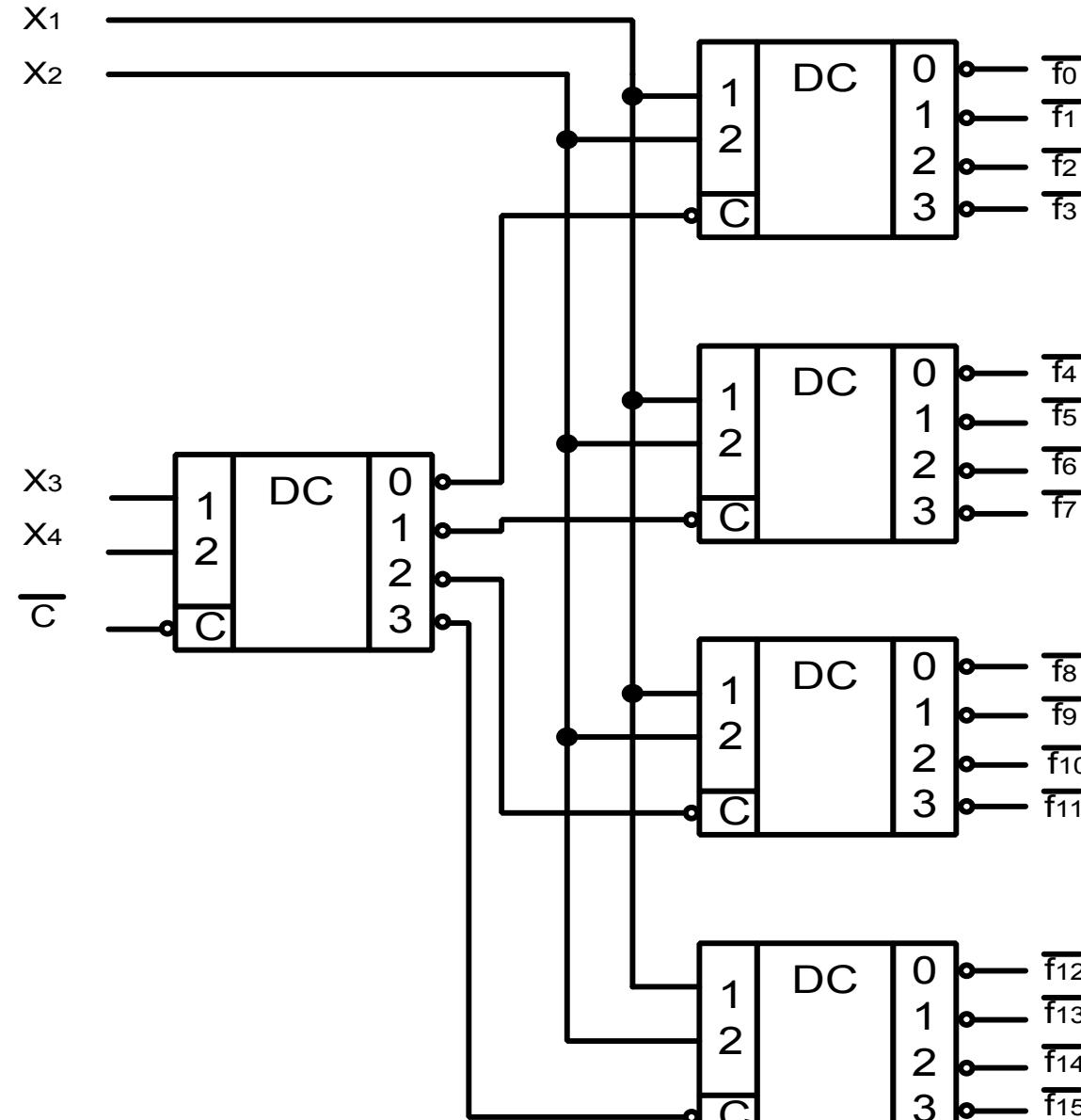
Дешифраторы

Дешифратором называется комбинационная схема, преобразующая код, подаваемый на входы, в сигнал на одном из выходов.



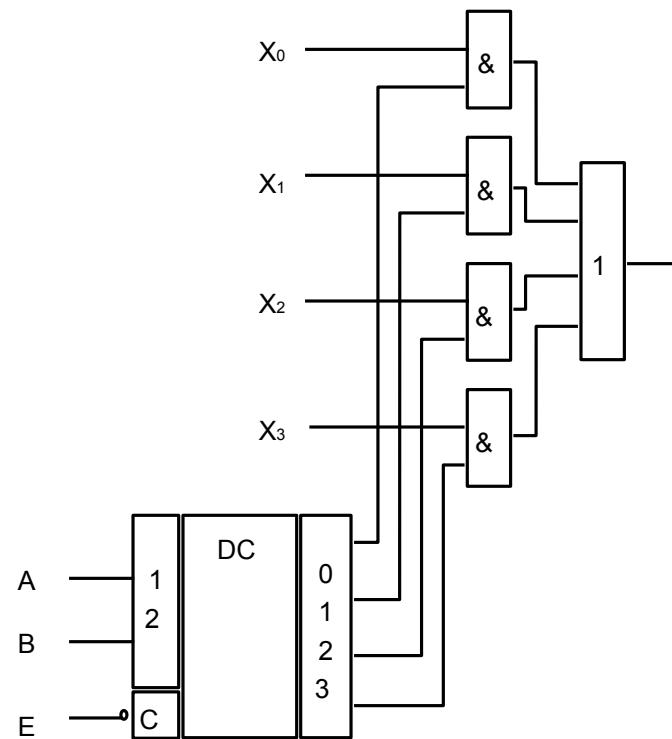
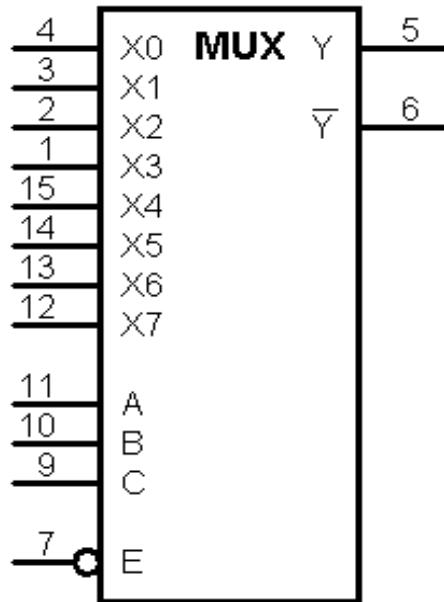
Статический риск:
кратковременное изменение
сигнала, который должен
остаться неизменным.
Динамический риск:
многократное переключение
элемента вместо ожидаемого
одночтожно.

Наращивание размерности дешифраторов

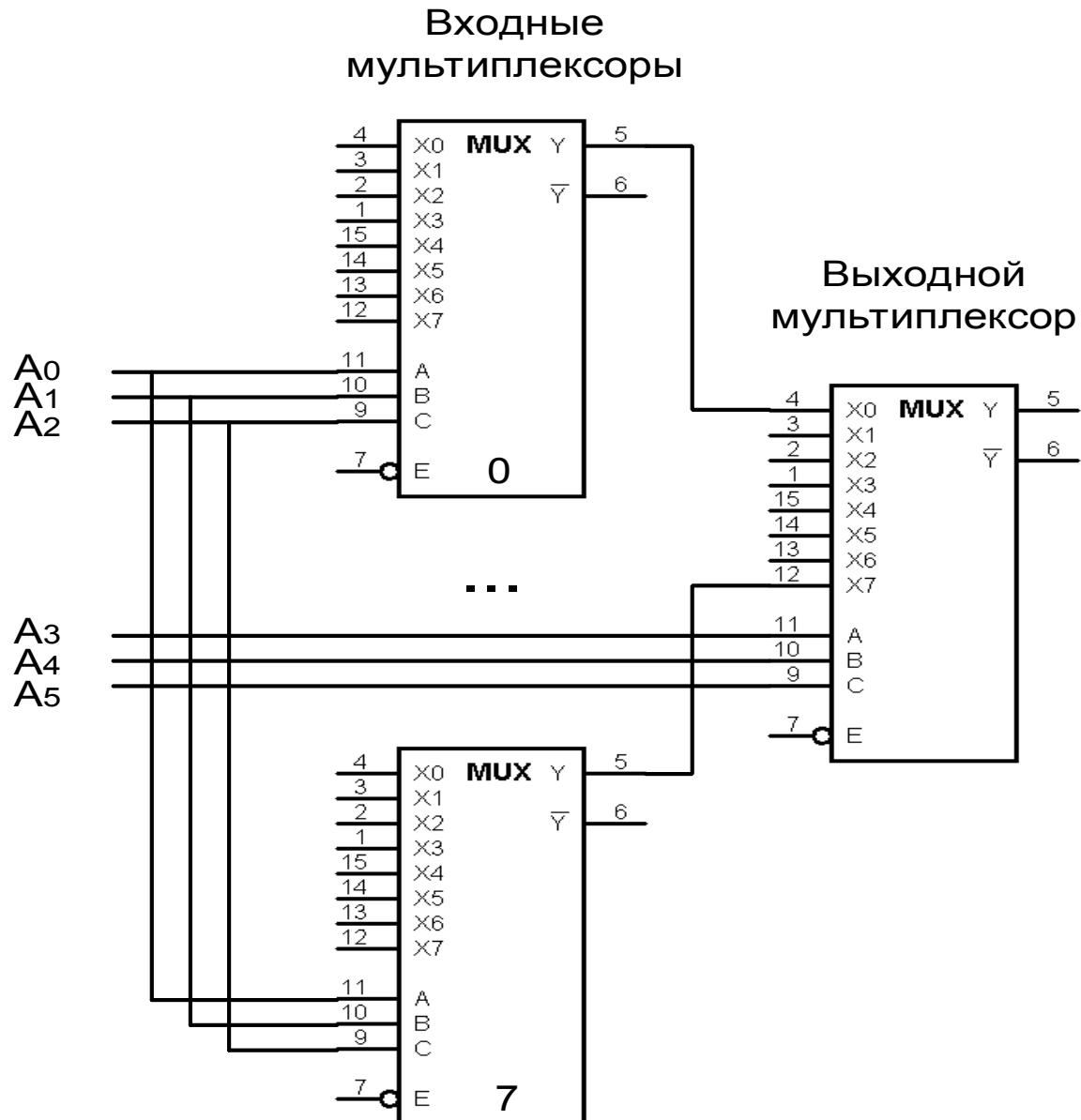


Мультиплексоры

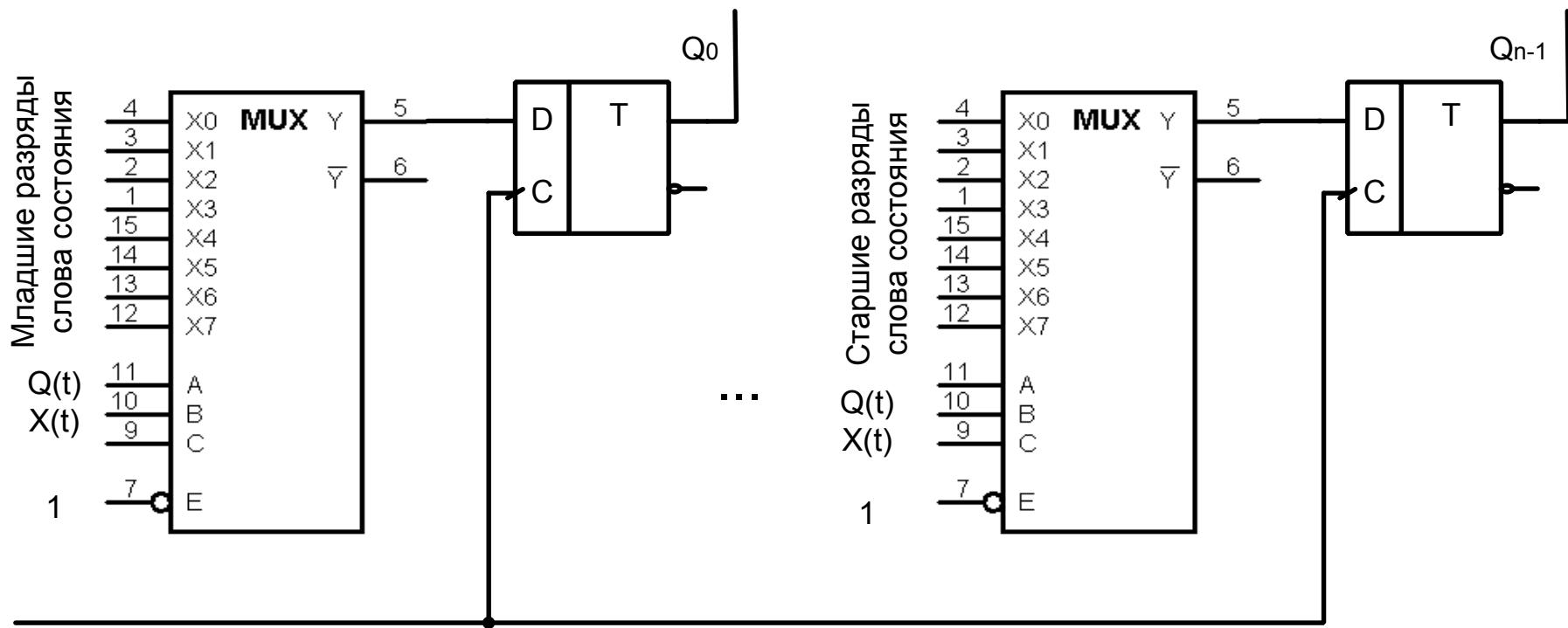
Мультиплексором называется комбинационная схема, осуществляющая передачу сигнала с одной из входных информационных линий на выход.



Наращивание размерности мультиплексоров

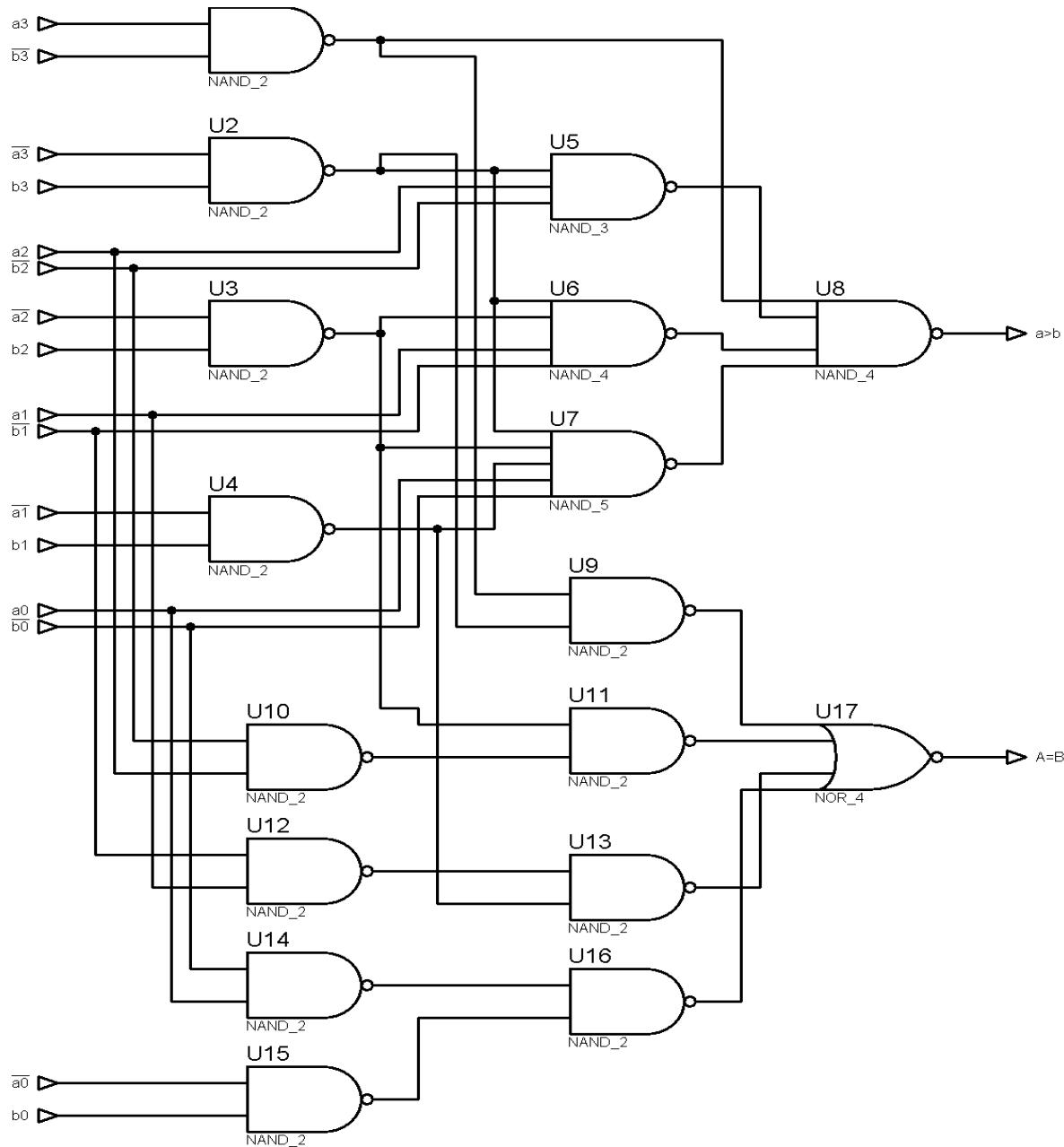
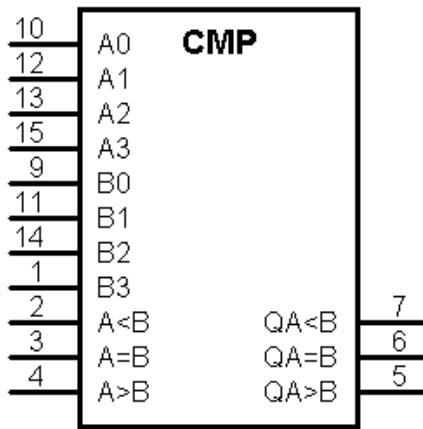


Пример построения автомата с помощью мультиплексоров



Компаратор

Компаратором называется комбинационная схема, определяющая отношение между двумя словами.



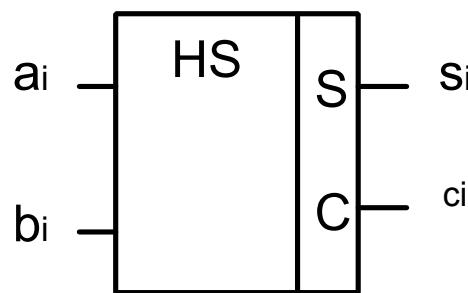
Сумматоры

Сумматором называется узел ЭВМ, выполняющий арифметическое сложение кодов чисел.

$$S_i = a_i \bar{b}_i \bar{c}_{i-1} \cup \bar{a}_i b_i \bar{c}_{i-1} \cup \bar{a} \bar{b}_i c_{i-1} \cup a_i b_i c_{i-1}$$

$$c_i = a_i b_i \bar{c}_{i-1} \cup a_i \bar{b}_i c_{i-1} \cup \bar{a} \bar{b}_i c_{i-1} \cup a_i b_i c_{i-1} = a_i b_i \cup a_i c_{i-1} \cup b_i c_{i-1}$$

Полусумматор выполняет арифметическое сложение кодов двух чисел.



Сумматор выполняет арифметическое сложение кодов двух чисел с учетом переноса в младший разряд.

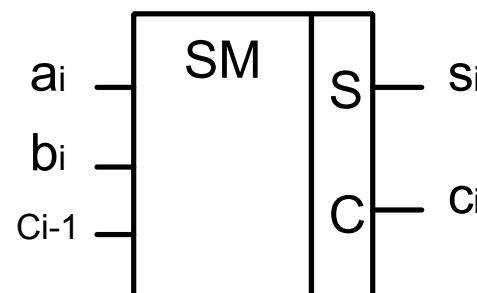


Схема одноразрядного сумматора

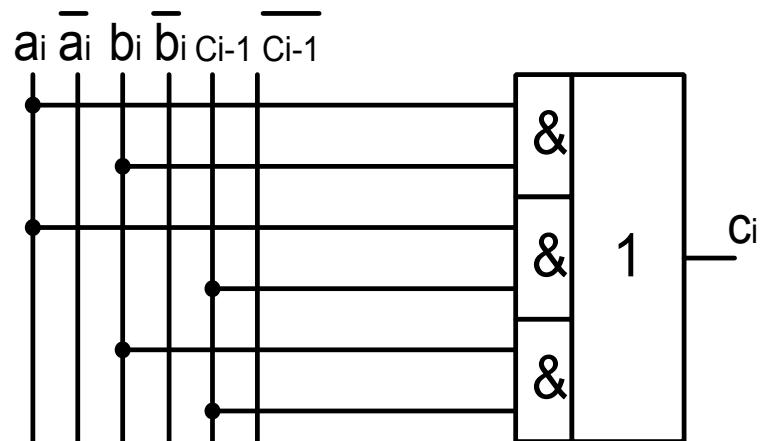


Схема параллельного сумматора с последовательным переносом

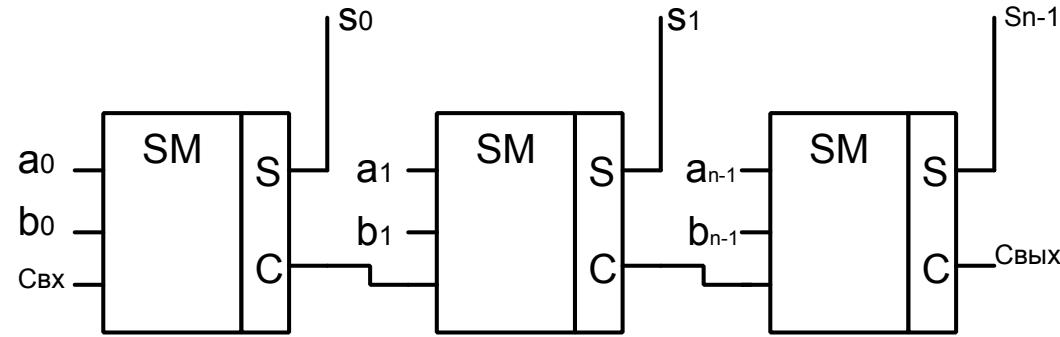


Схема параллельного сумматора с параллельным переносом

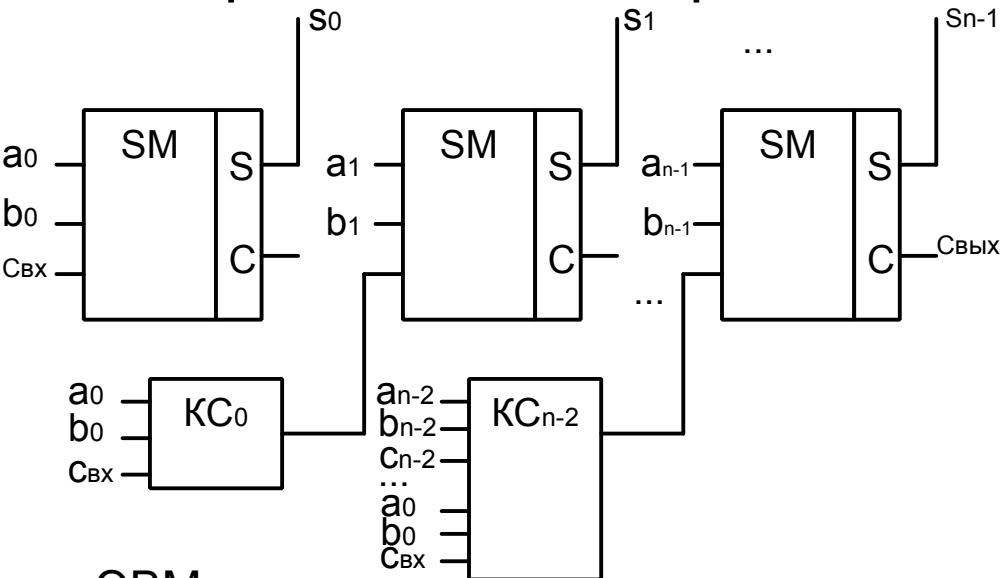
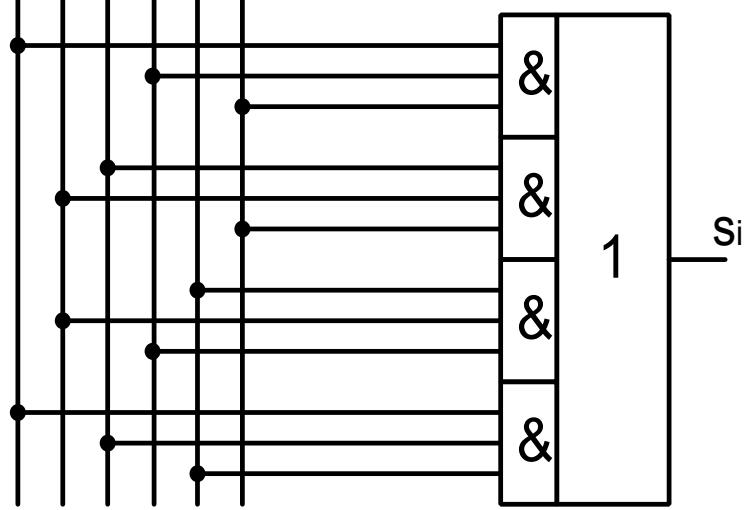
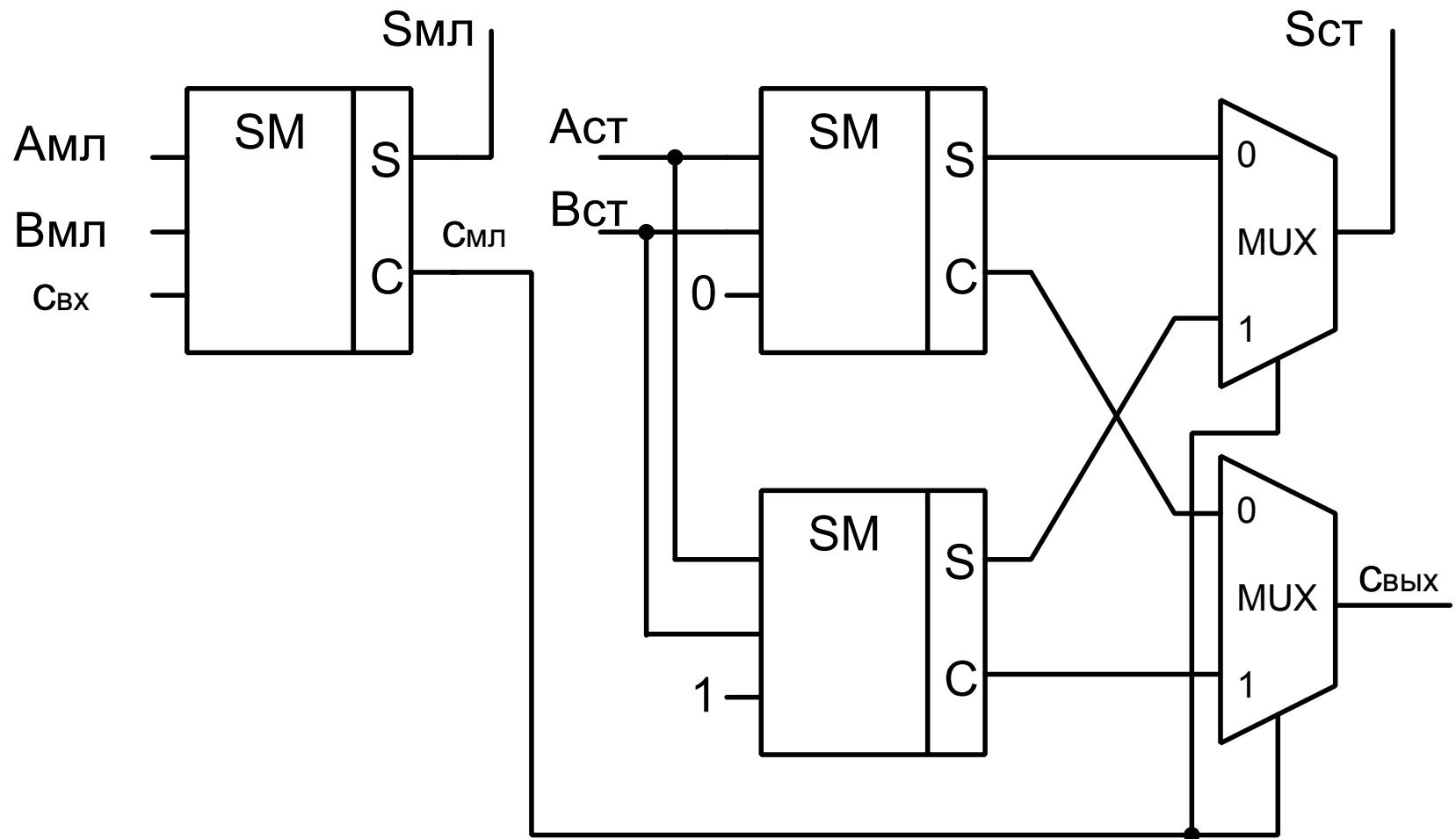
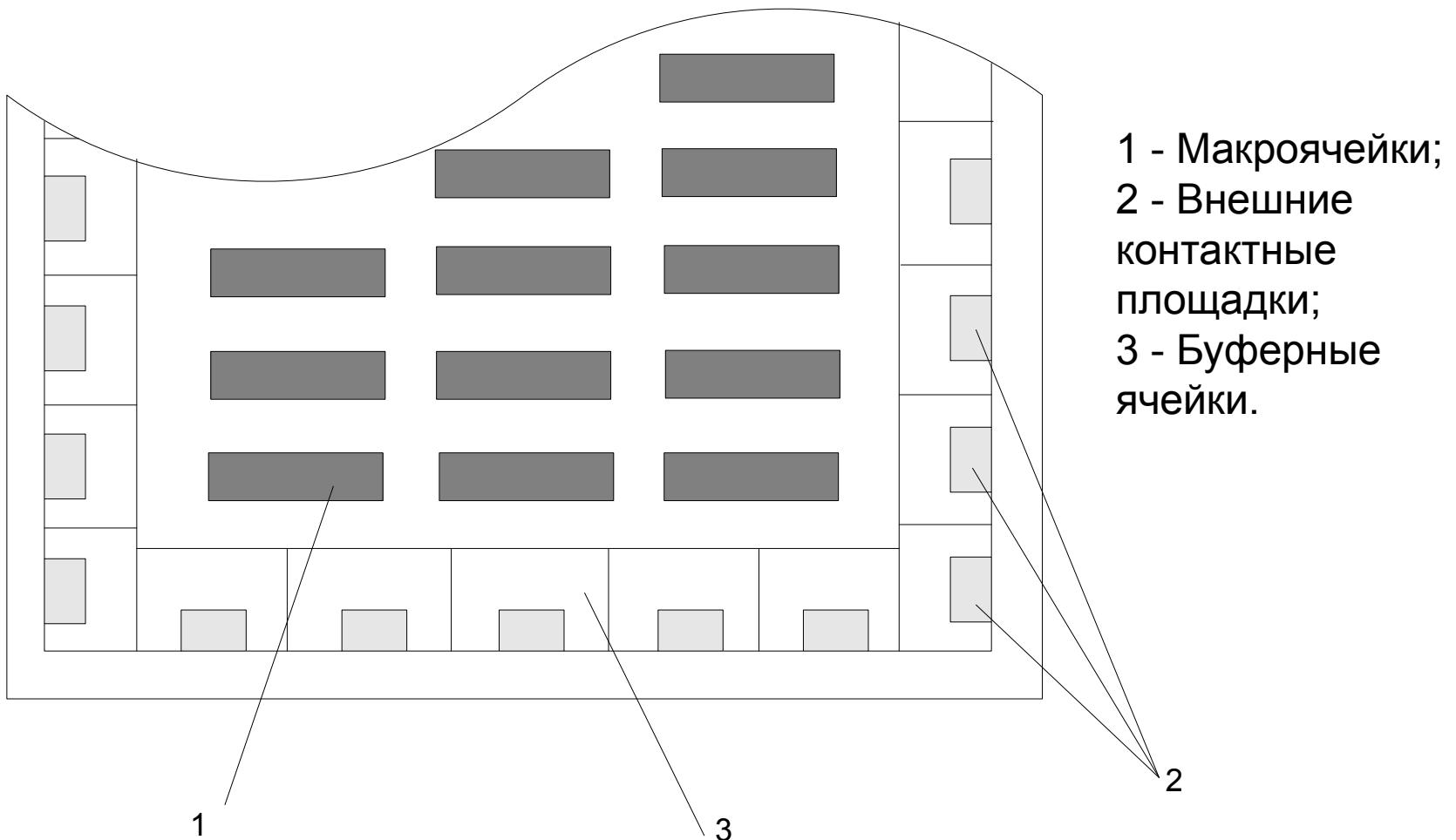


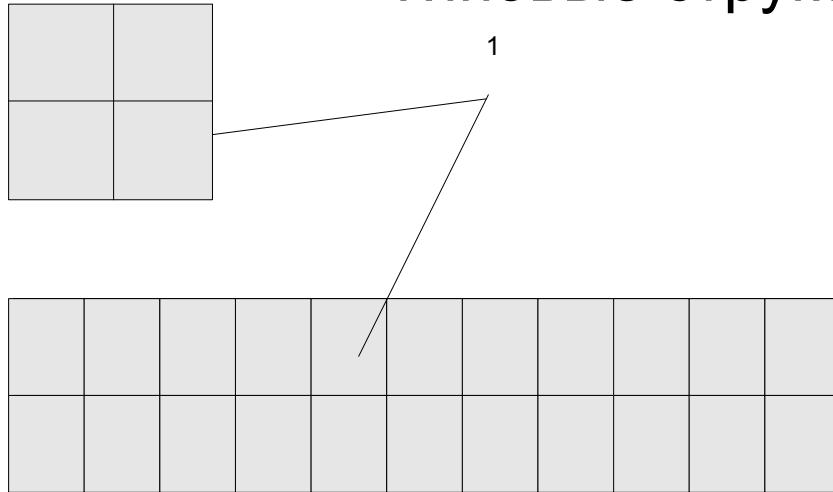
Схема сумматора с условным переносом



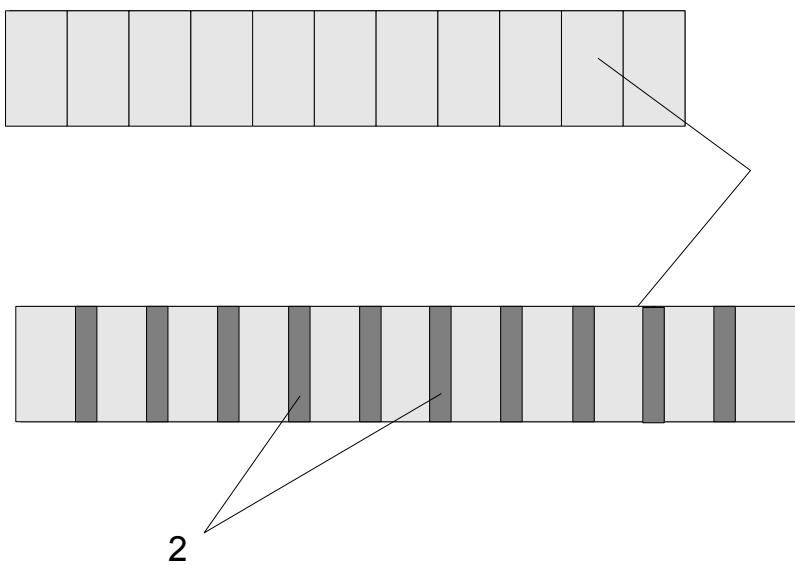
Структура базовых матричных кристаллов



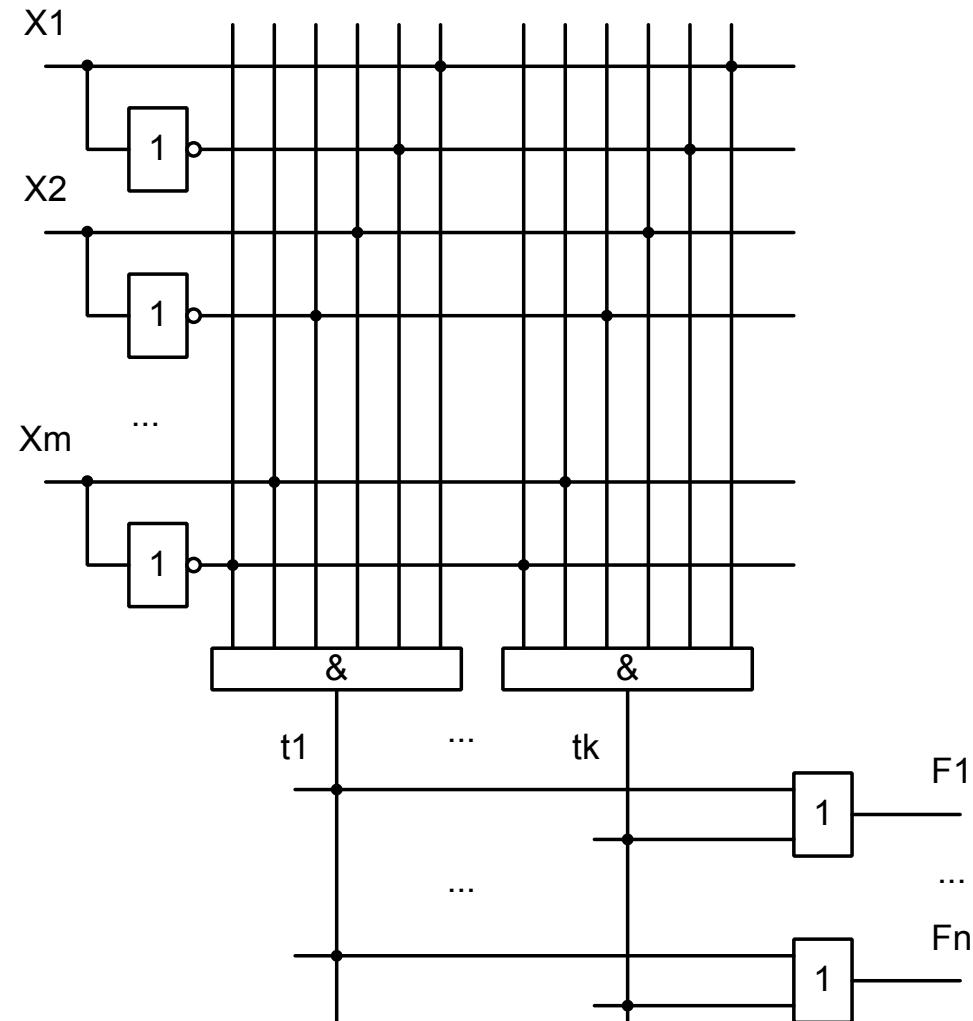
Типовые структуры макроячеек



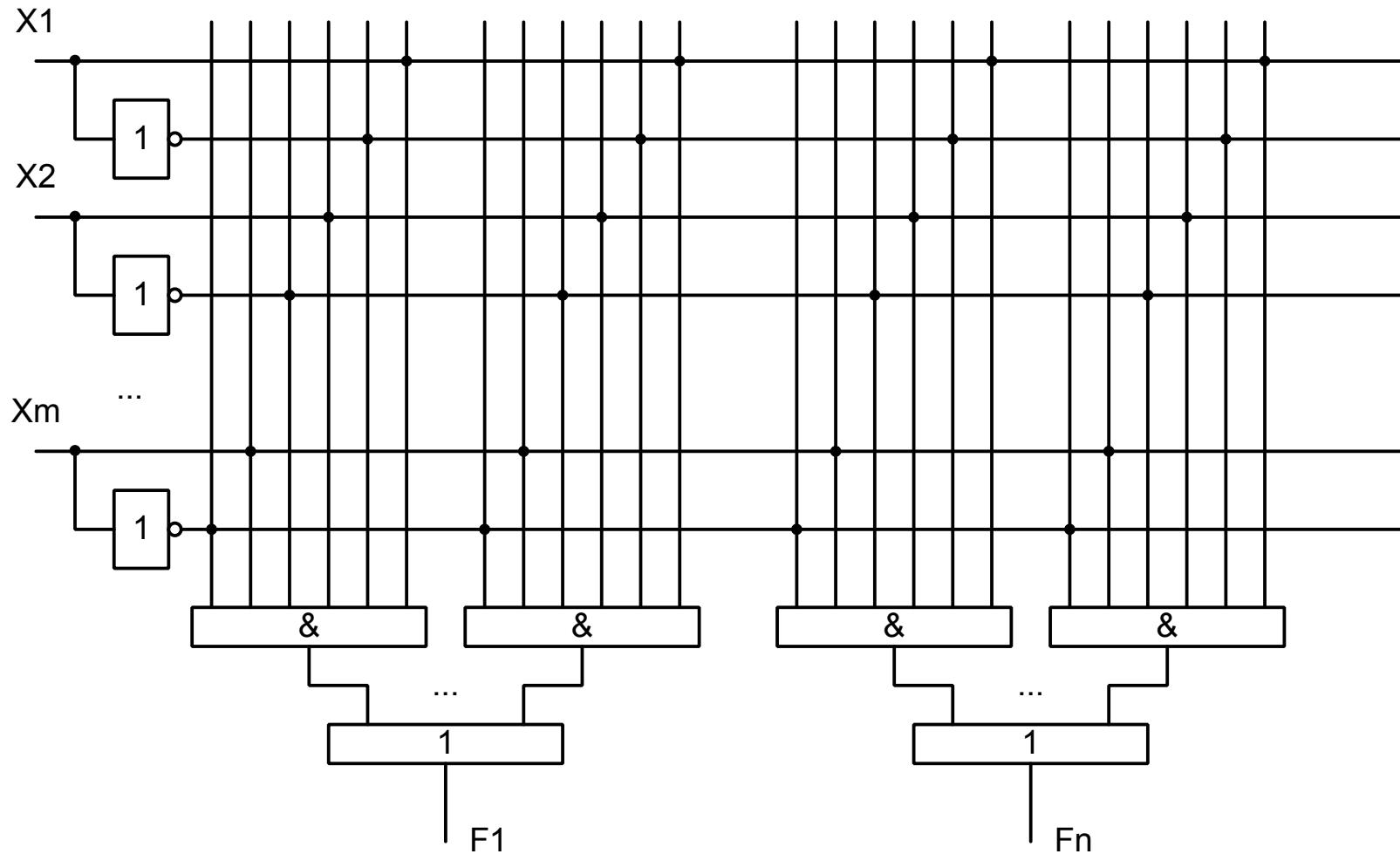
1 - Базовые ячейки (БЯ);
2 - Промежутки между БЯ для прокладки трасс (транзитные соединения).



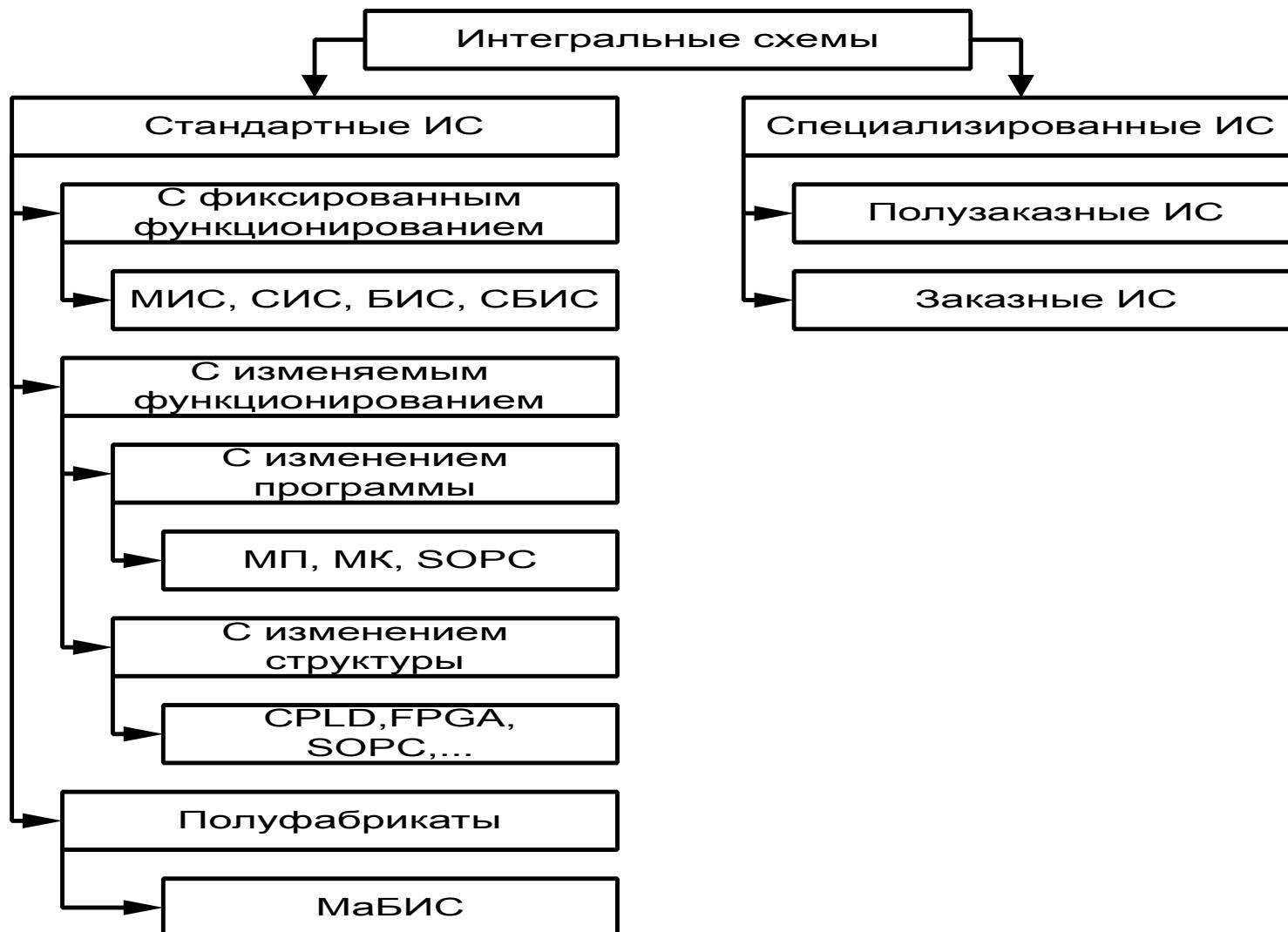
Программируемые логические матрицы



Программируемая матричная логика



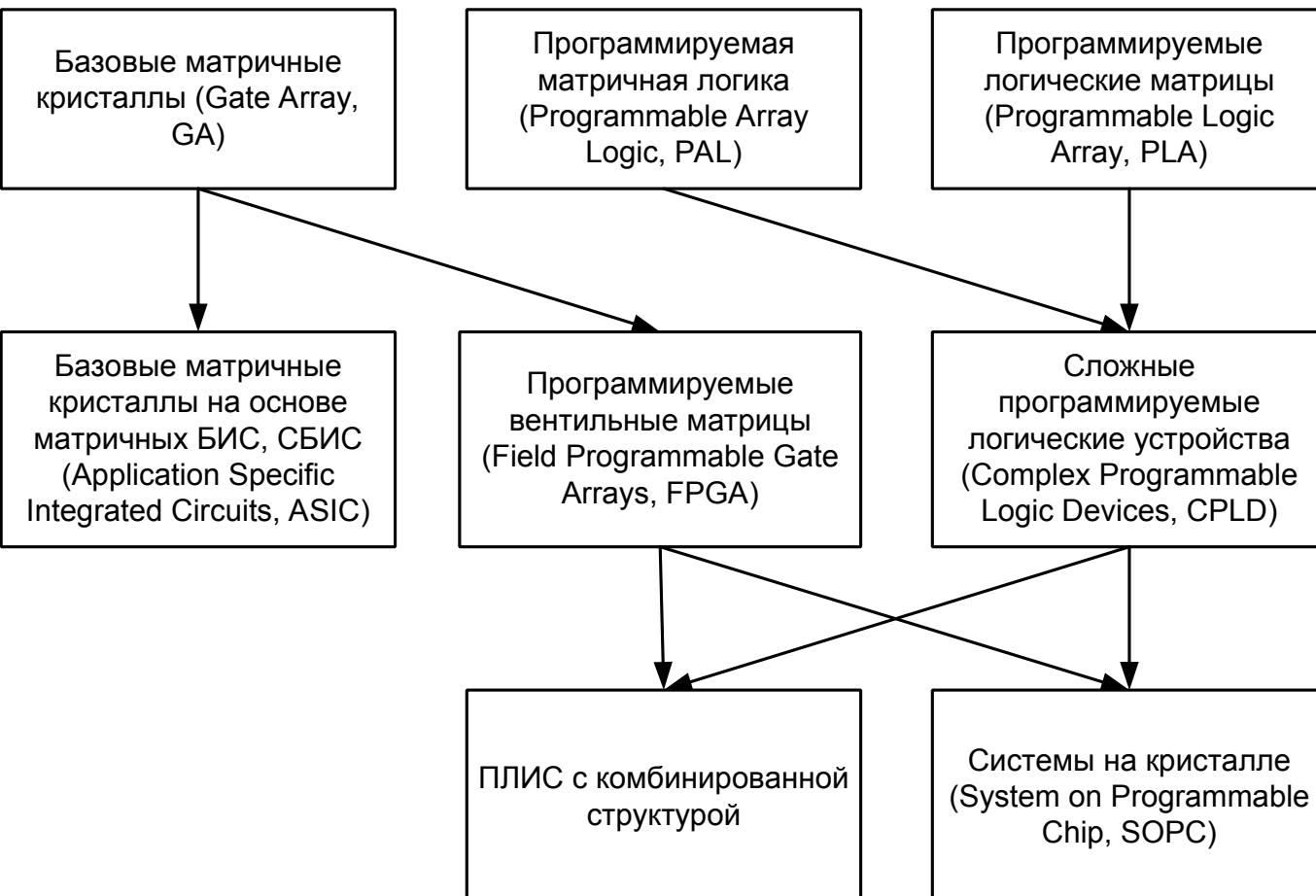
Классификация ИС по способу обеспечения функциональности



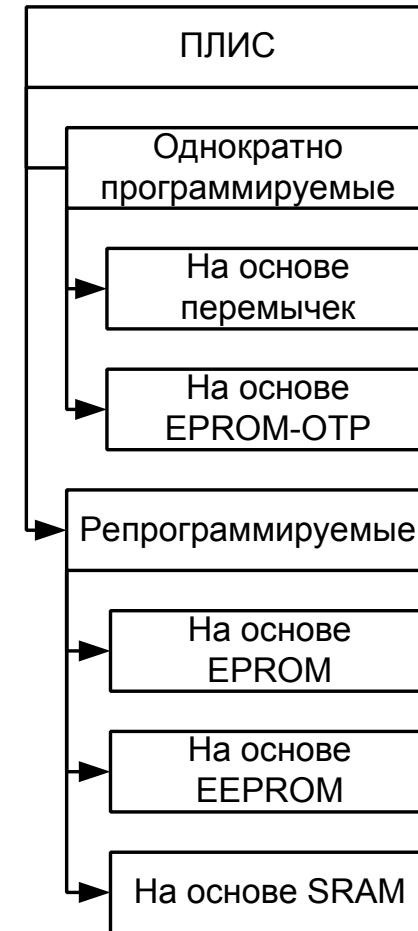
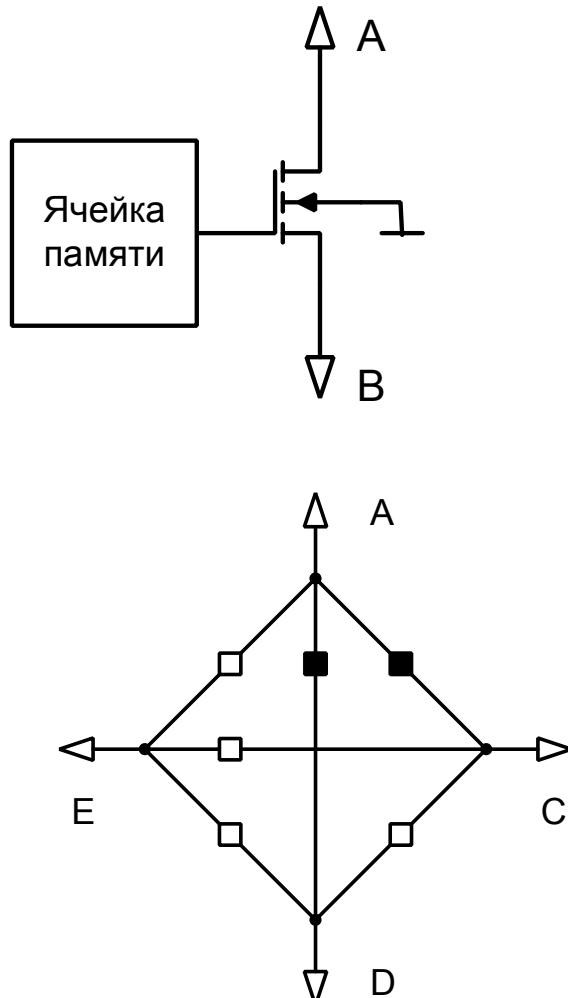
ЭВОЛЮЦИЯ ПЛИС

Полузаказные
интегральные схемы

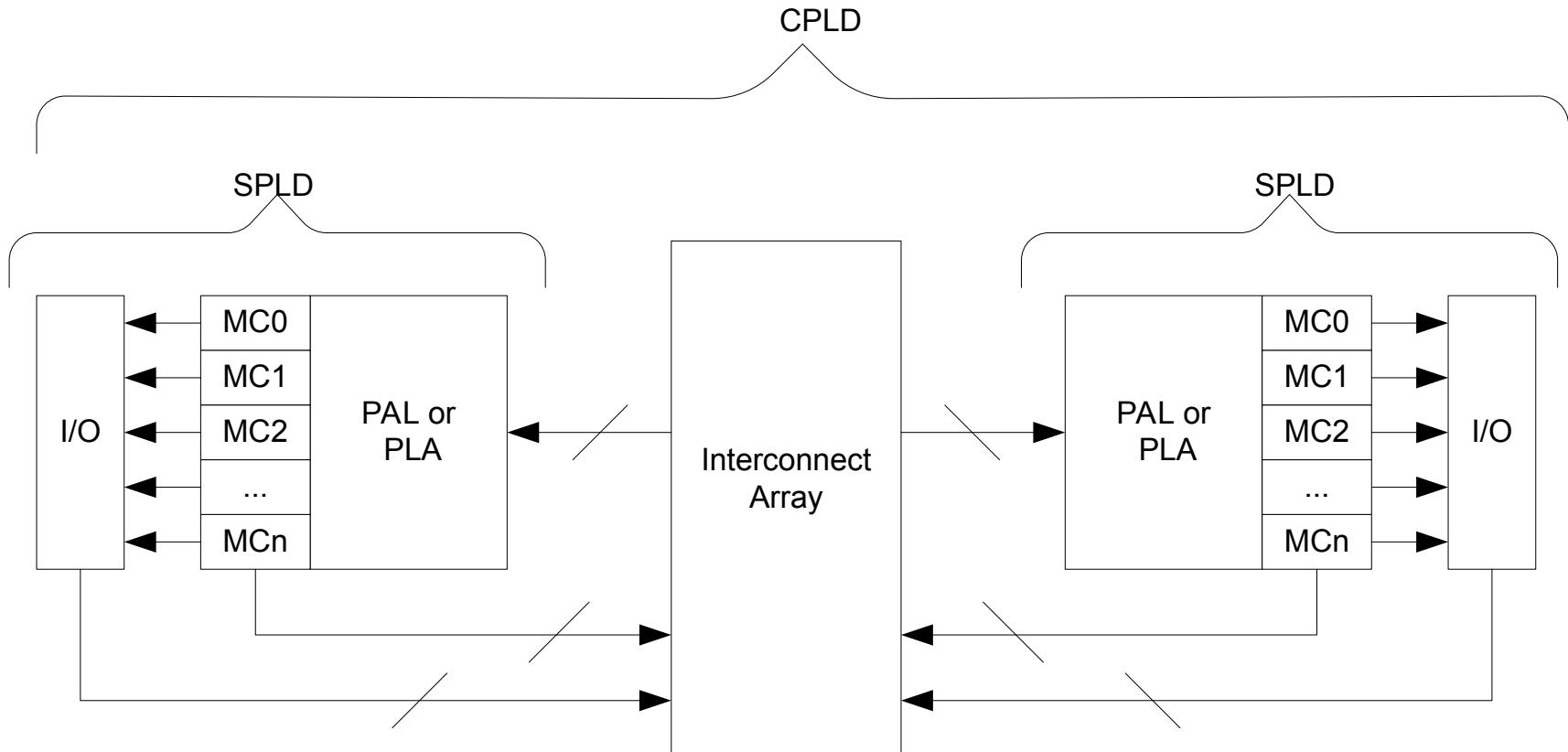
Программируемые пользователем
интегральные схемы



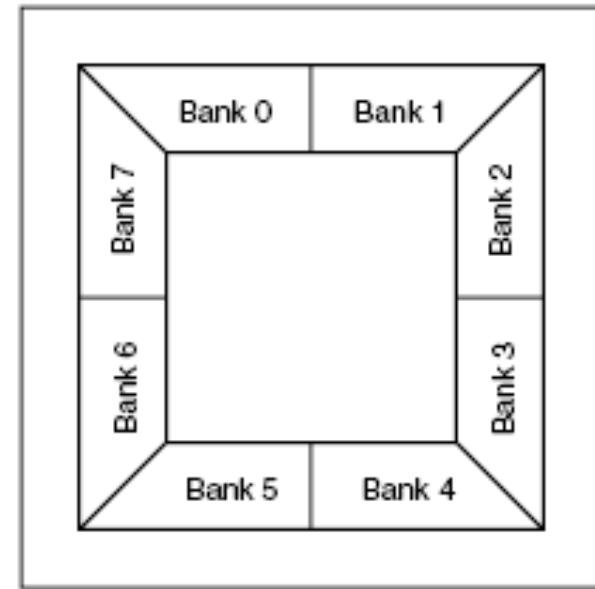
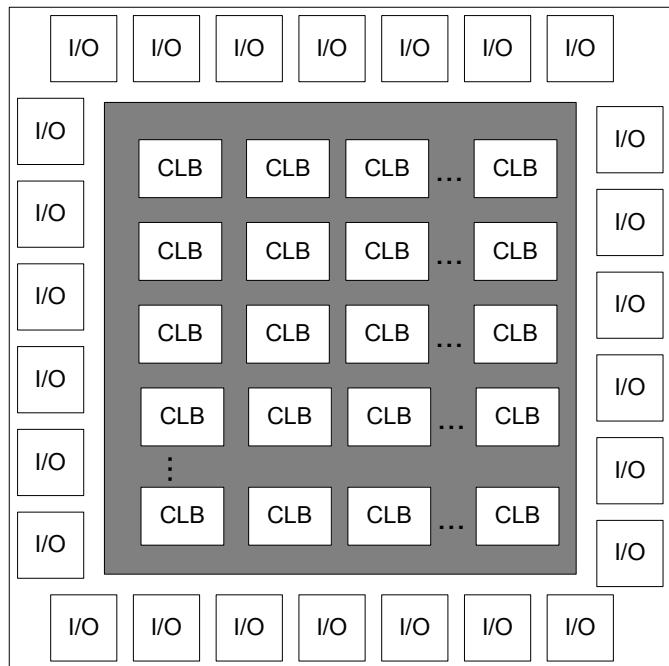
Классификация ПЛИС по типу программируемых связей



Архитектура сложных программируемых логических устройств (CPLD)



Программируемые вентильные матрицы (FPGA)



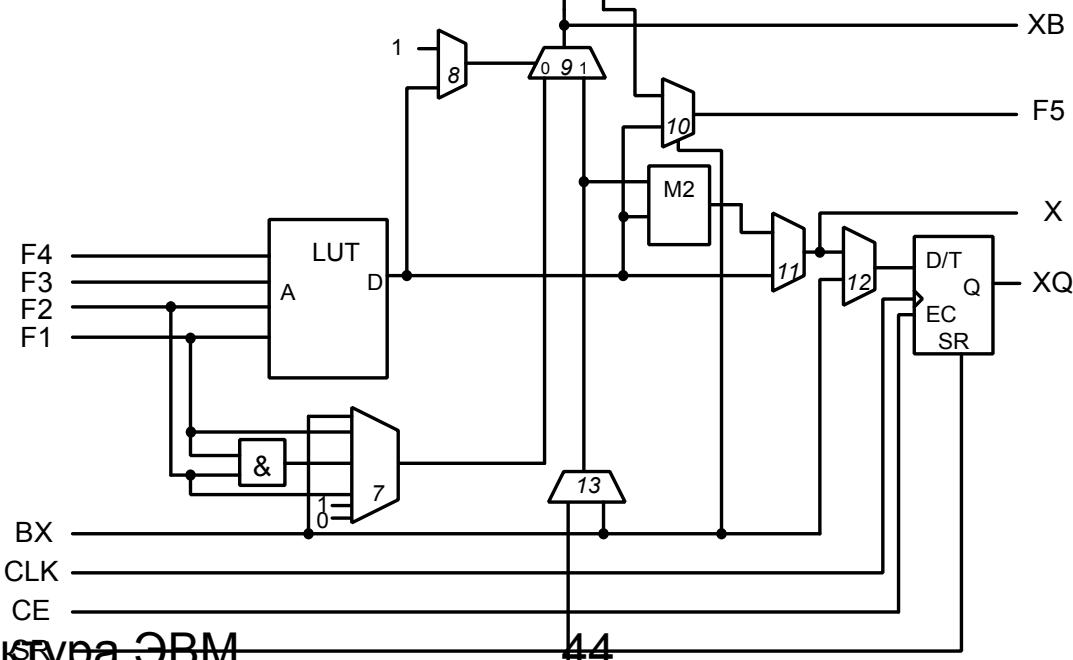
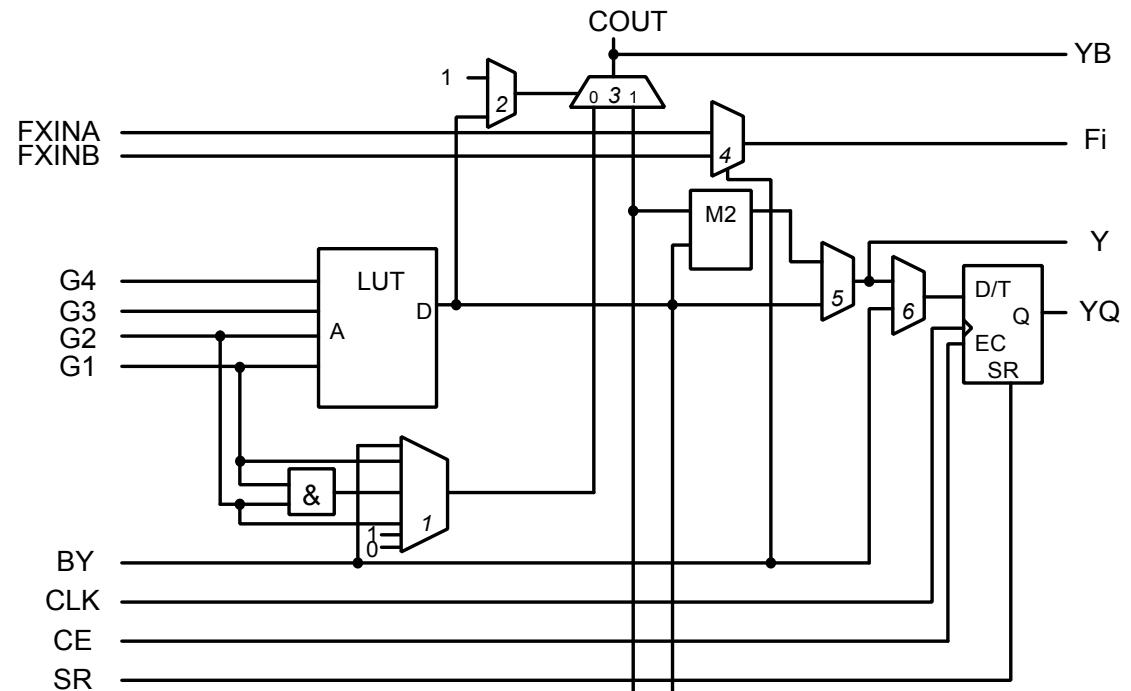
D8090-2_03_082104

Figure 4: Spartan-3 I/O Banks (top view)

Структура блока типа SLICEL

$D = A_i \text{ xor } B_i$,
 $M7 = A_i \text{ and } B_i$
 $S = D \text{ xor } C_{IN}$

F1	F2	D	M7	CIN	S	COUT
0	0	0	0	0	0	0
0	1	1	0	0	1	0(CIN)
1	0	1	0	0	1	0(CIN)



V.II Основы языка VHDL

(Very high speed integration circuits Hardware Description Language)

Стандарт VHDL-87, Стандарт VHDL-93, Стандарт VHDL-AMS

Язык VHDL используется для:

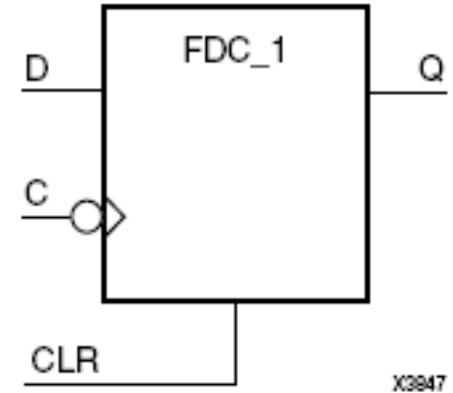
- описания поведения цифровых устройств во времени и при изменении входных воздействий;
- описания структуры цифровых устройств с различной степенью детализации (на системном и блочном уровнях, на уровне регистровых передач, на уровне вентилей);
 - моделирования цифровых устройств;
 - описания тестовых воздействий при моделировании устройств;
 - автоматизации преобразования исходного описания схемы в описание на более низком уровне (вплоть до вентильного уровня).

Стили описания:

- **поведенческий стиль**, при котором для описания проекта используются причинно-следственные связи между событиями на входах устройства и событиями на его выходах (без уточнения структуры);
- **структурный стиль описания**, при котором устройство представляется в виде иерархии взаимосвязанных простых устройств (подобно стилю, принятому в схемотехнике);
- **потоковый стиль описания** устройства основан на использовании логических уравнений, каждое из которых преобразует один или несколько входных информационных потоков в выходные потоки.

Примеры описания устройств на языках VHDL и Verilog

Динамический D-триггер (flip-flop with negative edge clock).
VHDL описание



```
library ieee;
use ieee.std_logic_1164.all;

entity registers_2 is
port(C, D, CLR : in std_logic;
      Q : out std_logic);
end registers_2;

architecture archi of registers_2 is
begin
```

```
process (C, CLR)
begin
  if (CLR = '1')then
    Q <= '0';
  elsif (C'event and C='0')then
    Q <= D;
  end if;
end process;
end archi;
```

V.III Основы языка Verilog HDL

Verilog был разработан фирмой Gateway Design Automation в 1984 г

Стандарт Verilog LRM (Language Reference Manual), IEEE1364-1995
принят в 1995 году

Verilog и VHDL

- VHDL обладает большей универсальностью и может быть использован не только для описания моделей цифровых электронных схем, но и для других моделей.

- Из-за своих расширенных возможностей VHDL проигрывает в эффективности и простоте, то есть на описание одной и той же конструкции в Verilog потребуется в 3–4 раза меньше символов (ASCII), чем в VHDL.

- Как и VHDL, Verilog изначально предназначался для моделирования цифровых систем и как средство описания синтезируемых проектов стал использоваться с 1987 г. В настоящее время ведущие пакеты синтеза систем на ПЛИС, такие как продукты фирм Synopsis, Caddence, Mentor Graphics, многих производителей ПЛИС, поддерживают синтез с описания на языке Verilog.

- Создавать свои типы данных в Verilog нельзя. Основной тип данных для синтезируемых описаний: целое — integer (32-битовое со знаком).

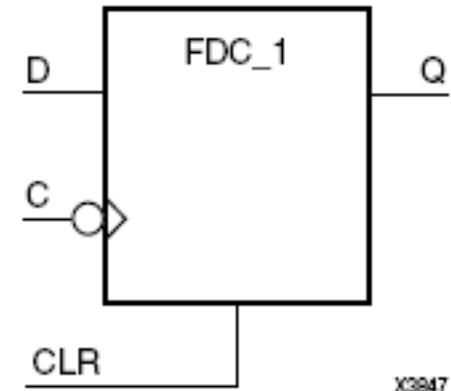
- В Verilog могут быть использованы специфические объекты (UDP, Specify-блоки), не имеющие аналогов в VHDL, а также многочисленные функции PLI (Program Language Interface).

Динамический D-триггер (flip-flop with positive edge clock). Verilog описание

```
module v_registers_2 (C, D, CLR, Q);

input C, D, CLR;
output Q;
reg Q;

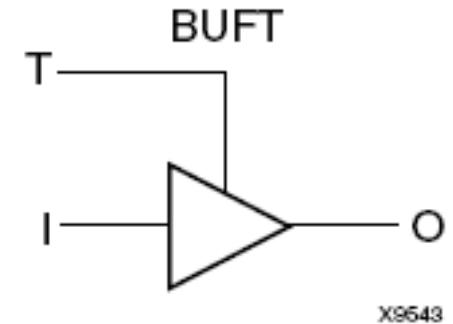
always @ (negedge C or posedge CLR)
begin
    if (CLR)
        Q <= 1'b0;
    else
        Q <= D;
end
endmodule
```



Буфер с третьим состоянием (buft). VHDL описание

```
entity three_st_1 is
port(T : in std_logic;
I : in std_logic;
O : out std_logic);
end three_st_1;
```

```
architecture archi of three_st_1 is
begin
process (I, T)
begin
    if (T='0') then
        O <= I;
    else
        O <= 'Z';
    end if;
end process;
-- Variant 2
-- O <= I when (T='0') else 'Z';
end archi;
```



Буфер с третьим состоянием (buft). Verilog описание

```
module v_three_st_1 (T, I, O);
```

```
    input T, I;  
    output O;  
    reg O;
```

```
    always @(T or I)  
    begin
```

```
        if (~T)
```

```
            O = I;
```

```
        else
```

```
            O = 1'bZ;
```

```
    end
```

```
//Variant 2
```

```
assign O = (~T) ? I: 1'bZ;
```

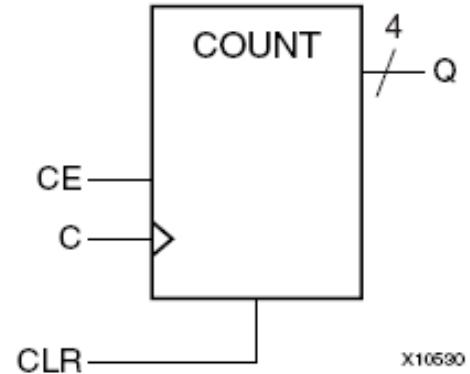
```
endmodule
```

Счетчик с разрешением счета (Counter). VHDL описание

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
```

```
entity counters_5 is
port(C, CLR, CE : in std_logic;
Q : out std_logic_vector(3 downto 0));
end counters_5;
```

```
architecture archi of counters_5 is
signal tmp: std_logic_vector(3 downto 0);
begin
process (C, CLR)
begin
if (CLR='1') then
tmp <= "0000";
elsif (C'event and C='1') then
if (CE='1') then
tmp <= tmp + 1;
end if;
end if;
end process;
Q <= tmp;
end archi;
```



Счетчик с разрешением счета (Counter). Verilog описание

```
module v_counters_5 (C, CLR, CE, Q);

input C, CLR, CE;
output [3:0] Q;
reg [3:0] tmp;

always @(posedge C or posedge CLR)
begin
    if (CLR)
        tmp <= 4'b0000;
    else if (CE)
        tmp <= tmp + 1'b1;
    end
    assign Q = tmp;
endmodule
```

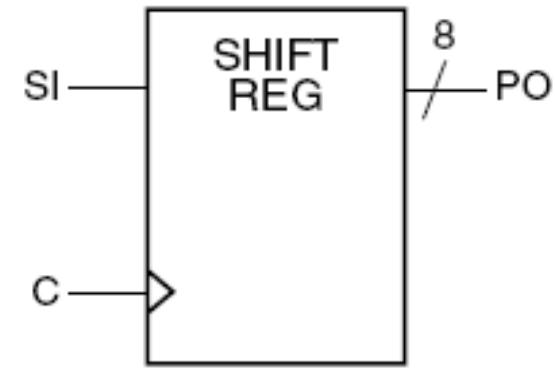
Сдвиговый регистр с последовательной загрузкой (Shift register with serial in and parallel out).

VHDL описание

```
library ieee;
use ieee.std_logic_1164.all;

entity shift_registers_5 is
port(C, SI : in std_logic;
PO : out std_logic_vector(7 downto 0));
end shift_registers_5;

architecture archi of shift_registers_5 is
signal tmp: std_logic_vector(7 downto 0);
begin
process (C)
begin
if (C'event and C='1') then
    tmp <= tmp(6 downto 0)& SI;
end if;
end process;
PO <= tmp;
end archi;
```



Сдвиговый регистр с последовательной загрузкой (Shift register with serial in and parallel out).

Verilog описание

```
module v_shift_registers_5 (C, SI, PO);
input C,SI;
output [7:0] PO;
reg [7:0] tmp;

always @(posedge C)
    tmp <= {tmp[6:0], SI};

assign PO = tmp;

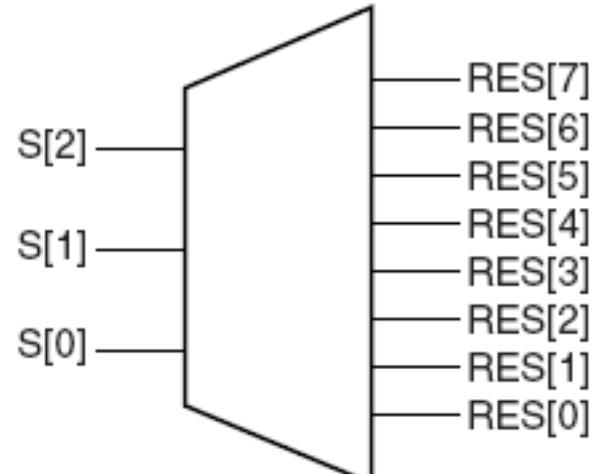
endmodule
```

Дешифратор (Decoder). VHDL описание

```
library ieee;
use ieee.std_logic_1164.all;

entity decoders_1 is
port (sel: in std_logic_vector (2 downto 0);
res: out std_logic_vector (7 downto 0));
end decoders_1;

architecture archi of decoders_1 is
begin
    res <= "00000001" when sel = "000" else
        "00000010" when sel = "001" else
        "00000100" when sel = "010" else
        "00001000" when sel = "011" else
        "00010000" when sel = "100" else
        "00100000" when sel = "101" else
        "01000000" when sel = "110" else
        "10000000";
end archi;
```



X10547

Дешифратор (Decoder). Verilog описание

```
module v_decoders_1 (sel, res);

    input [2:0] sel;
    output [7:0] res;
    reg [7:0] res;

    always @ (sel or res)
    begin
        case (sel)
            3'b000 : res = 8'b00000001;
            3'b001 : res = 8'b00000010;
            3'b010 : res = 8'b00000100;
            3'b011 : res = 8'b00001000;
            3'b100 : res = 8'b00010000;
            3'b101 : res = 8'b00100000;
            3'b110 : res = 8'b01000000;
            default : res = 8'b10000000;
        endcase
    end
endmodule
```

Архитектура ПЛИС типа SOPC

Варианты реализации библиотечных блоков:

Soft - ядра.

Firm - ядра.

Hard – ядра.

Назначение ядер:

Память (ОЗУ, FIFO, кэш- память, ...).

АЛУ (умножители, ...).

Интерфейсная логика (JTAG, PCI, SPI, UART, ...).

МП и МК.

VIII. Организация памяти ЭВМ

Памятью ЭВМ называется совокупность устройств, служащих для запоминания, хранения и выдачи информации.

Характеристики памяти ЭВМ:

- Назначение.
- Информационная емкость.
- Информационная емкость читаемого слова.
- Способ доступа.
- Быстродействие.
- Физический способ хранения информации.

Классификация запоминающих устройств по способу доступа.

- Адресные ЗУ

Постоянные ЗУ, ПЗУ (ROM)

ЗУ с произвольным доступом (RAM)

- Ассоциативные ЗУ

Полностью ассоциативные ЗУ

Ассоциативные ЗУ с прямым размещением

Наборно-ассоциативные ЗУ

- Последовательные ЗУ

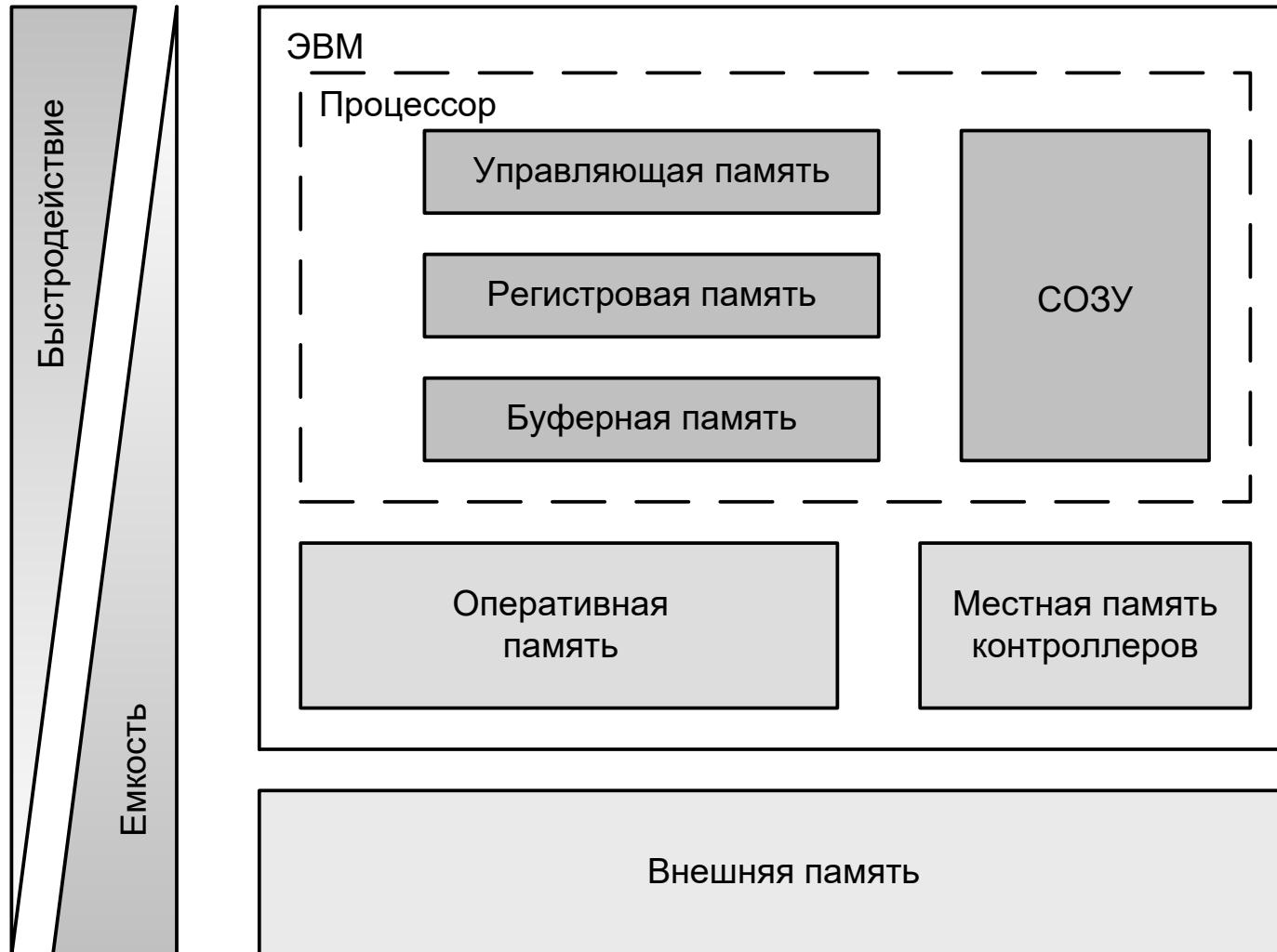
FIFO

LIFO

Файловые

Циклические

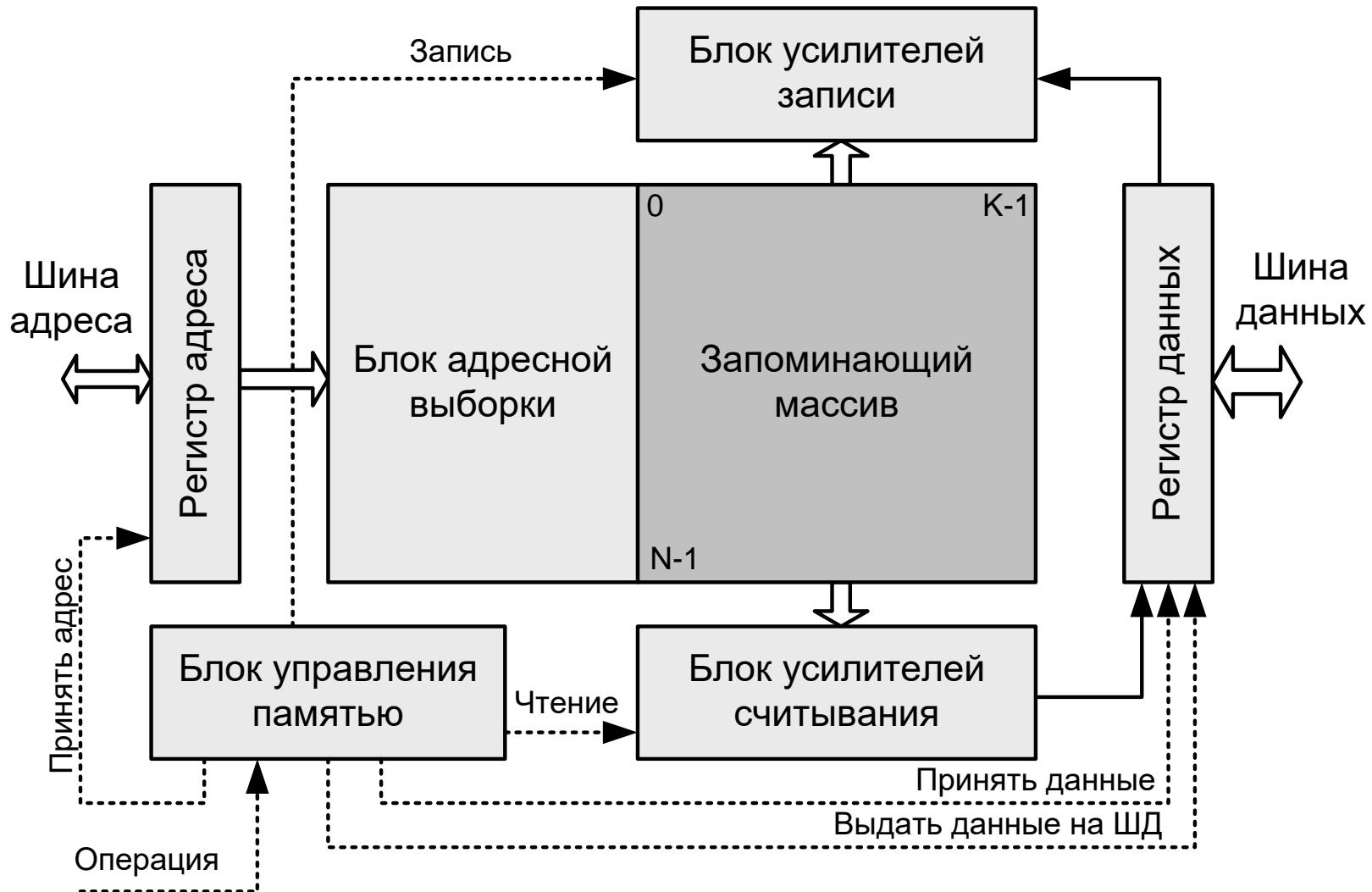
Классификация запоминающих устройств по назначению.



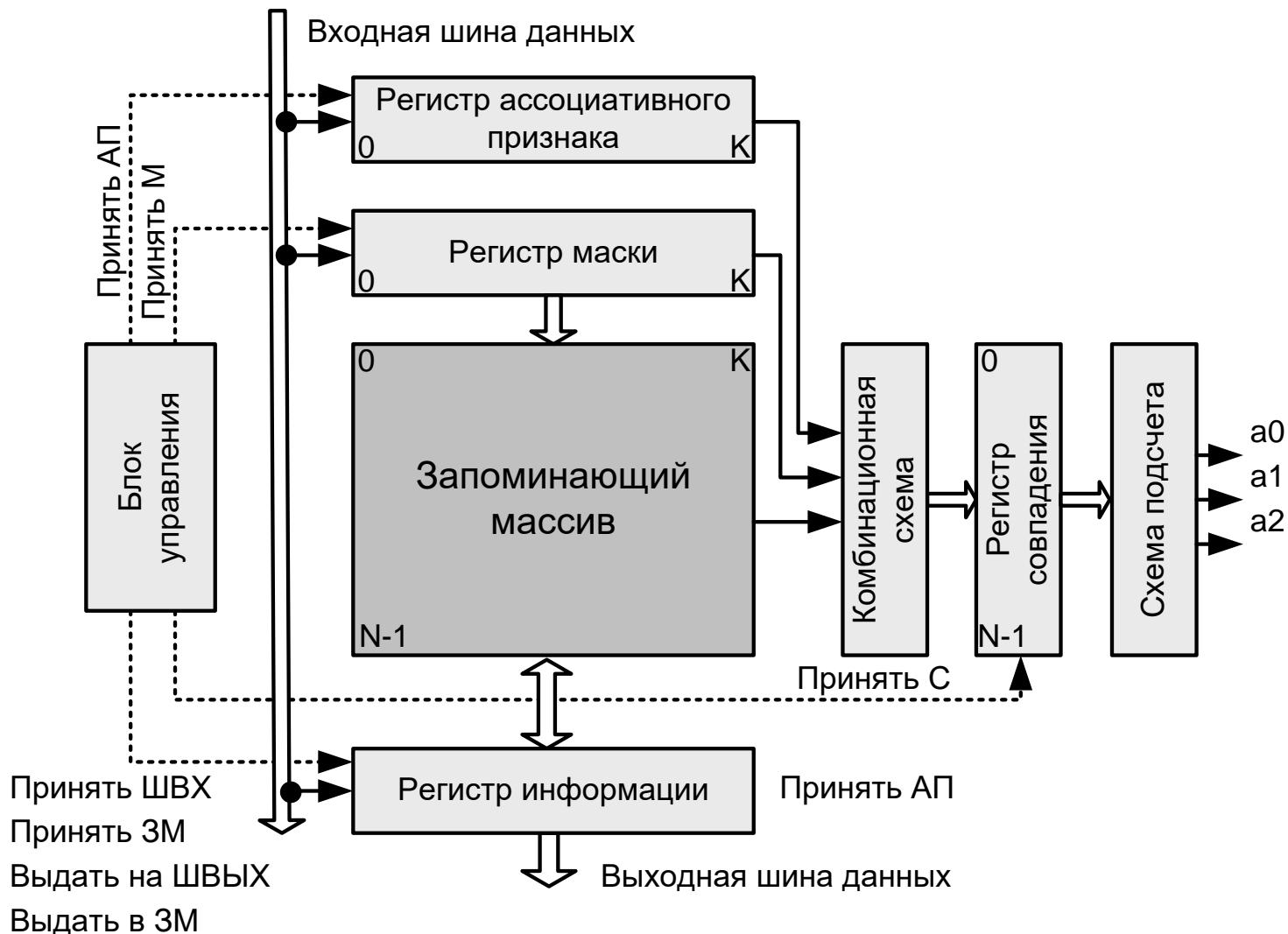
Латентность при обращении к подсистеме памяти

Тип обращения к памяти	Объем памяти, байт	Латентность, такты процессора
Регистры процессора	$2^6..2^{10}$	1
Кэш первого уровня	$2^{14}..2^{16}$	2..4
Кэш второго уровня	$2^{15}..2^{22}$	10..12
Кэш третьего уровня	$2^{21}..2^{26}$	15..50
ОЗУ на одном кристалле с процессором при попадании в TLB (доступ по случайным адресам)	$2^{14}..2^{24}$	10..75
Внешнее ОЗУ при попадании в TLB (доступ по случайным адресам)	$2^{20}..2^{40}$	200..400
Внешнее ОЗУ при промахе в TLB (доступ по случайным адресам)	$2^{20}..2^{40}$	2000..2500
Внешнее ОЗУ при выгруженной во внешний Flash диск странице	$2^{30}..2^{42}$	$1*10^5..1*10^6$
Внешнее ОЗУ при выгруженной во внешний жесткий диск странице	$2^{30}..2^{50}$	$1*10^6..1*10^8$

Обобщенная схема адресного ЗУ

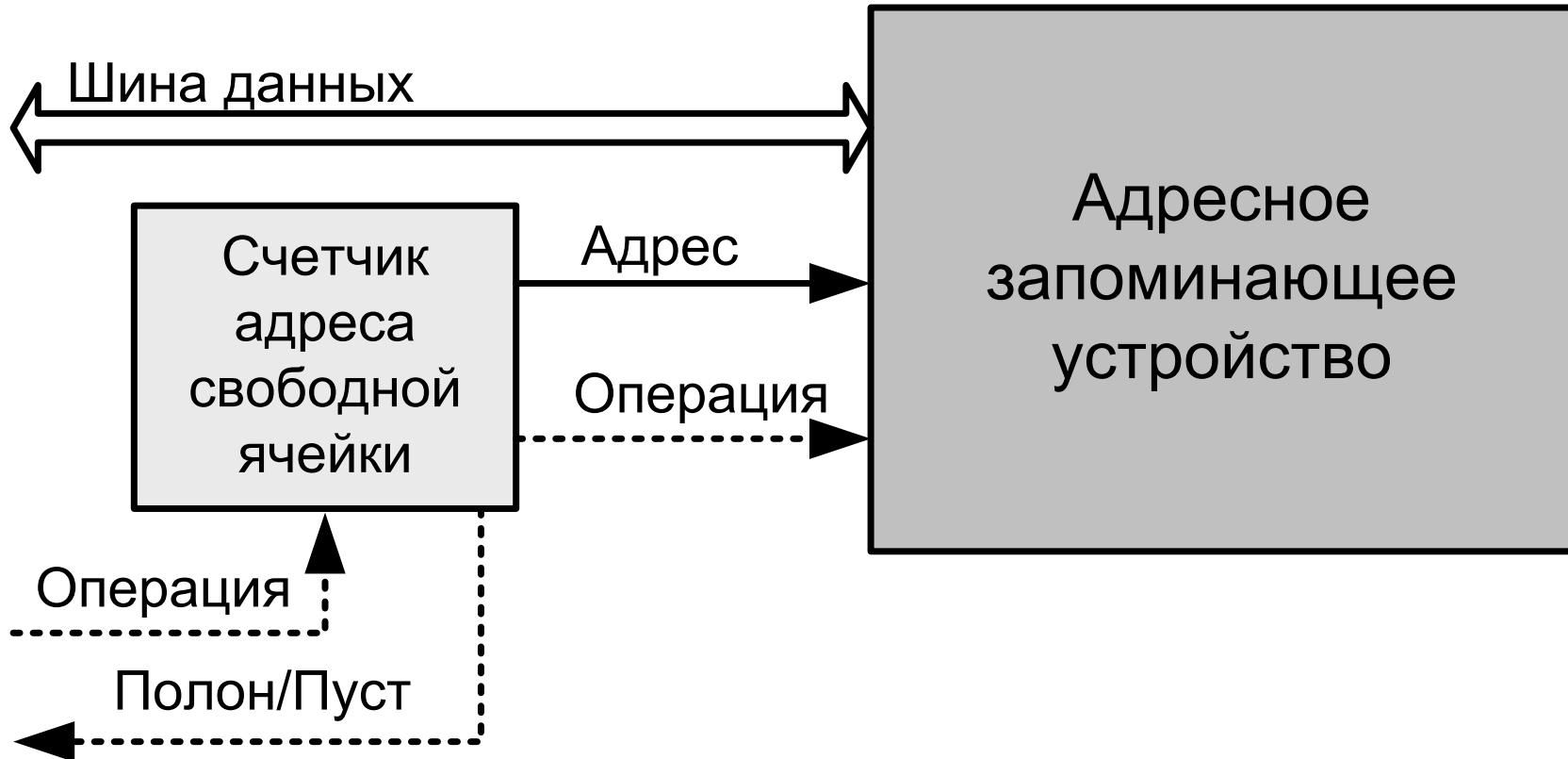


Обобщенная схема ассоциативного ЗУ

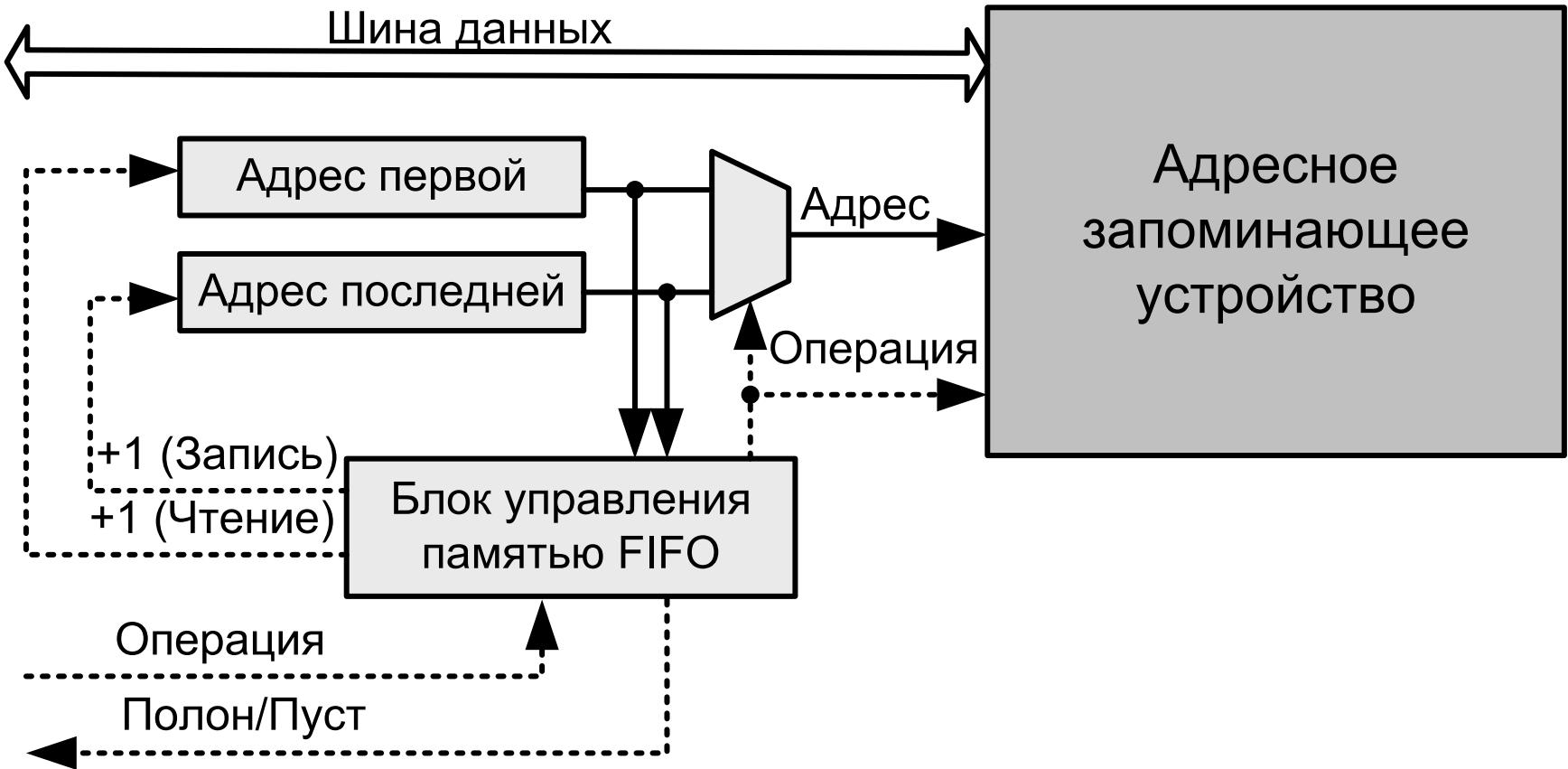


Обобщенная схема последовательного ЗУ

Стек (память типа LIFO)



Буфер (память типа FIFO)



Адресные запоминающие устройства

Постоянные ЗУ, ПЗУ (ROM)

МПЗУ (MROM)

ППЗУ (PROM)

РПЗУ-УФ (EPROM)

ОПРПЗУ-УФ (EPROM-OTP)

РПЗУ-ЭС (EEPROM)

FLASH

ЗУ с произвольным доступом (RAM)

Динамические ЗУПД (DRAM)

Использующие кучность
адресов

FPM DRAM

EDO DRAM

BEDO DRAM

SDRAM

DDR SDRAM

RDRAM

Не использующие кучность
адресов

DRAM

RLDRAM

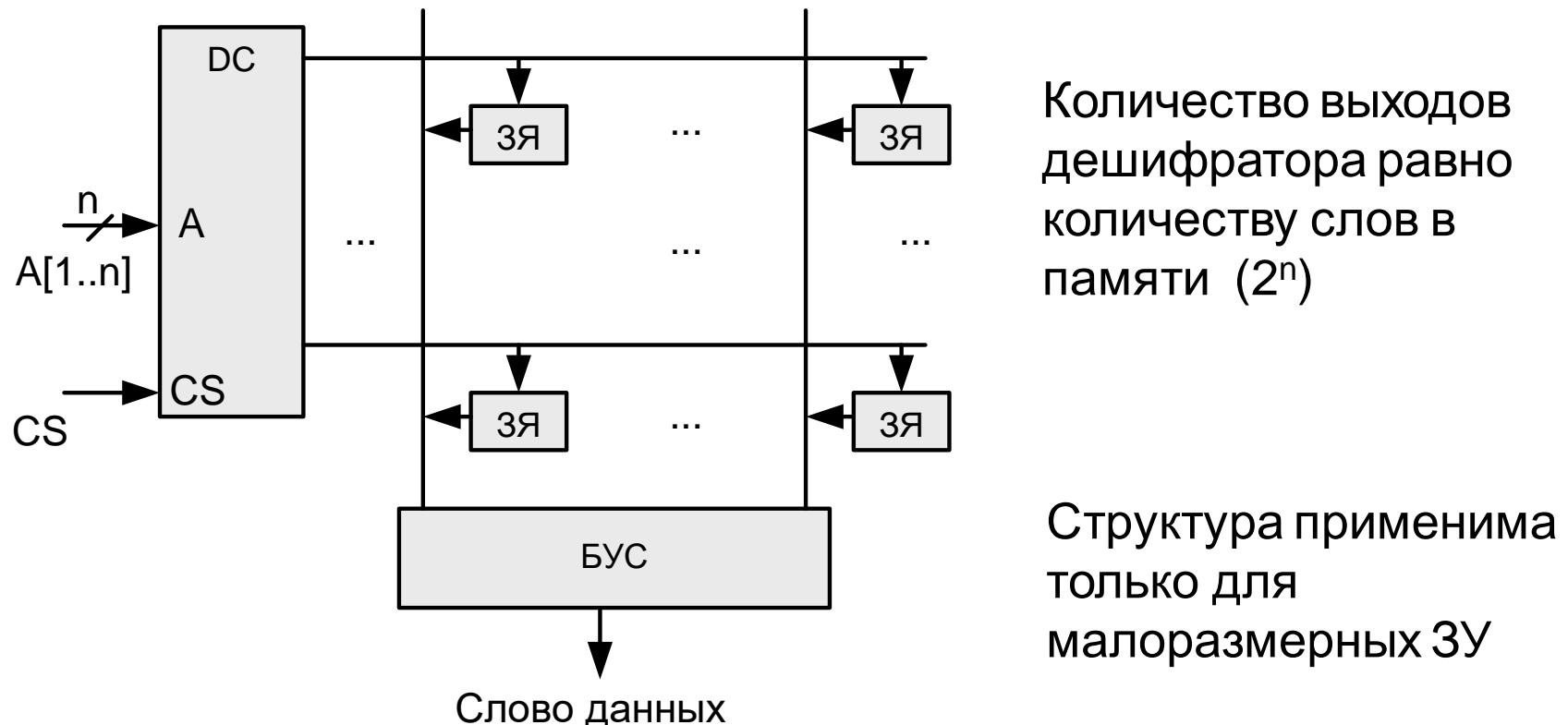
Статические ЗУПД (SRAM)

Асинхронные

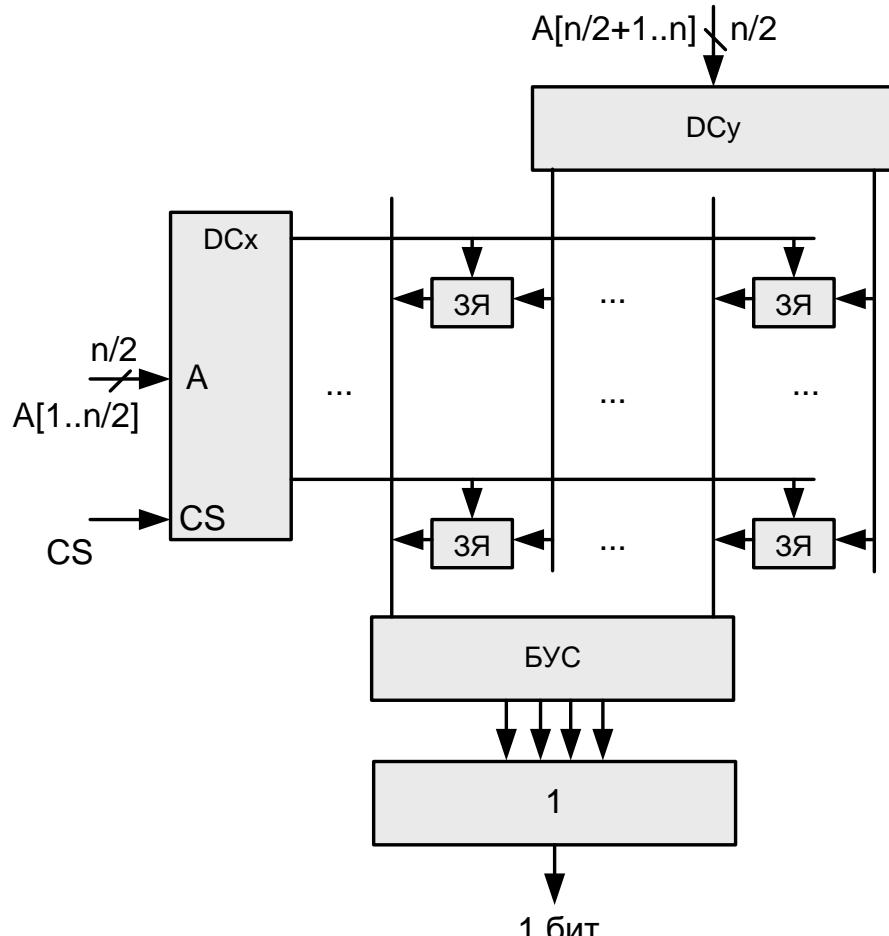
Синхронные

Организация запоминающих массивов адресных ЗУ

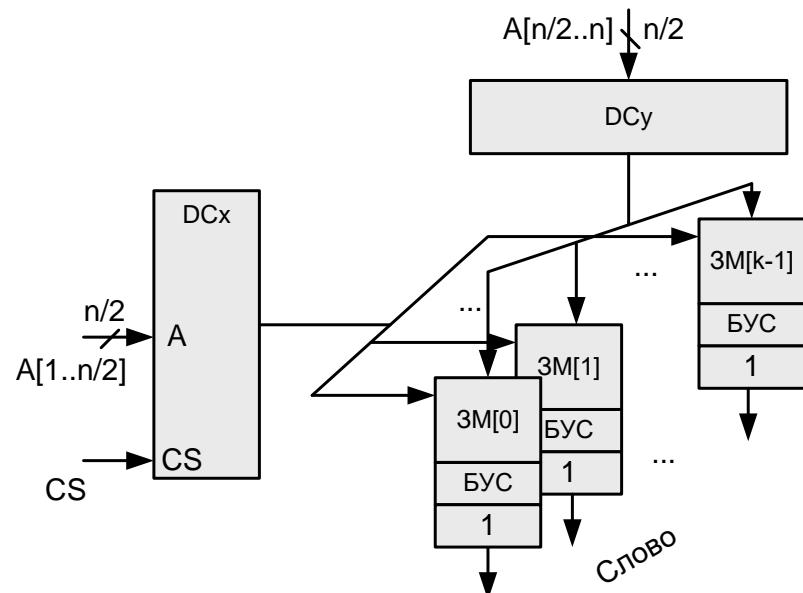
Структура ЗМ типа 2D



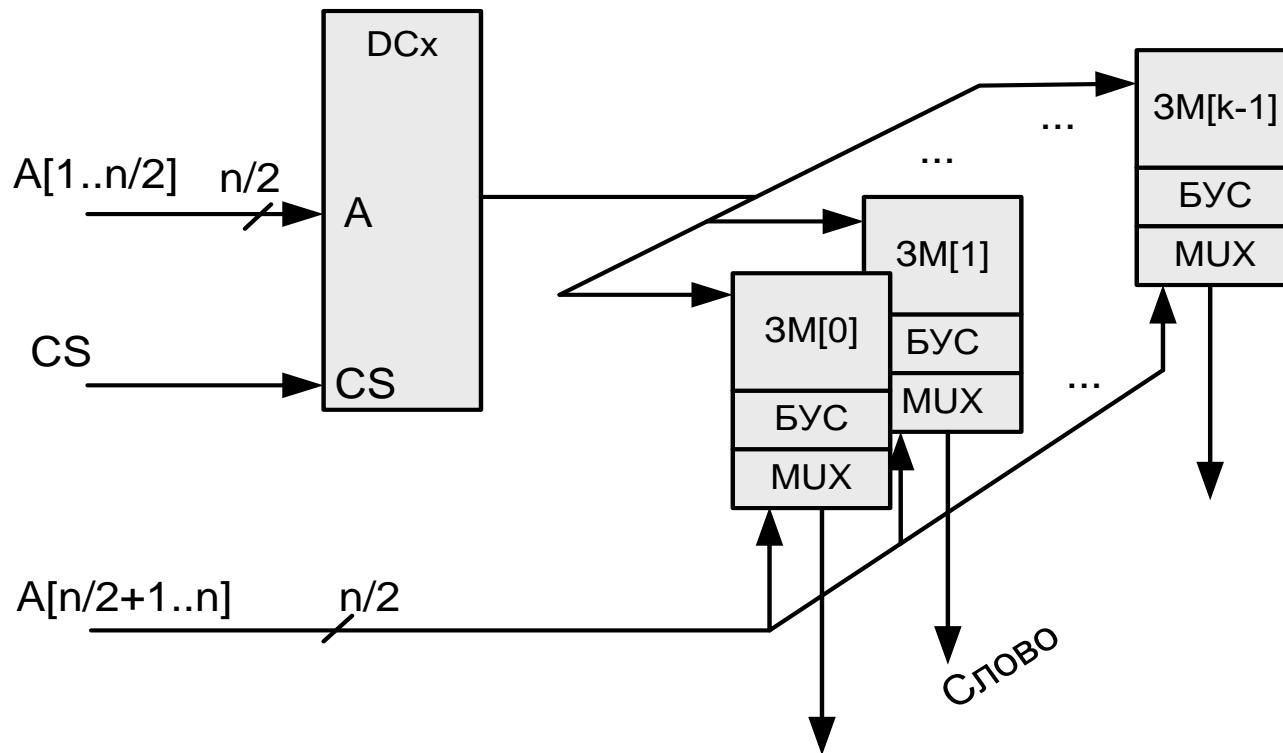
Структура 3М типа 3D



Адрес делится на две части (двухкоординатная выборка).
Количество выходов дешифраторов: $2^{n/2} + 2^{n/2}$



Структура ЗМ типа 2DM

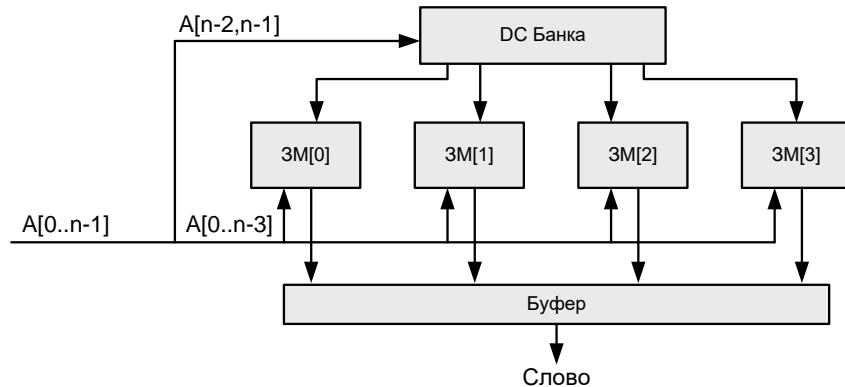


Мультиплексоры позволяют выбрать один из $2^{n/2}$ разрядов каждого из запоминающих массивов

- Размеры массивов близки к оптимальным.
- Количество линий записи/считывания минимально.

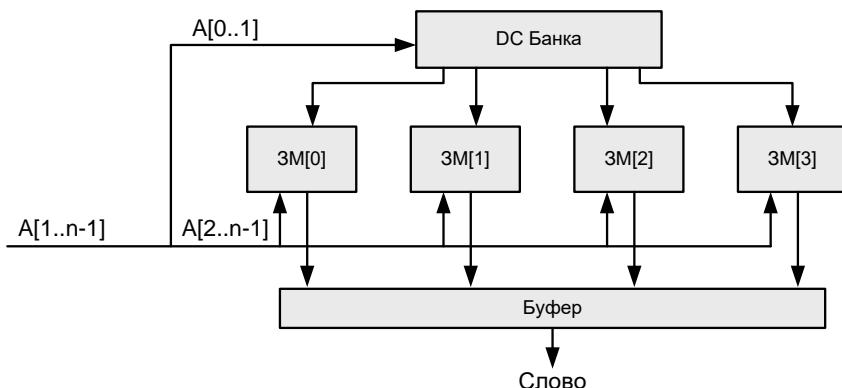
Расслоение памяти

Блочное разделение адреса



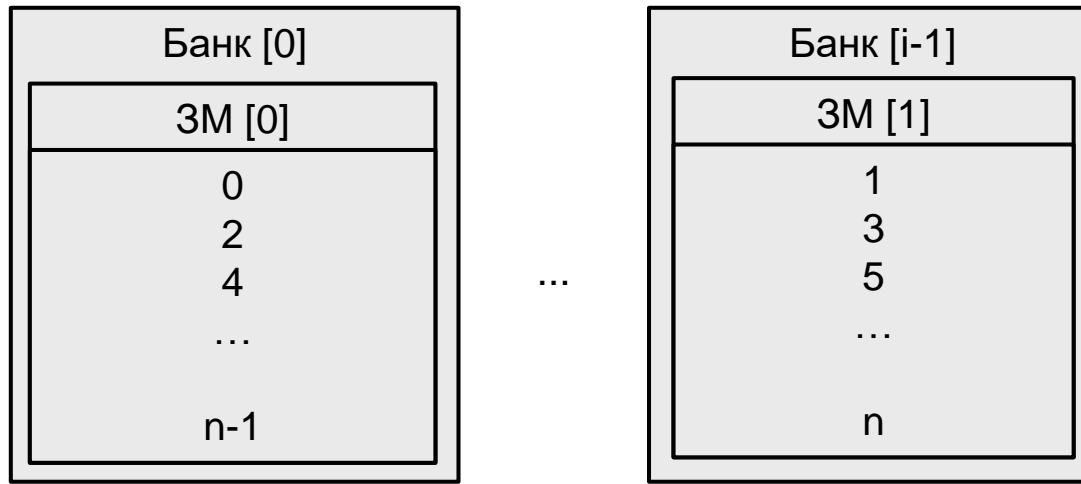
Номер банка
определяется старшей
частью адреса.

Циклическое разделение адреса



Номер банка определяется
младшой частью адреса

Блочно-циклическое разделение адреса



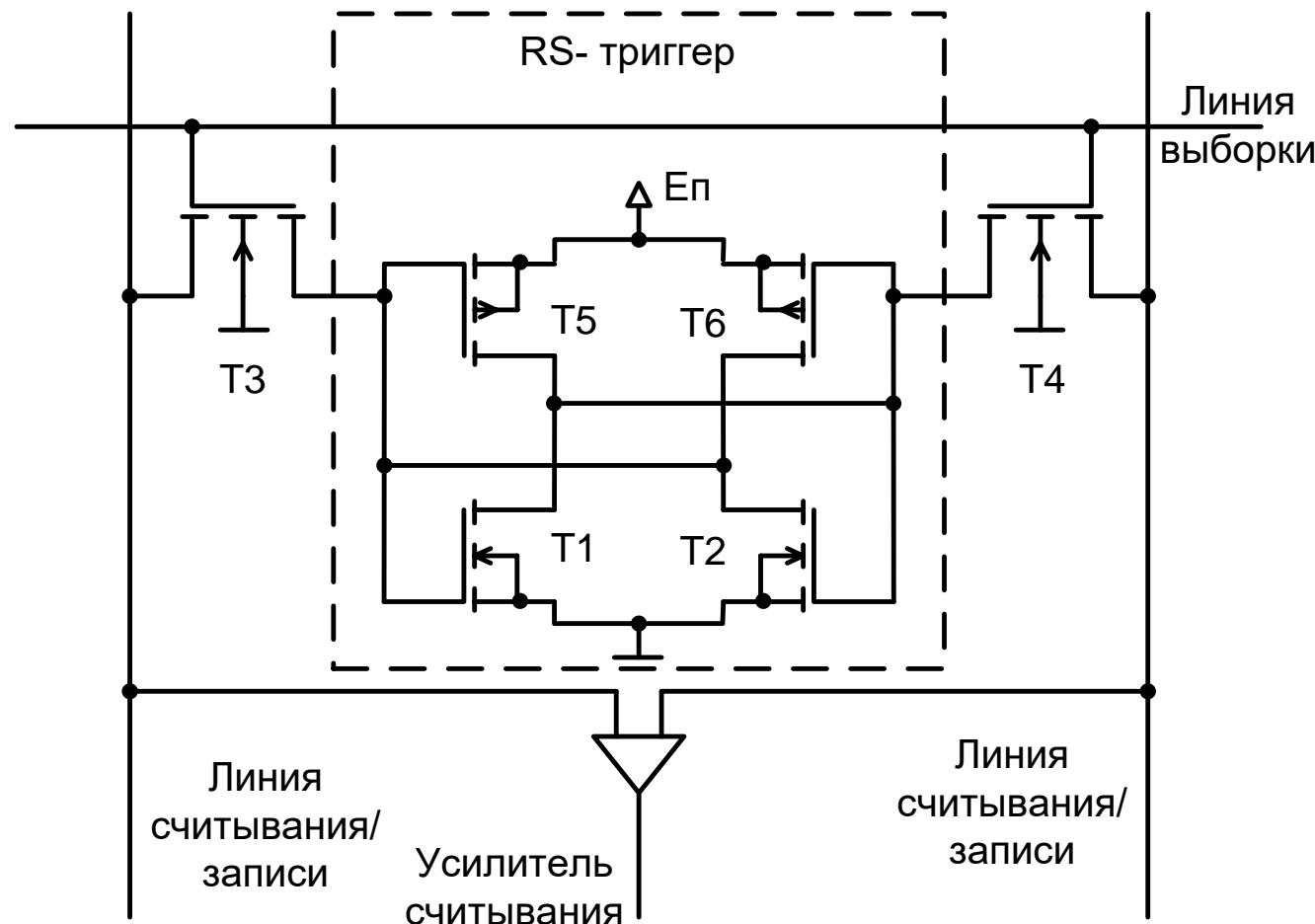
Блочно-циклический способ обеспечивает возможность пакетной передачи и ускоряет доступ при кучности адресов

Пример разделения адреса в SDRAM (РІІІ)

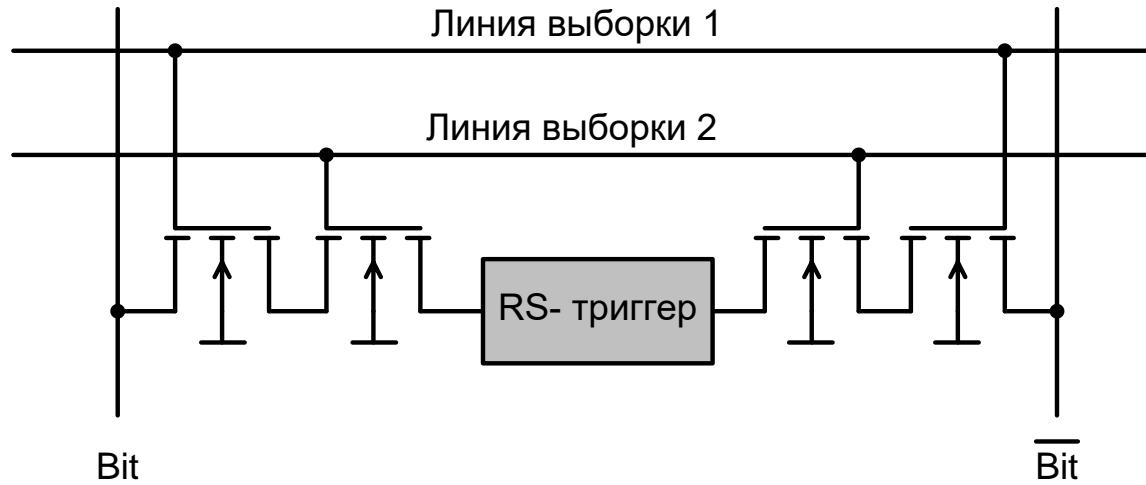
Старшая часть номера столбца	Номер строки				Номер банка	Младшая часть номера столбца	Смещение в пакете (16 байт)
31	25	24	13	12 11	10	4	3 0

Статические ЗУ с произвольной выборкой (SRAM)

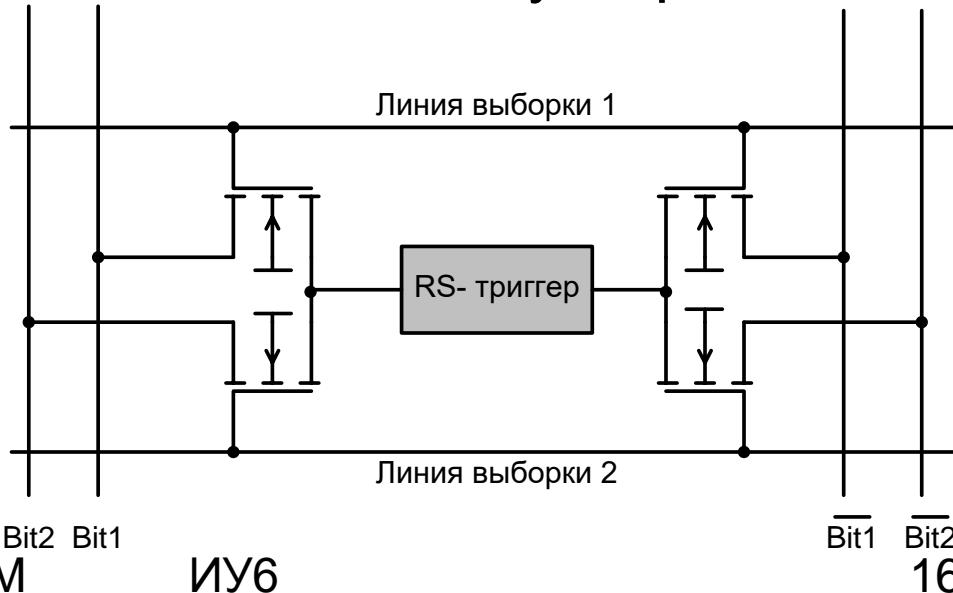
Запоминающая ячейка статической памяти



Запоминающая ячейка с двухкоординатной выборкой



Запоминающая ячейка двухпортовой выборкой



Микросхема статической памяти

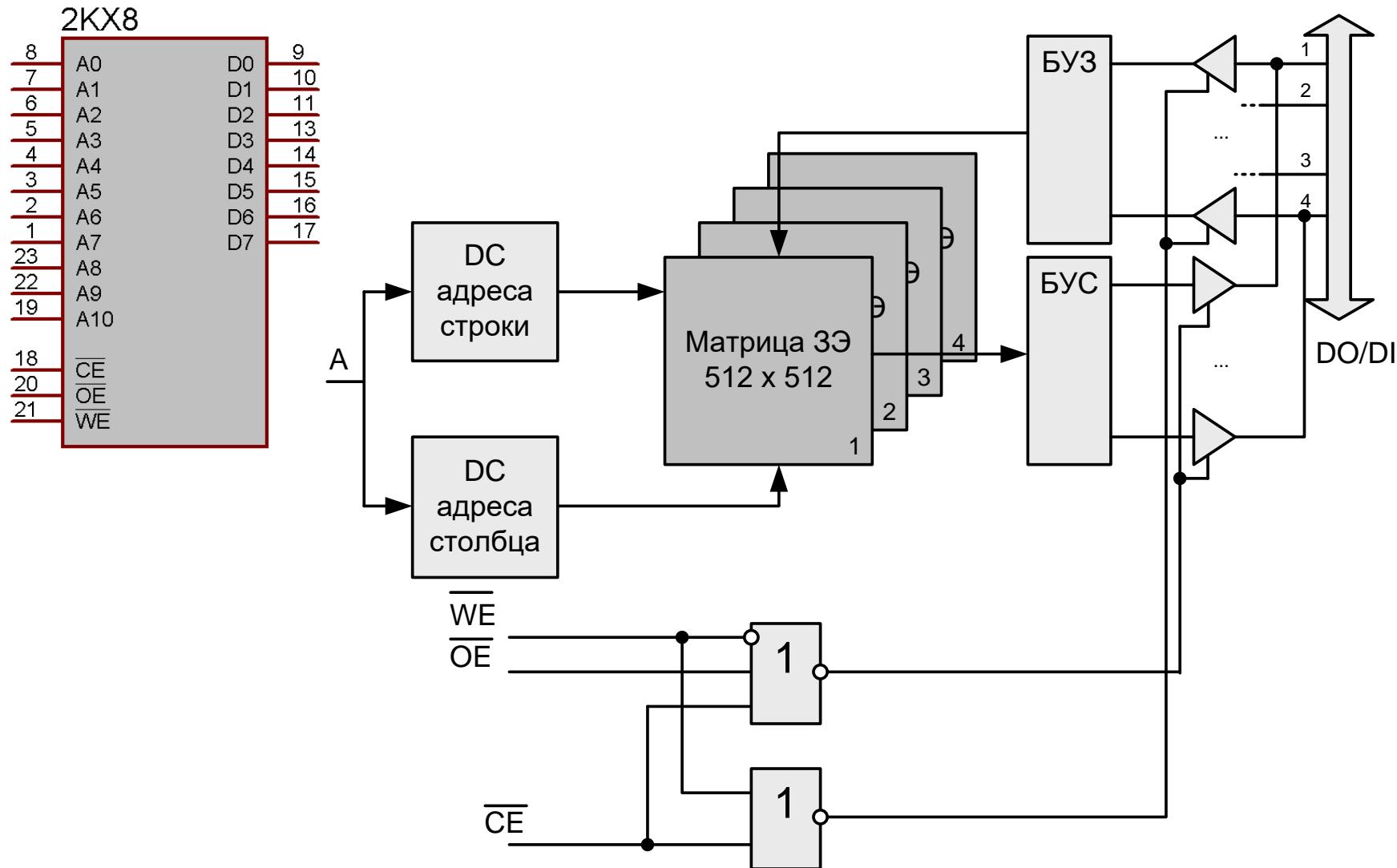
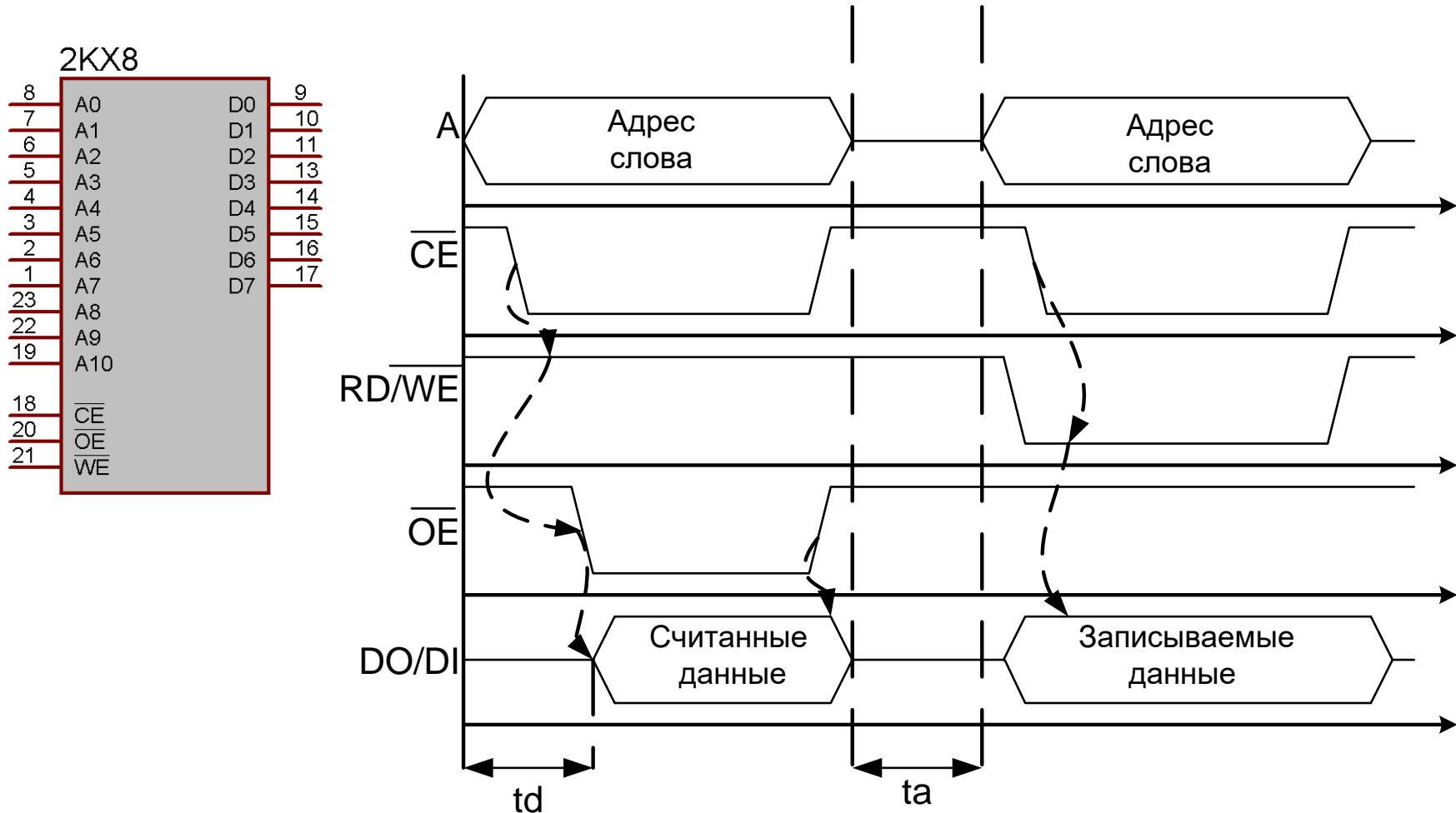
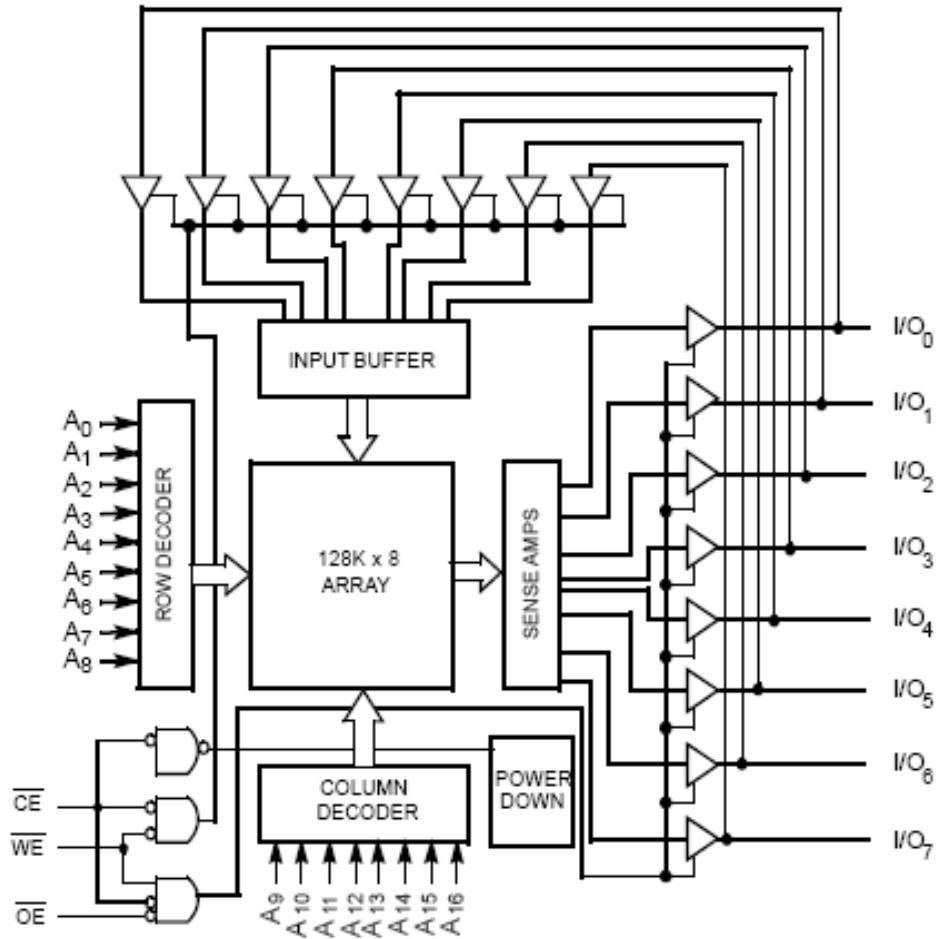


Диаграмма работы статической памяти





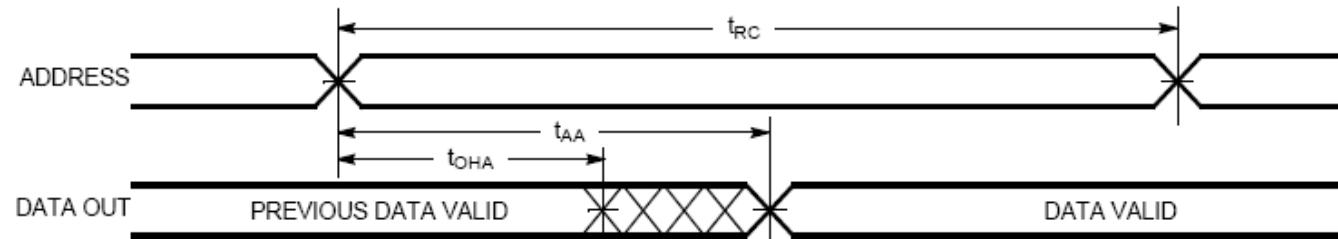
SOJ/TSOP II
Top View

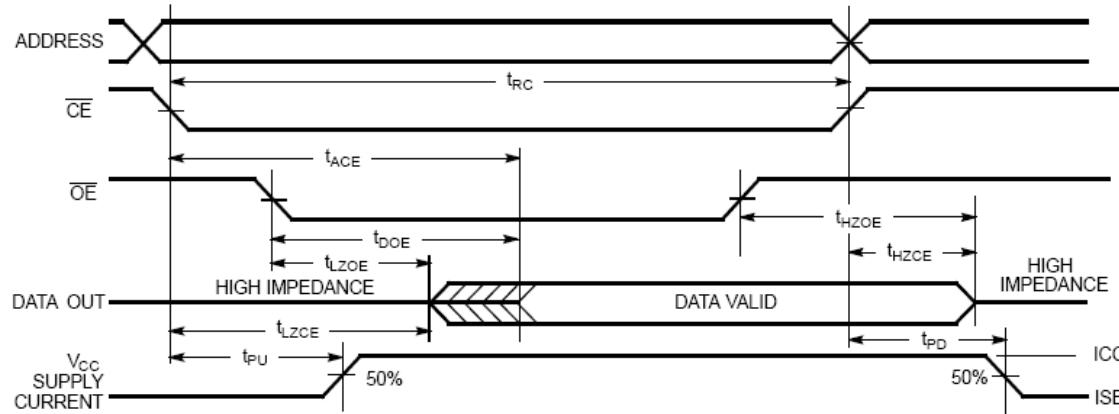
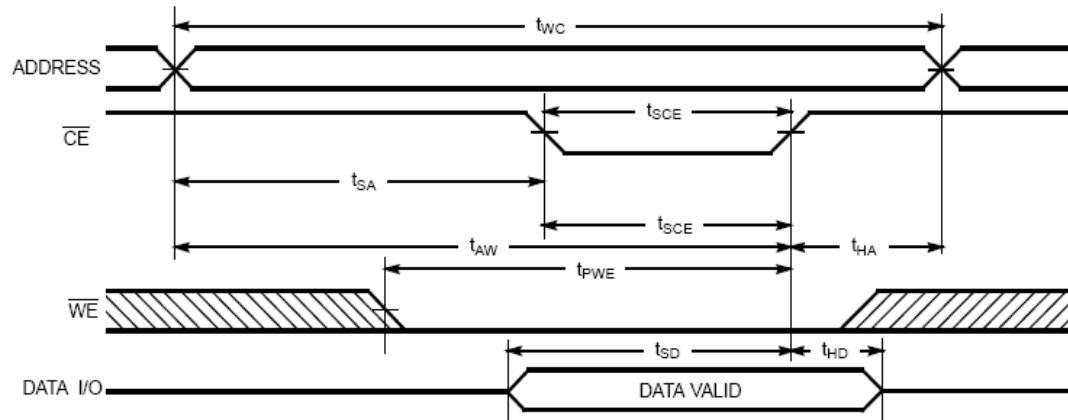
A ₀	1	32	A ₁₆
A ₁	2	31	A ₁₅
A ₂	3	30	A ₁₄
A ₃	4	29	A ₁₃
CE	5	28	OE
I/O ₀	6	27	I/O ₇
I/O ₁	7	26	I/O ₆
V _{CC}	8	25	V _{SS}
V _{SS}	9	24	V _{CC}
I/O ₂	10	23	I/O ₅
I/O ₃	11	22	I/O ₄
WE	12	21	A ₁₂
A ₄	13	20	A ₁₁
A ₅	14	19	A ₁₀
A ₆	15	18	A ₉
A ₇	16	17	A ₈

Truth Table

\overline{CE}	\overline{OE}	\overline{WE}	I/O₀-I/O₇	Mode	Power
H	X	X	High Z	Power-Down	Standby (I_{SB})
L	L	H	Data Out	Read	Active (I_{CC})
L	X	L	Data In	Write	Active (I_{CC})
L	H	H	High Z	Selected, Outputs Disabled	Active (I_{CC})

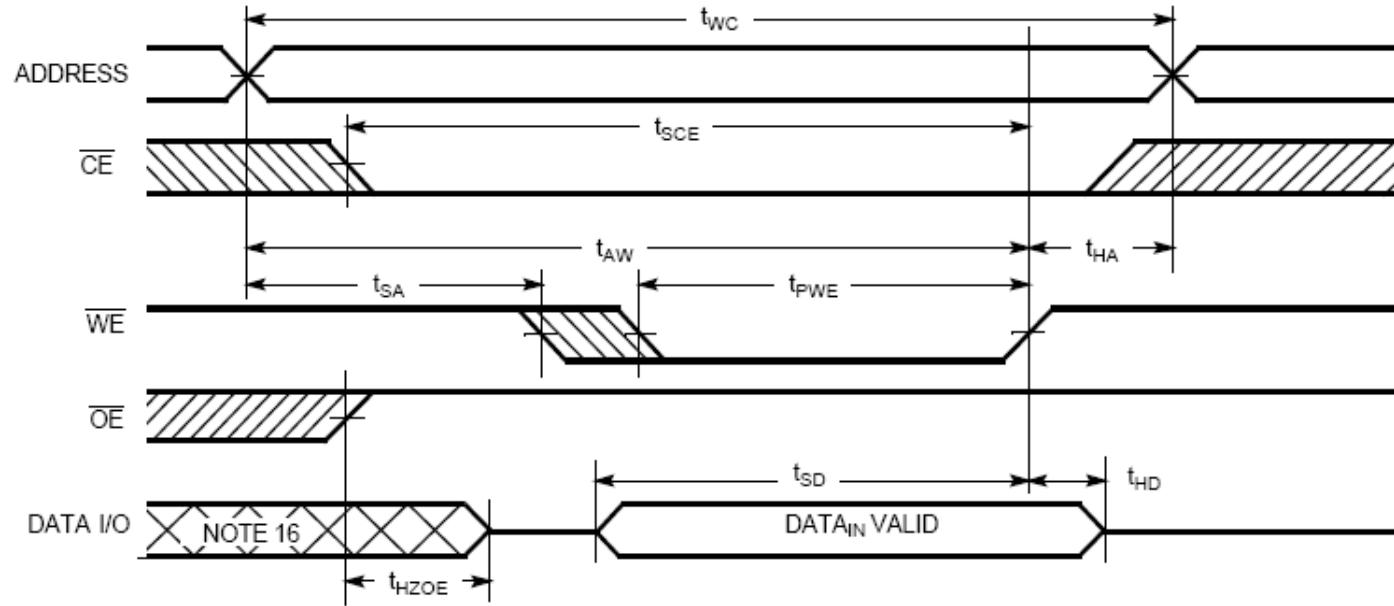
Read Cycle No. 1^[11, 12]



Read Cycle No. 2 (\overline{OE} Controlled)^[12, 13]**Write Cycle No. 1 (\overline{CE} Controlled)^[14, 15]****Notes:**

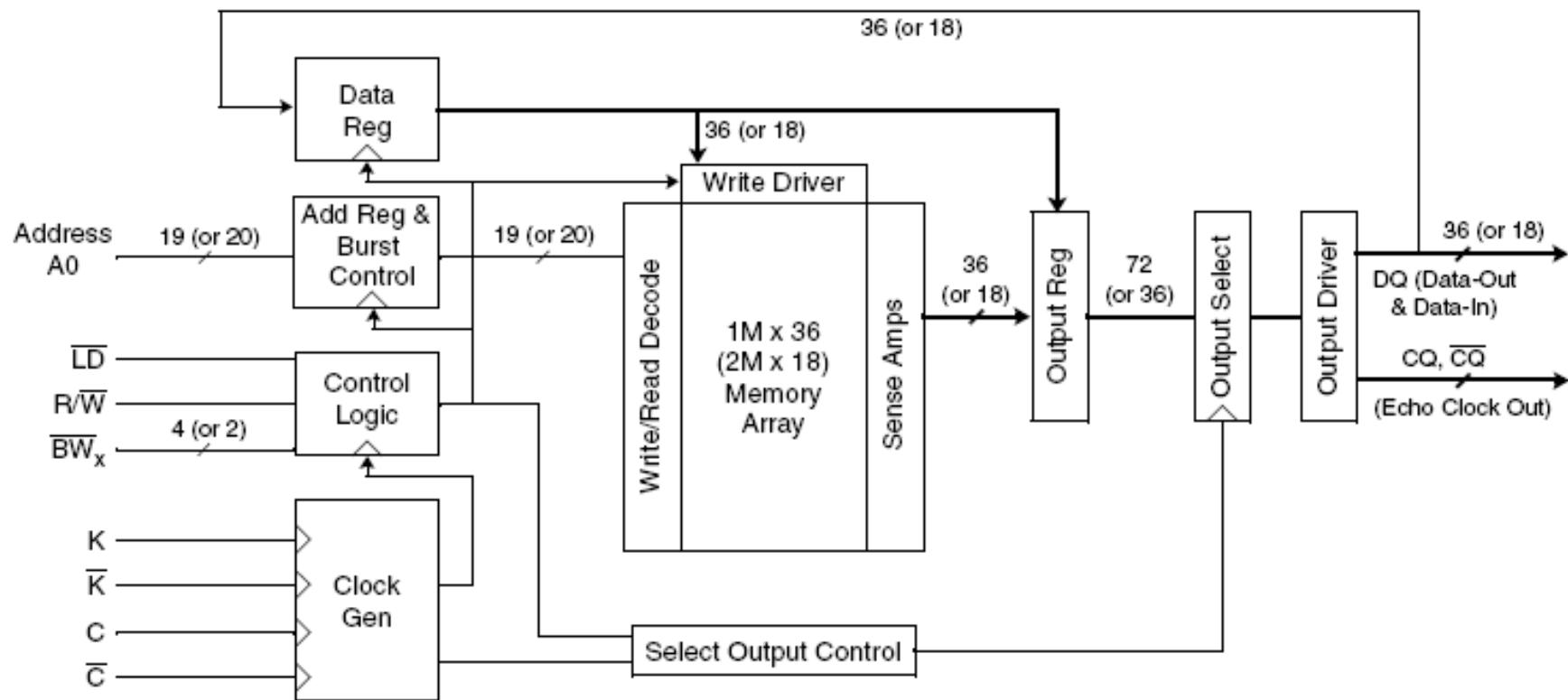
11. Device is continuously selected. \overline{OE} , $\overline{CE} = V_{IL}$.
12. WE is HIGH for read cycle.
13. Address valid prior to or coincident with \overline{CE} transition LOW.
14. Data I/O is high impedance if $\overline{OE} = V_{IH}$.
15. If CE goes HIGH simultaneously with WE going HIGH, the output remains in a high-impedance state.

Write Cycle No. 2 (\overline{WE} Controlled, \overline{OE} HIGH During Write)^[14, 15]



36 Mb (1M x 36 & 2M x 18) DDR-II (Burst of 2) CIO Synchronous SRAMs

ISSI®



36 Mb (1M x 36 & 2M x 18) DDR-II (Burst of 2) CIO Synchronous SRAMs

ISSI®

Features

- 1M x 36 or 2M x 18.
- On-chip delay-locked loop (DLL) for wide data valid window.
- Common data input/output bus.
- Synchronous pipeline read with self-timed late write operation.
- Double data rate (DDR-II) interface for read and write input ports.
- Fixed 2-bit burst for read and write operations.
- Clock stop support.
- Two input clocks (K and \bar{K}) for address and control registering at rising edges only.
- Two input clocks (C and \bar{C}) for data output control.
- Two echo clocks (CQ and \bar{CQ}) that are delivered simultaneously with data.
- +1.8V core power supply and 1.5, 1.8V V_{DDQ} , used with 0.75, 0.9V V_{REF}
- HSTL input and output levels.
- Registered addresses, write and read controls, byte writes, data in, and data outputs.
- Full data coherency.
- Boundary scan using limited set of JTAG 1149.1 functions.
- Byte write capability.
- Fine ball grid array (FBGA) package
 - 15mm x 17mm body size
 - 1mm pitch
 - 165-ball (11 x 15) array
- Programmable impedance output drivers via 5x user-supplied precision resistor.

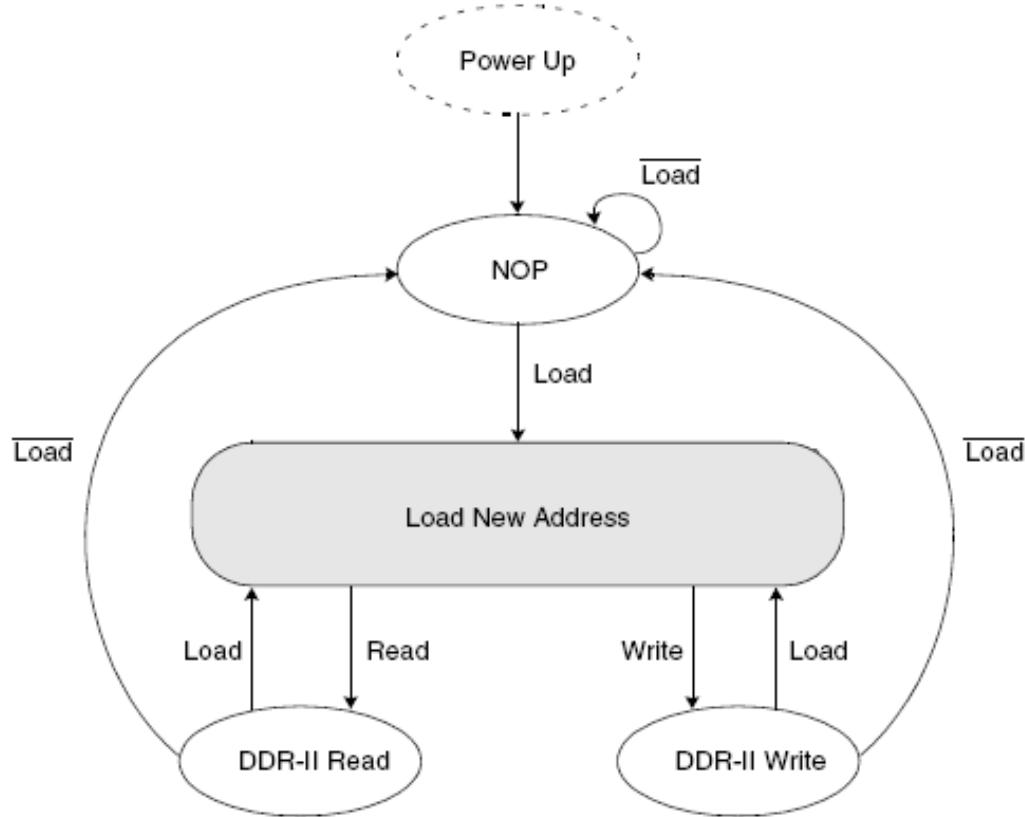
36 Mb (1M x 36 & 2M x 18) DDR-II (Burst of 2) CIO Synchronous SRAMs

ISSI®

Symbol	Pin Number	Description
K, \bar{K}	6B, 6A	Input clock.
C, \bar{C}	6P, 6R	Input clock for output data control.
CQ, \bar{CQ}	11A, 1A	Output echo clock.
Doff	1H	DLL disable when low.
SA ₀	6C	Burst count address input.
SA	9A, 4B, 8B, 5C, 7C, 5N, 6N, 7N, 4P, 5P, 7P, 8P, 3R, 4R, 5R, 7R, 8R, 9R	1M x 36 address inputs.
SA	3A, 9A, 4B, 8B, 5C, 7C, 5N, 6N, 7N, 4P, 5P, 7P, 8P, 3R, 4R, 5R, 7R, 8R, 9R	2M x 18 address inputs.
DQ0–DQ8 DQ9–DQ17 DQ18–DQ26 DQ27–DQ35	11P, 11M, 11L, 11K, 11J, 11F, 11E, 11C, 11B 10P, 11N, 10M, 10K, 10J, 11G, 10E, 11D, 10C 3B, 3D, 3E, 3F, 3G, 3K, 3L, 3N, 3P 2B, 3C, 2D, 2F, 2G, 3J, 2L, 3M, 2N	1M x 36 DQ pins
DQ0–DQ8 DQ9–DQ17	11P, 10M, 11L, 11K, 10J, 11F, 11E, 10C, 11B 2B, 3D, 3E, 2F, 3G, 3K, 2L, 3N, 3P	2M x 18 DQ pins
R/W	4A	Read/write control. Read when active high.
LD	8A	Synchronizes load. Loads new address when low.
\overline{BW}_0 , \overline{BW}_1 , \overline{BW}_2 , \overline{BW}_3	7B, 7A, 5A, 5B	1M x 36 byte write control, active low.
\overline{BW}_0 , \overline{BW}_1	7B, 5A	2M x 18 byte write control, active low.

36 Mb (1M x 36 & 2M x 18) DDR-II (Burst of 2) CIO Synchronous SRAMs

ISSI®

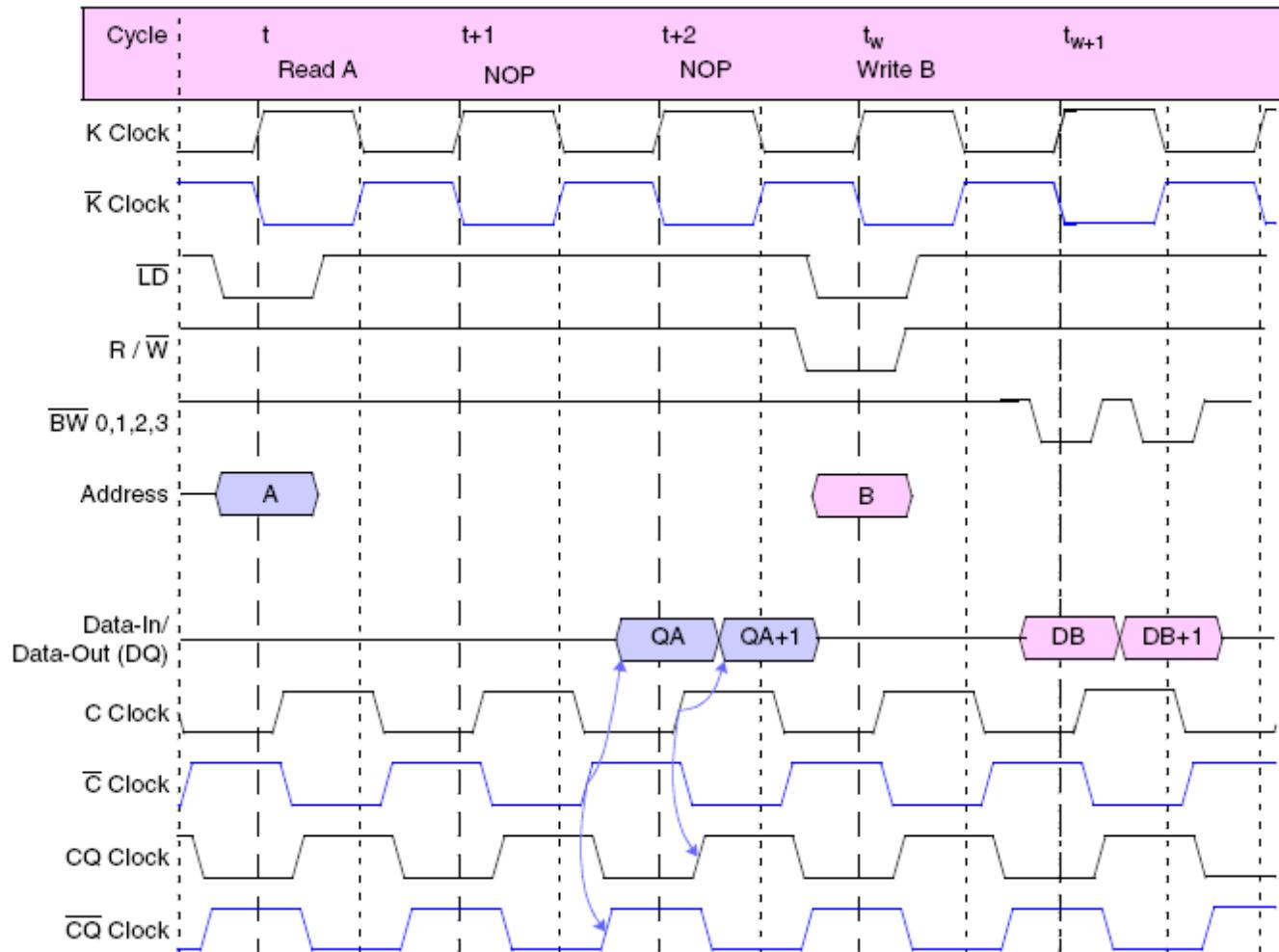


- Notes:
1. Internal burst counter is fixed as two-bit linear; that is, when first address is A0+0, next internal burst address is A0+1.
 2. *Read* refers to read active status with R/W = high.
 3. *Write* refers to write active status with R/W = low.
 4. *Load* refers to read new address active status with \overline{LD} = low.
 5. *Load* is read new address inactive status with \overline{LD} = high.

36 Mb (1M x 36 & 2M x 18) DDR-II (Burst of 2) CIO Synchronous SRAMs

ISSI®

ПРИМЕР



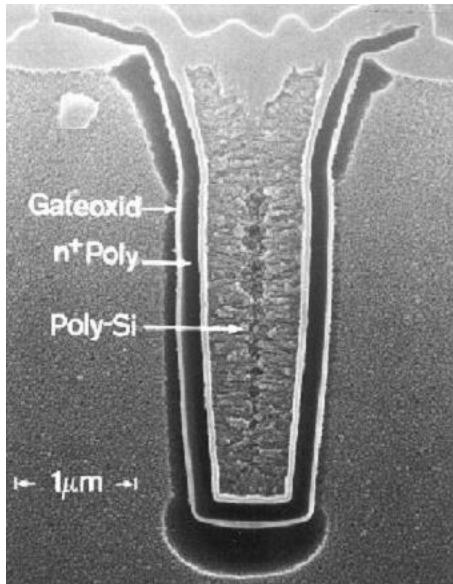
Динамические ЗУ с произвольной выборкой (DRAM)

DRAM для обращения по произвольным адресам

DRAM, RLDRAM

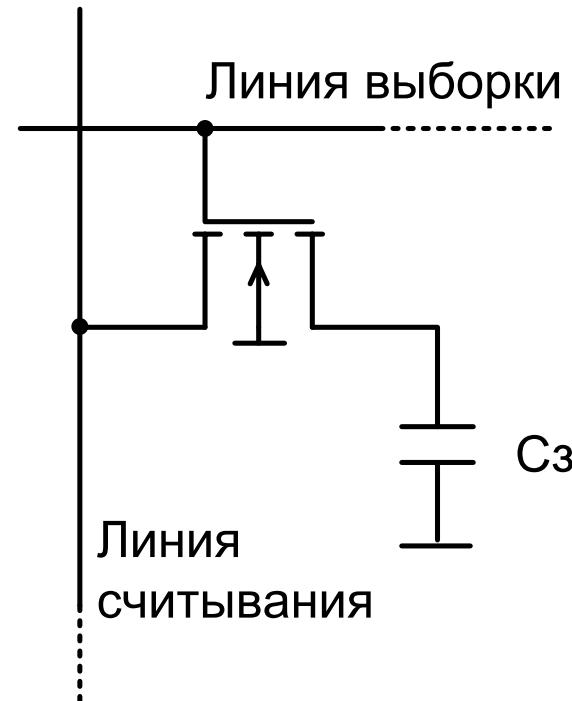
DRAM, оптимизированные для обращения по последовательным адресам:

FPM DRAM, EDO DRAM, BEDO DRAM, SDRAM, DDR SDRAM, RDRAM

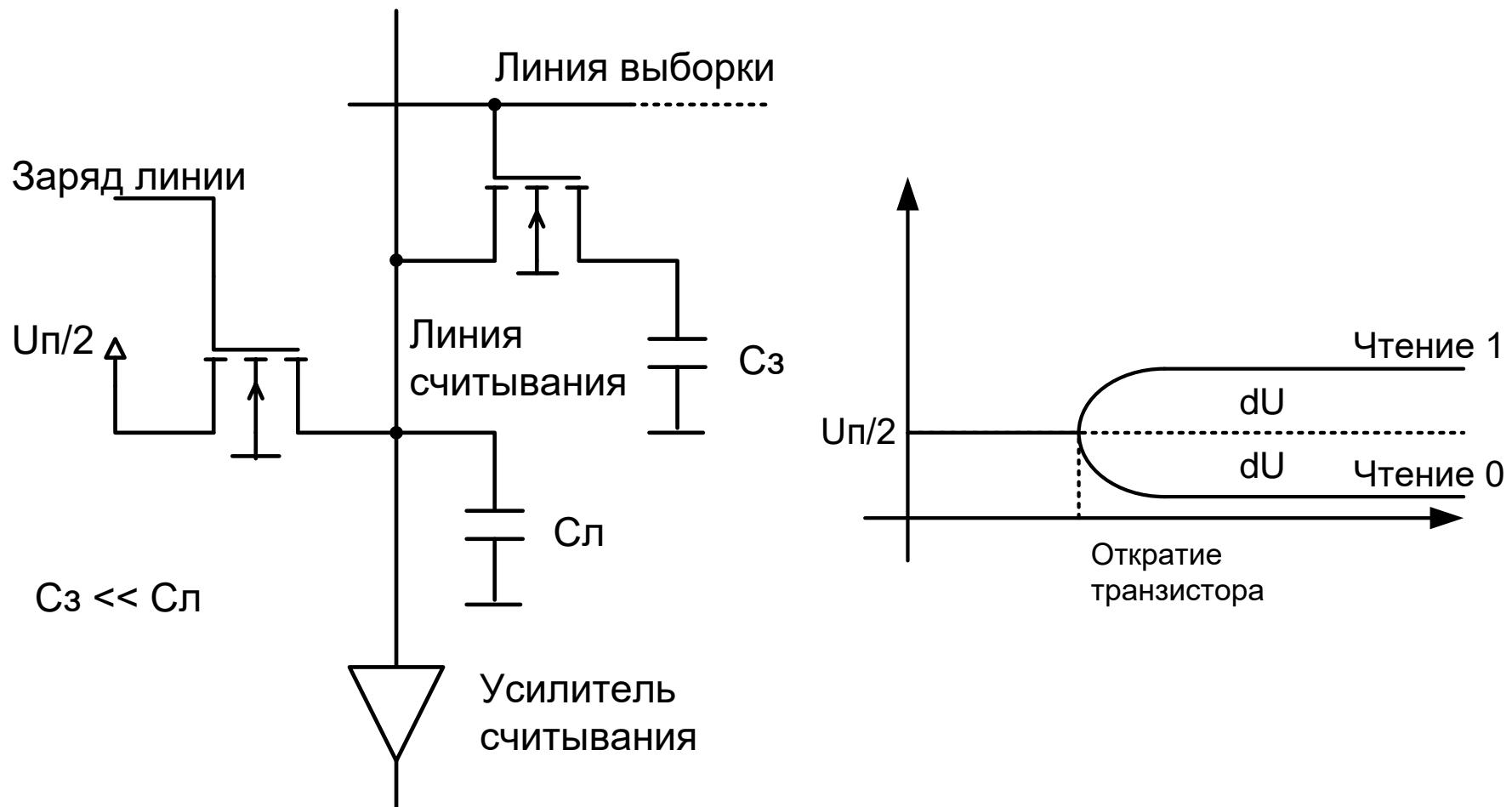


При выборке строки все Сз подключаются к линиям считывания.

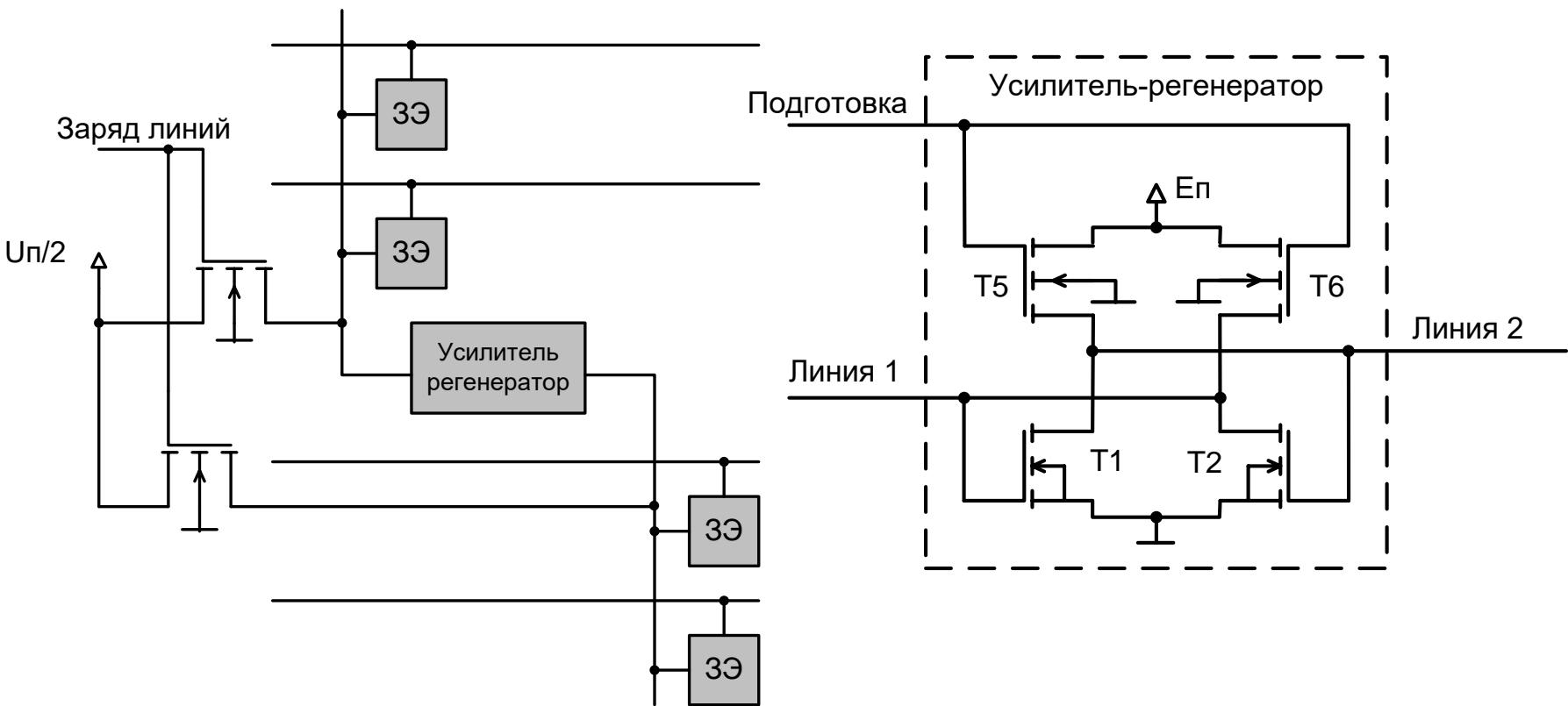
После считывания необходимо произвести обратную запись информации – регенерацию.



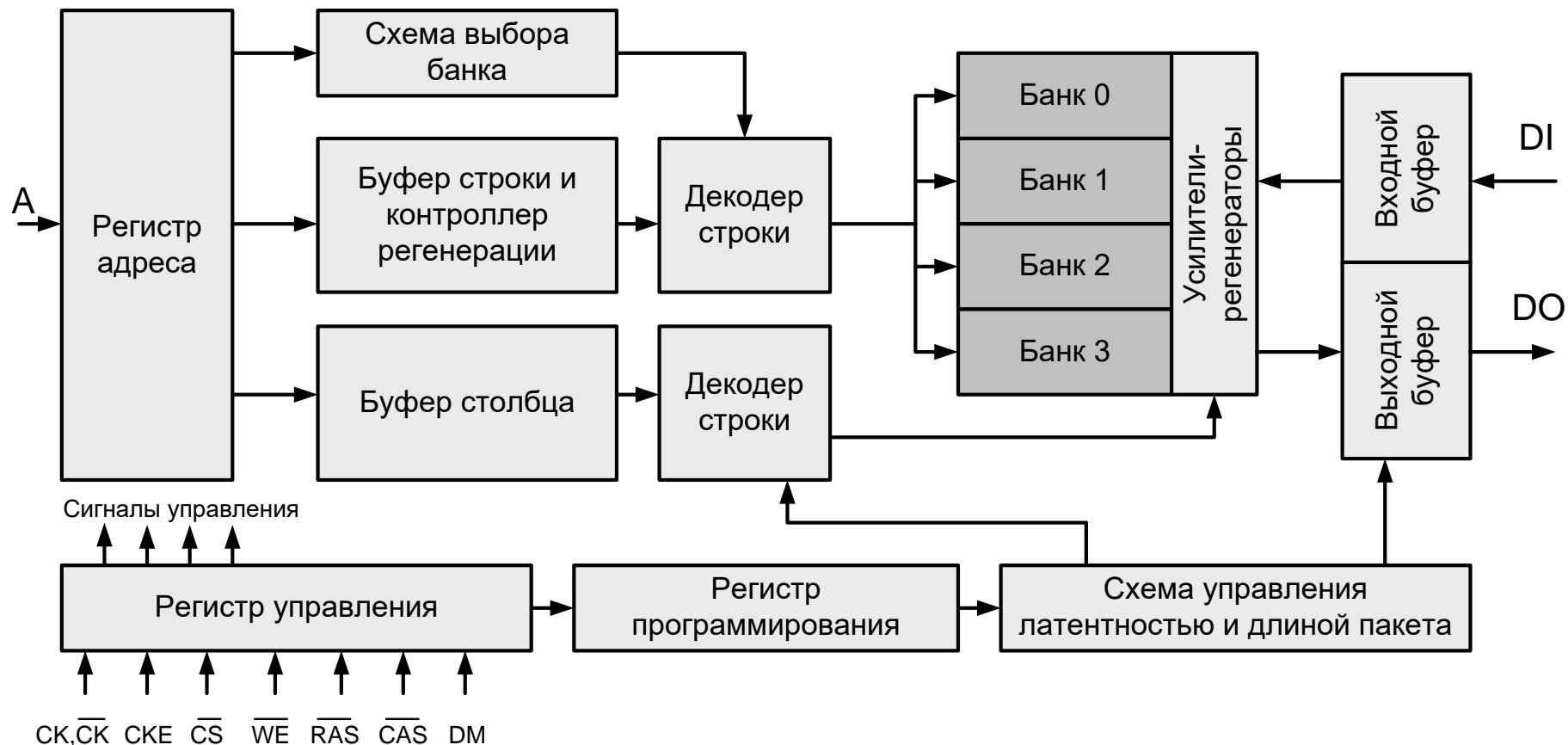
Процесс считывания в DRAM

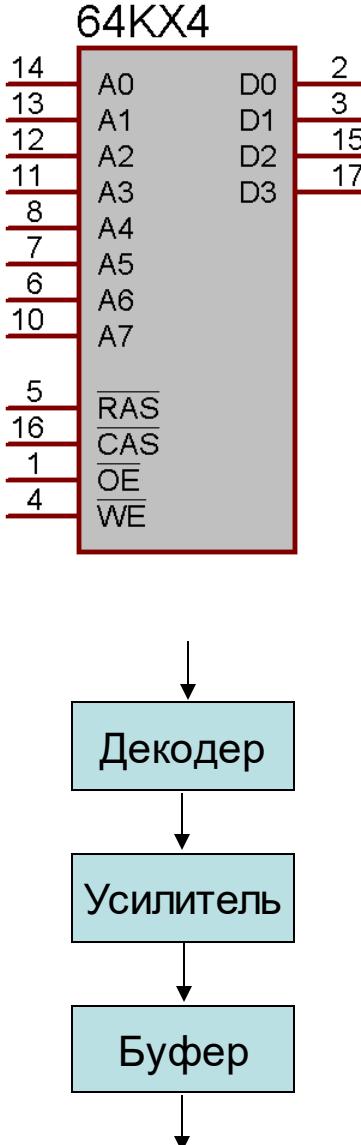


Принцип действия усилителя-регенератора



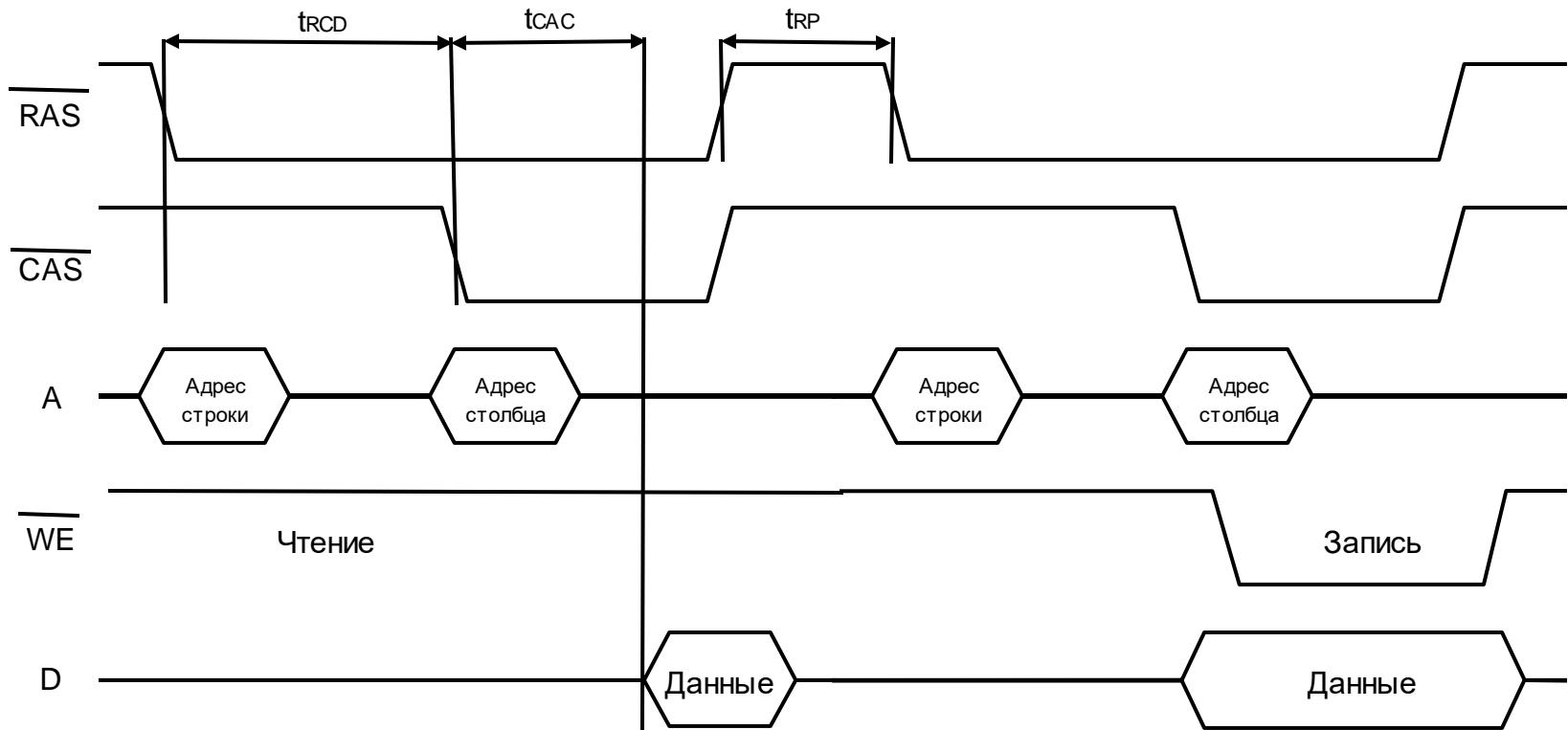
Микросхема динамической памяти





- Функциональные возможности SDRAM памяти:
- Многобанковая организация.
 - Командный режим работы.
 - Команды пакетного чтения/записи.
 - Использование чередования банков при последовательном увеличении адресов.
 - Команды пакетного чтения/записи с авто-подзарядом.
 - Возможность останова чтения/записи по режиму регенерации.
 - Возможность останова чтения/записи по новому запросу чтения/записи.
 - Управление маскированием шины данных по сигналу DQM.
 - Минимальное время (1 CLK) между последовательными командами.
 - Команда PrechargeAll.
 - CAS латентность 2 и 3 CLK.
 - Длина пакета 1,2 и 4 слова.
 - Команда само-регенерации.
 - Режим энергосбережения.

Диаграмма работы DRAM памяти



t_{RCD} – RAS to CAS Delay.

t_{RP} – RAS Precharge.

t_{CAC} – CAS Delay.

Контроллер динамической памяти

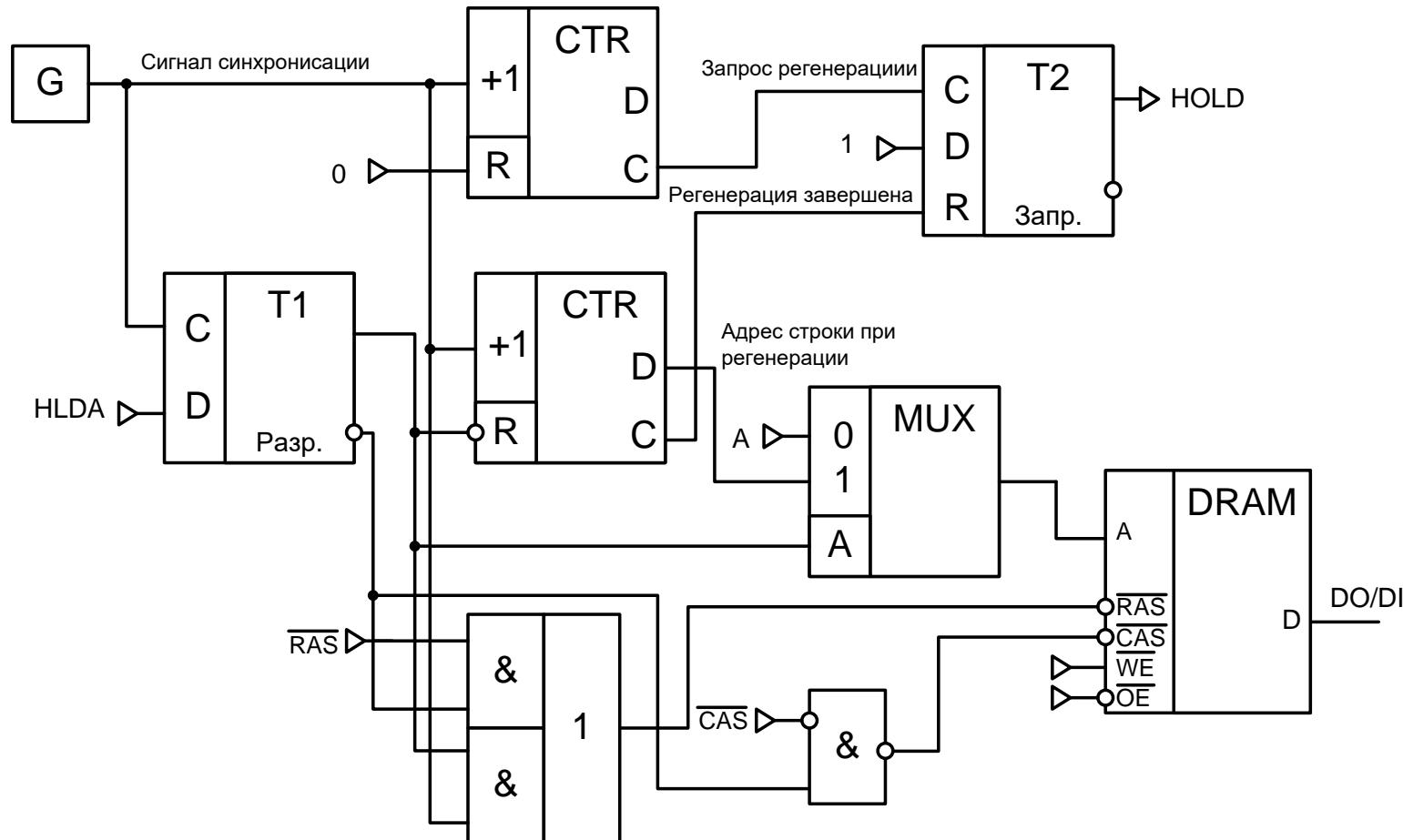


Диаграмма работы FPM DRAM памяти

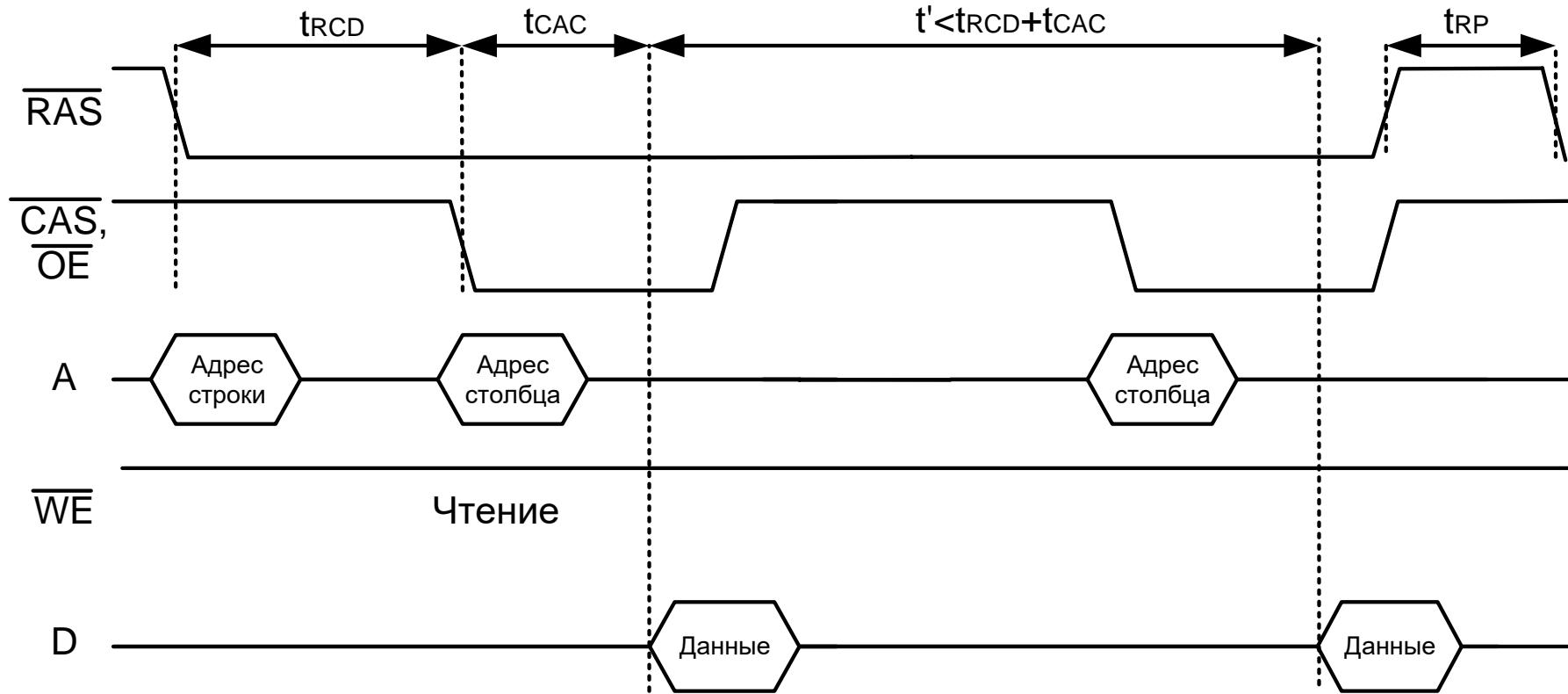


Диаграмма работы BEDO DRAM памяти

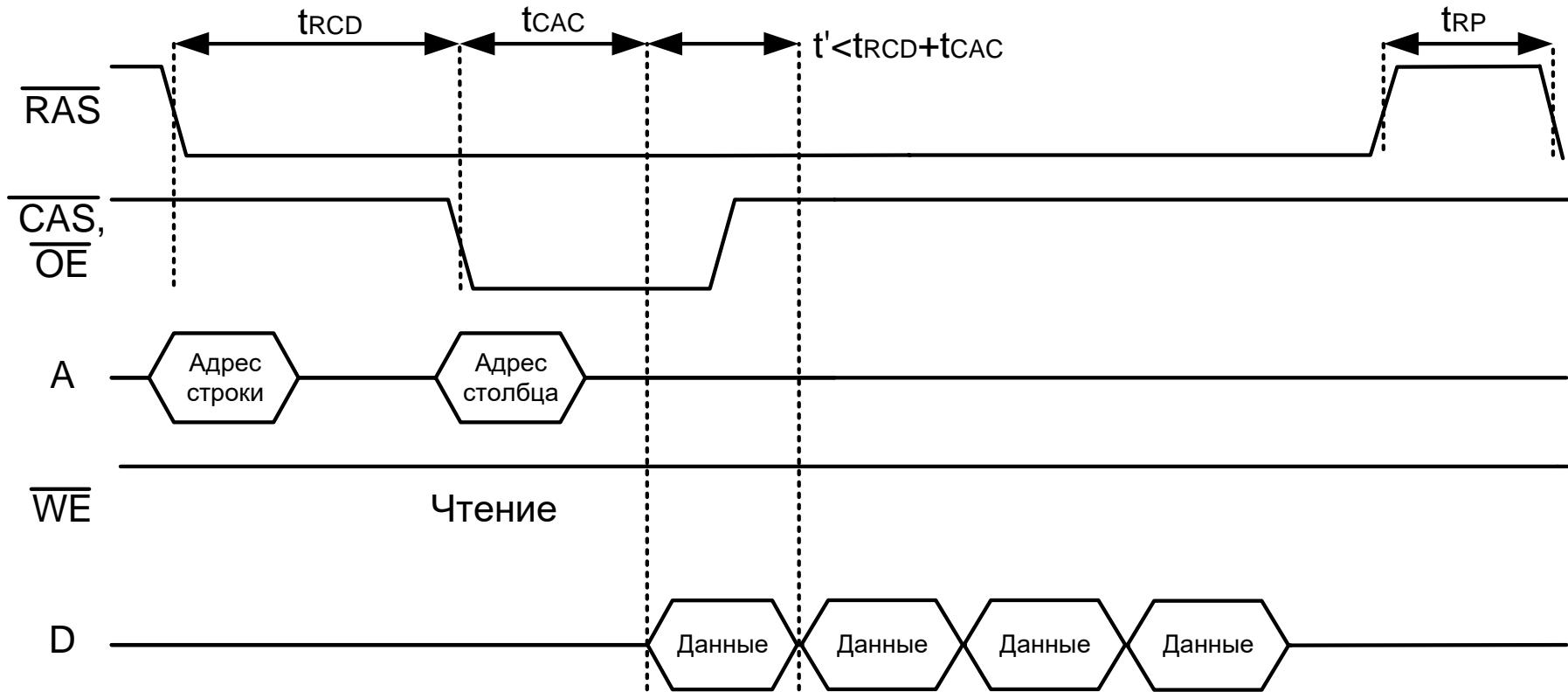


Диаграмма работы SDRAM памяти

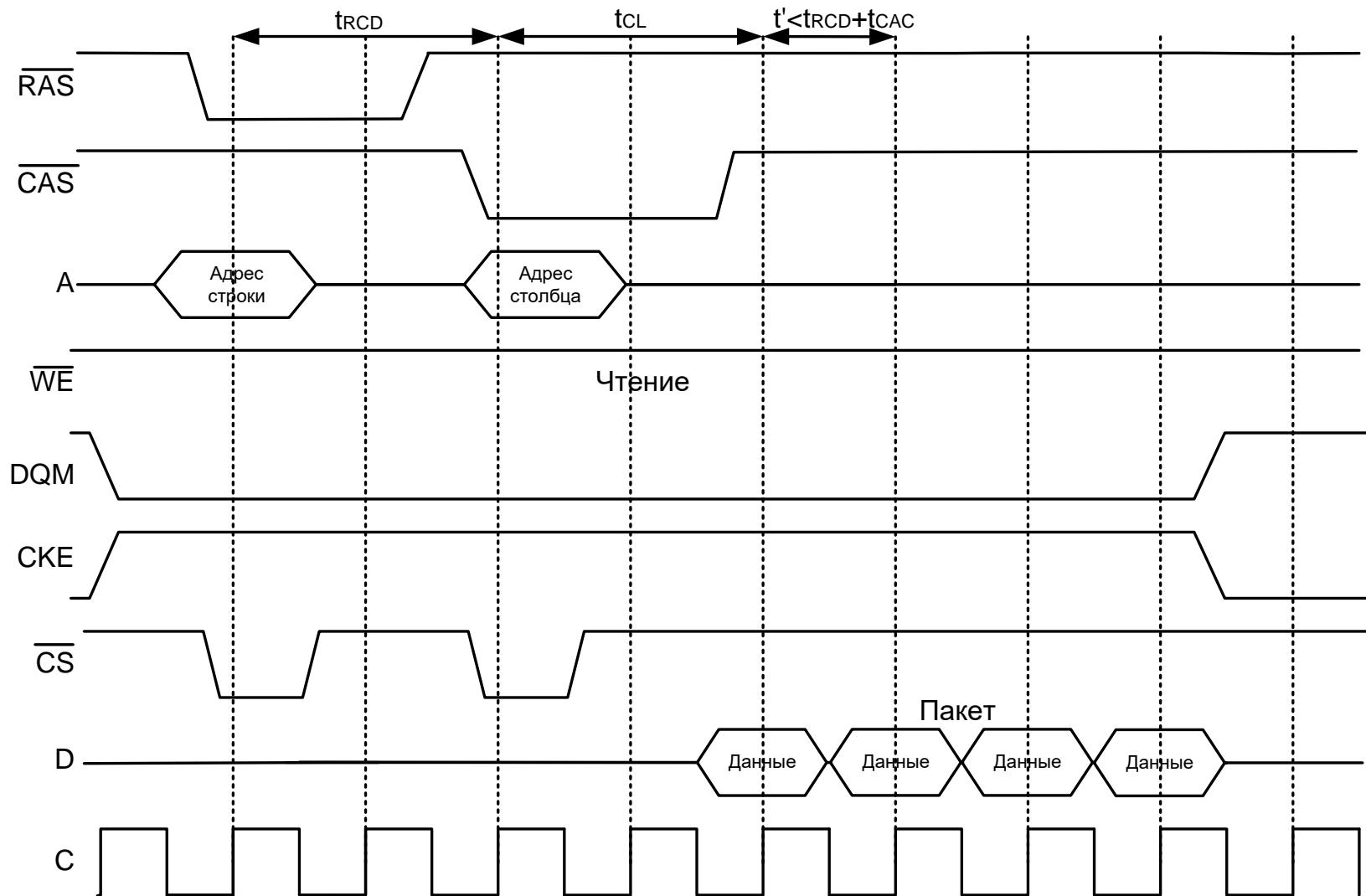
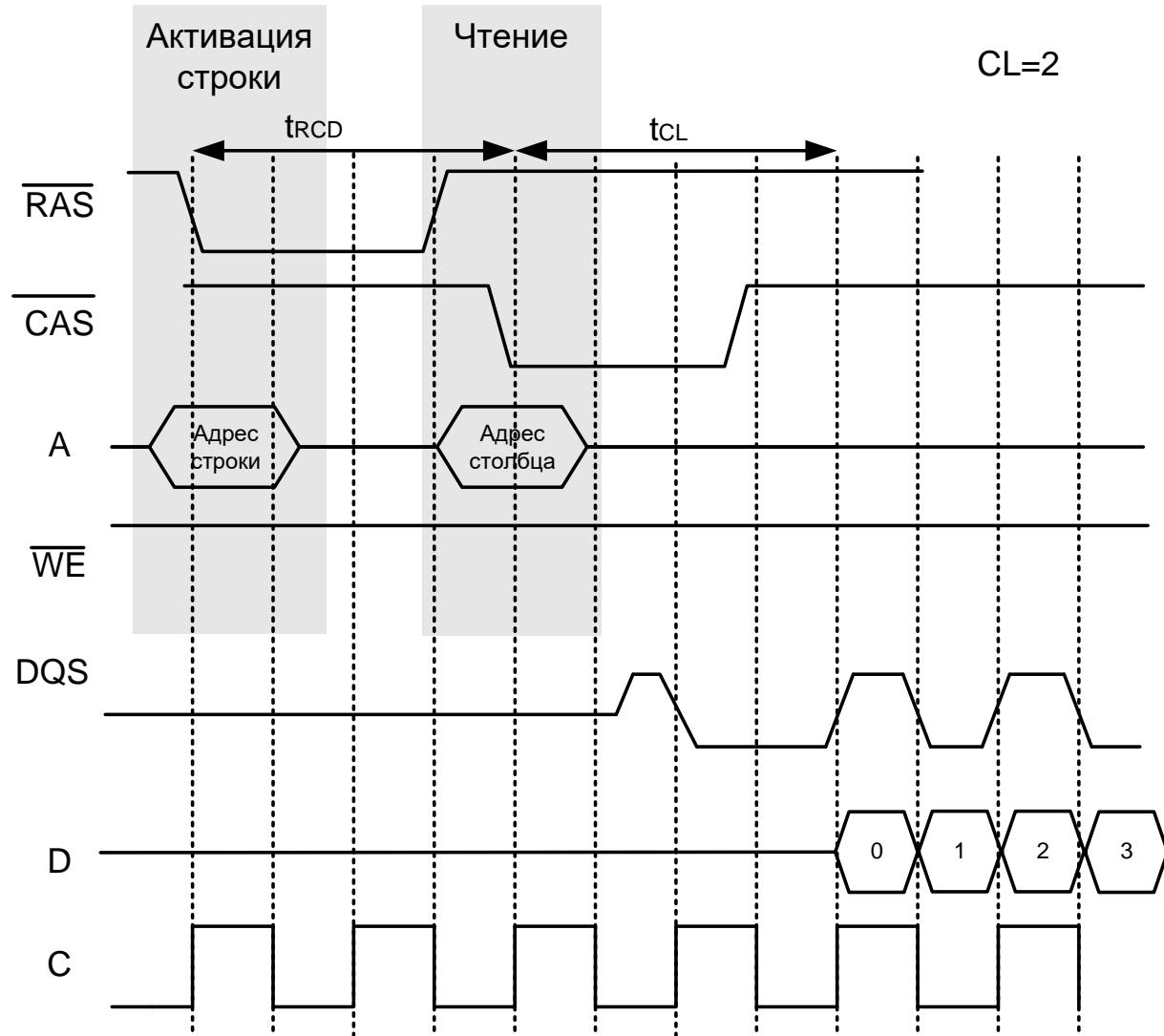
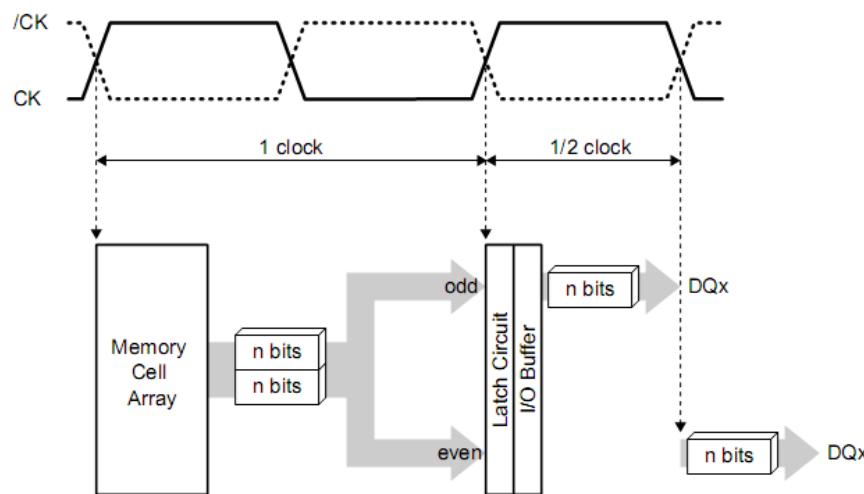


Диаграмма работы DDR SDRAM памяти



Способы повышения производительности RAM

- Синхронизация.
- Конвейеризация.
- Пакетный режим обмена.
- Ускорение реверса шины.
- Чередование банков при обращении по последовательным адресам.
- Удвоение скорости.



Регистр DDR

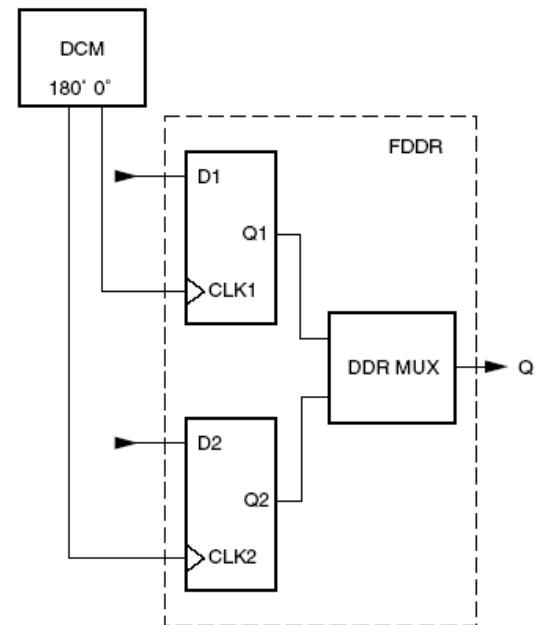
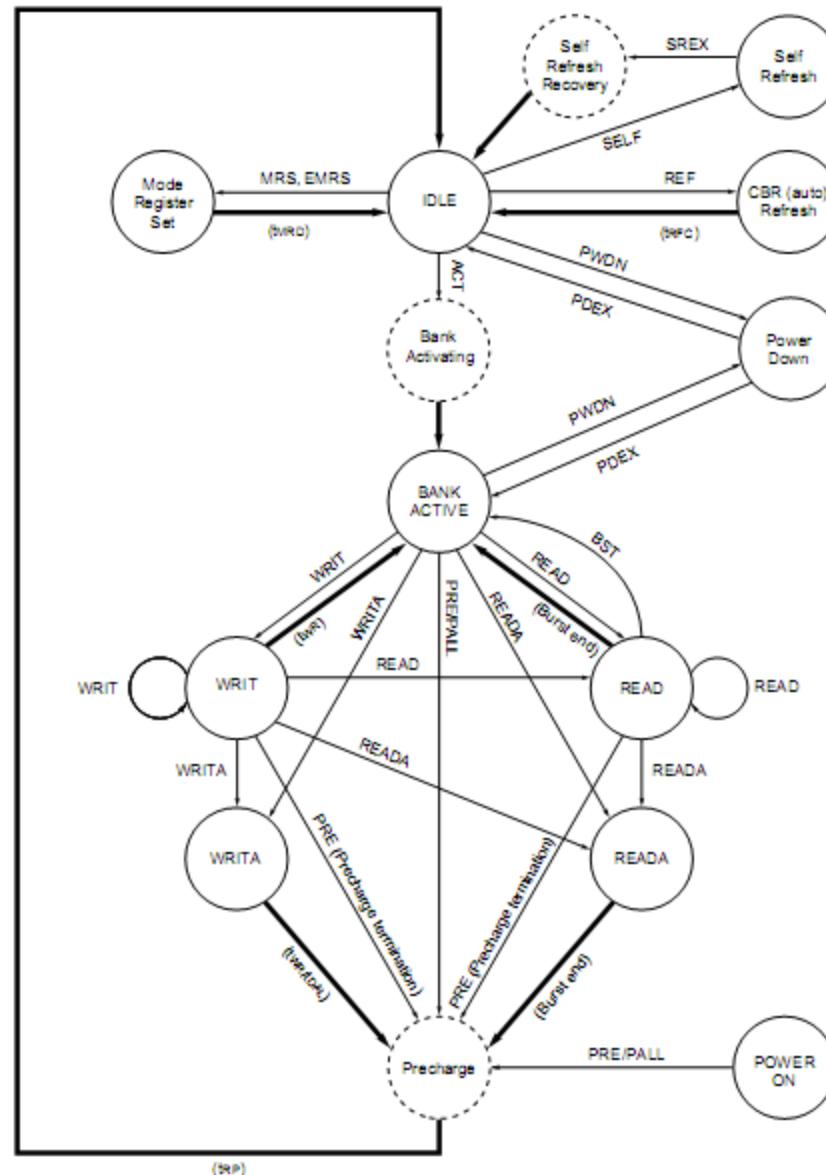


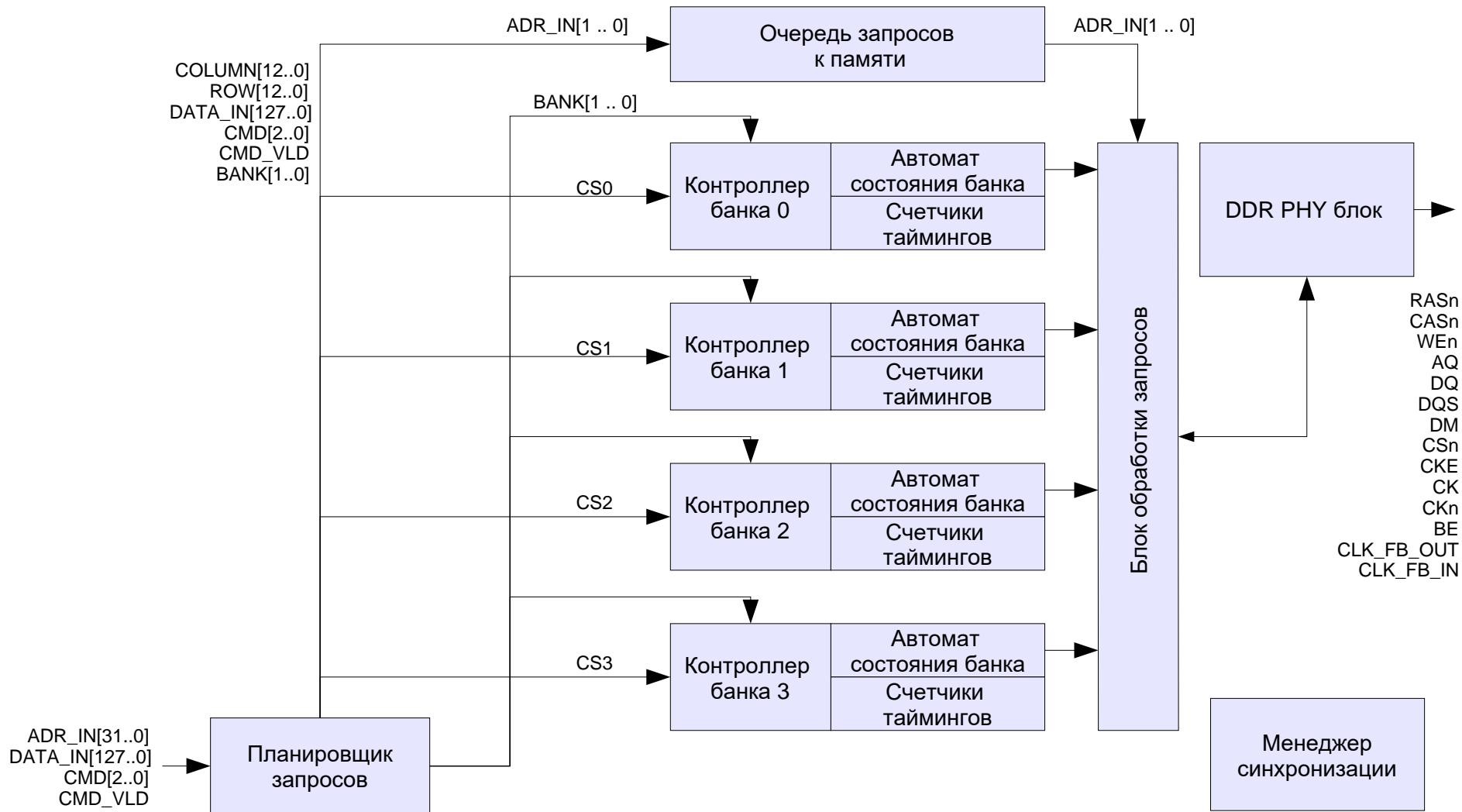
Диаграмма состояний УА DDR SDRAM



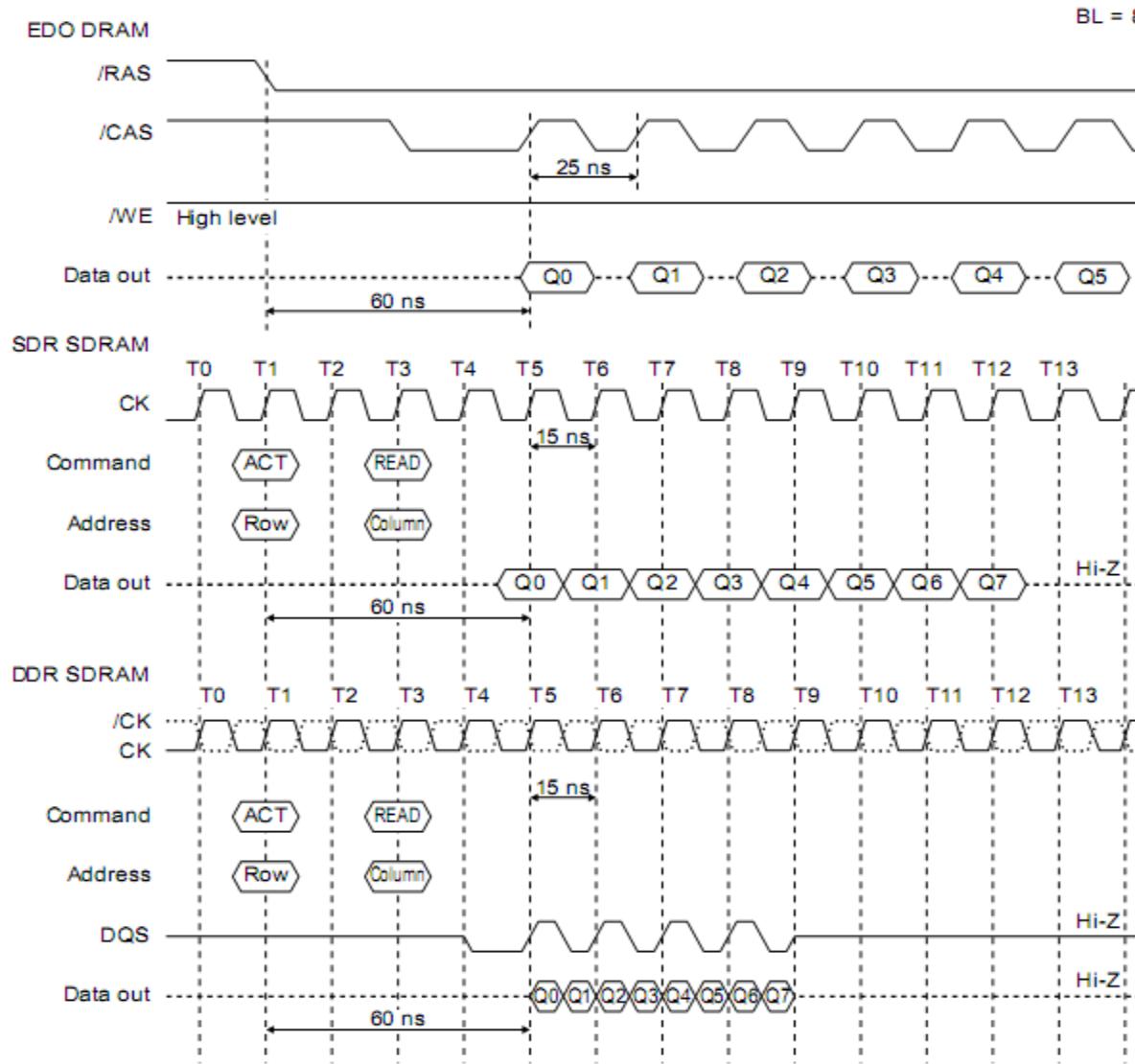
Automatic sequence
Manual input

Архитектура ЭЕ

Контроллер DDR/DDR2



Сравнение EDO RAM, SDRAM, DDR SDRAM



Document No. E0234E30 (Ver.3.0)
Date Published April 2002 (K) Japan
URL: <http://www.elpida.com>

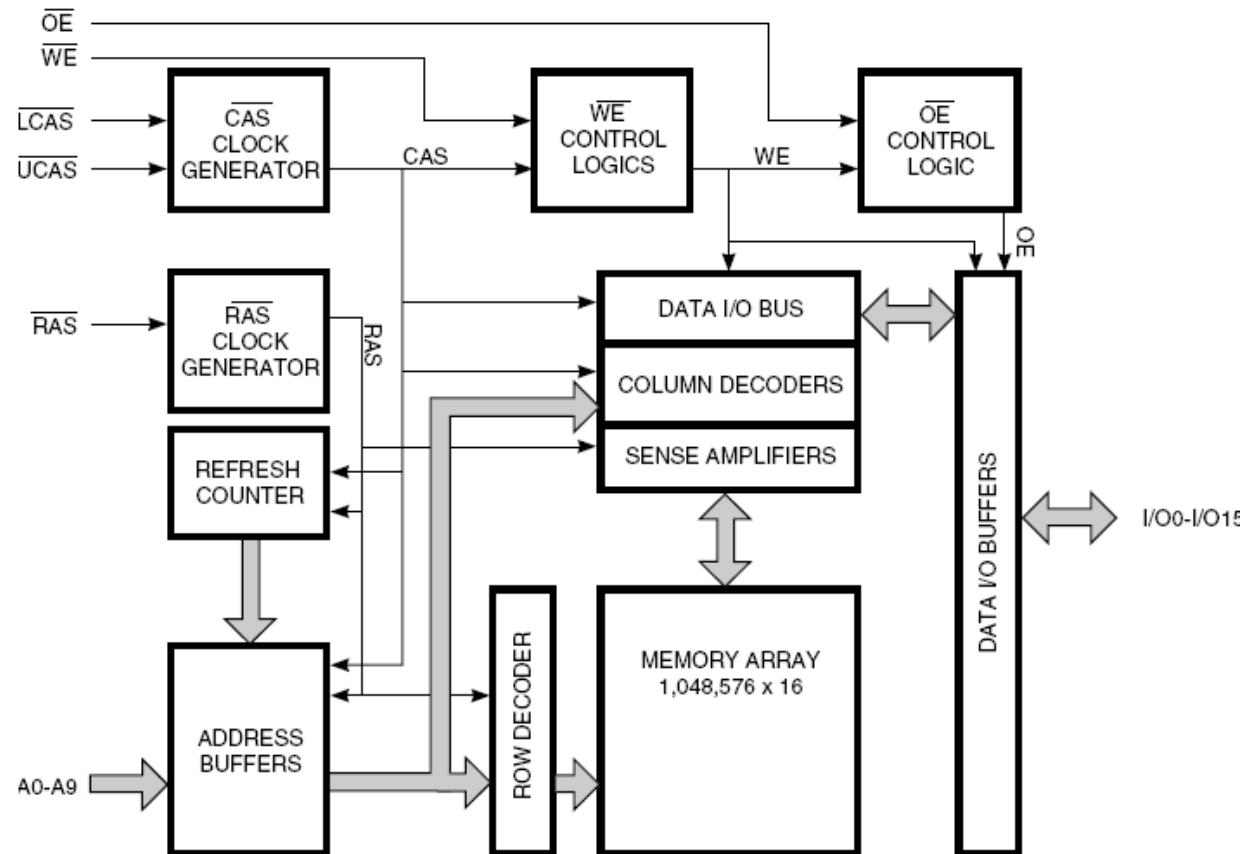
Сравнение DDR и DDR2

DDR память

DDR2 память

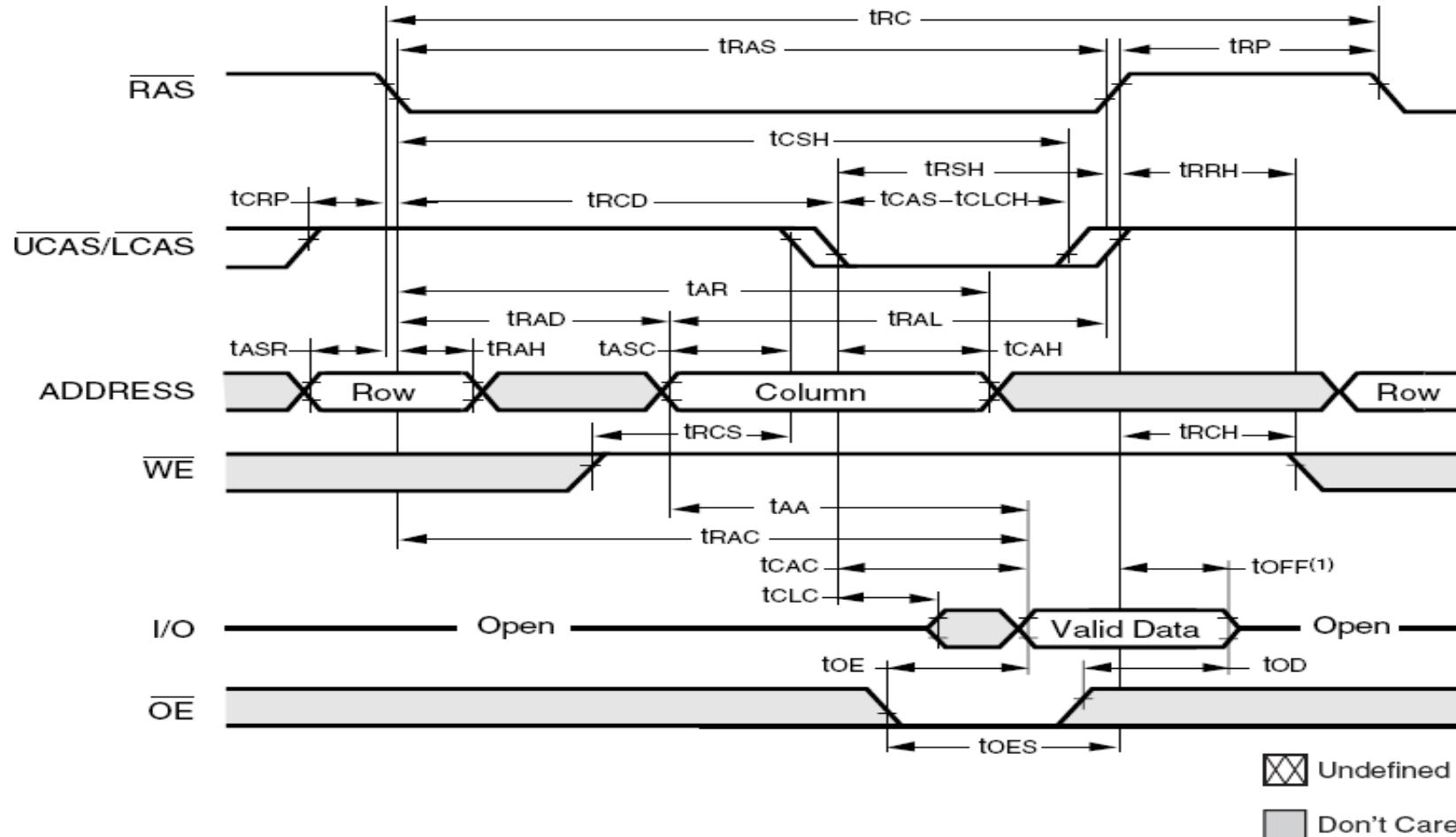
1M x 16 (16-MBIT) DYNAMIC RAM
WITH EDO PAGE MODE

December 2005



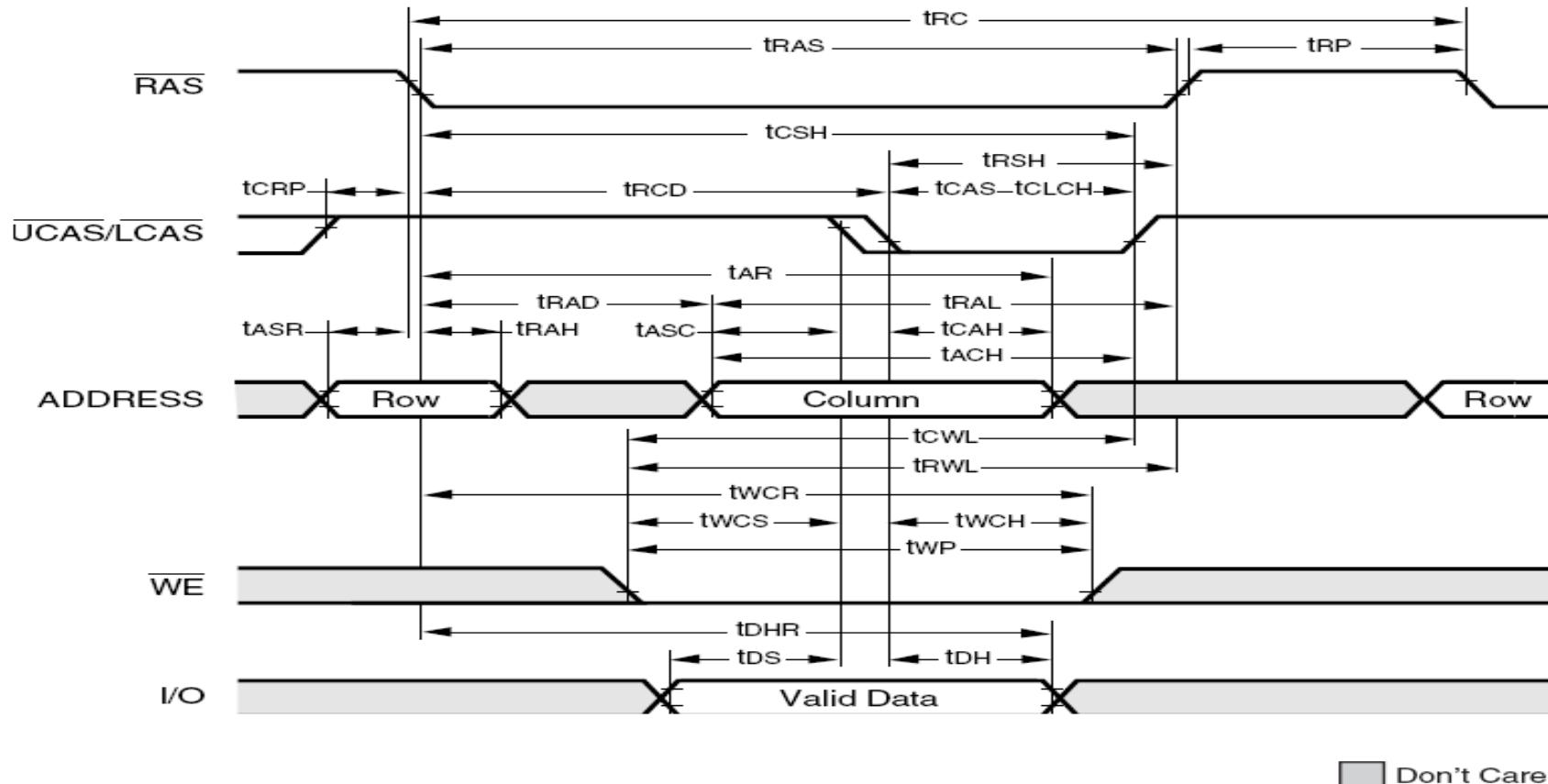
1M x 16 (16-MBIT) DYNAMIC RAM
WITH EDO PAGE MODE

December 2005

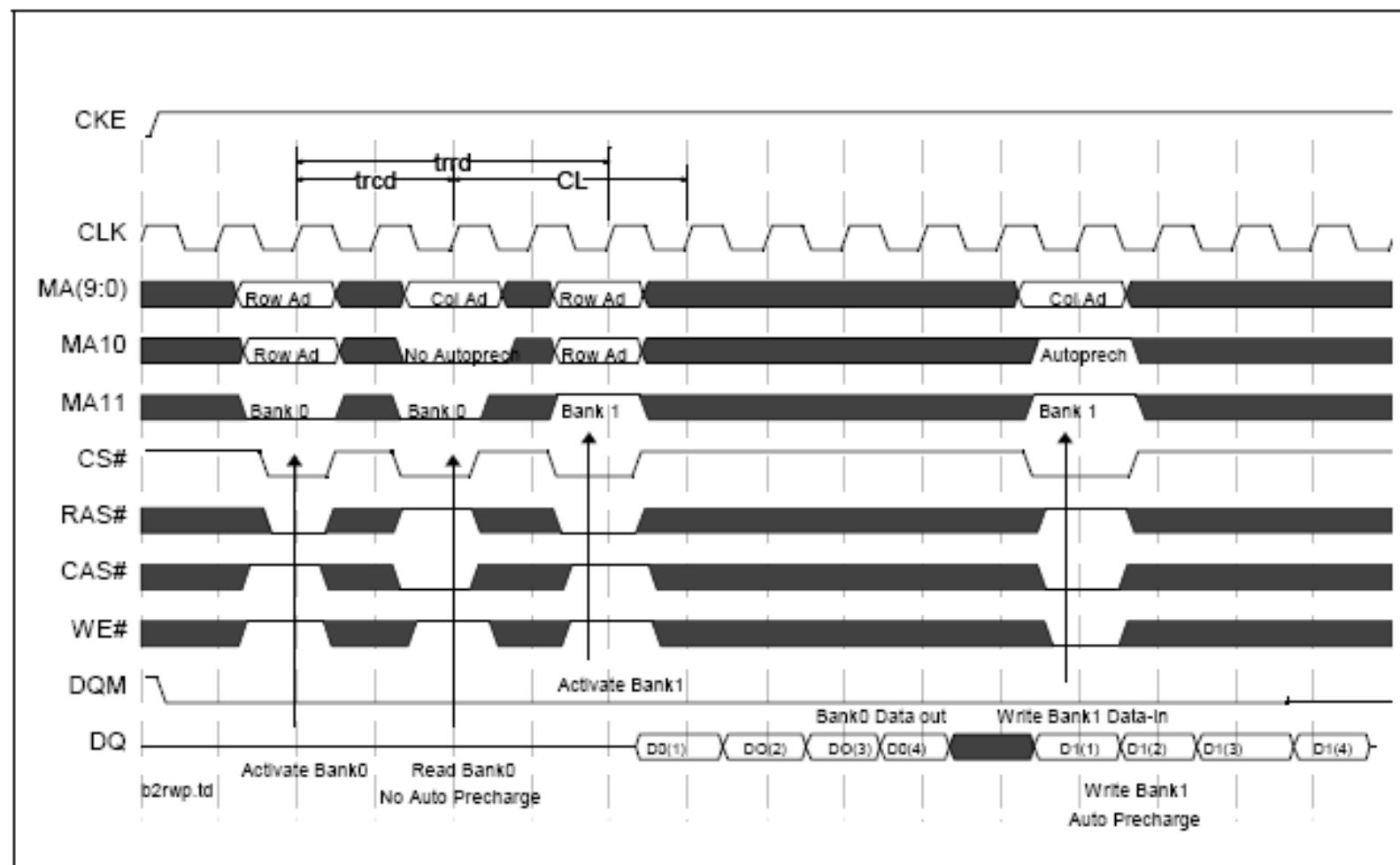


1M x 16 (16-MBIT) DYNAMIC RAM
WITH EDO PAGE MODE

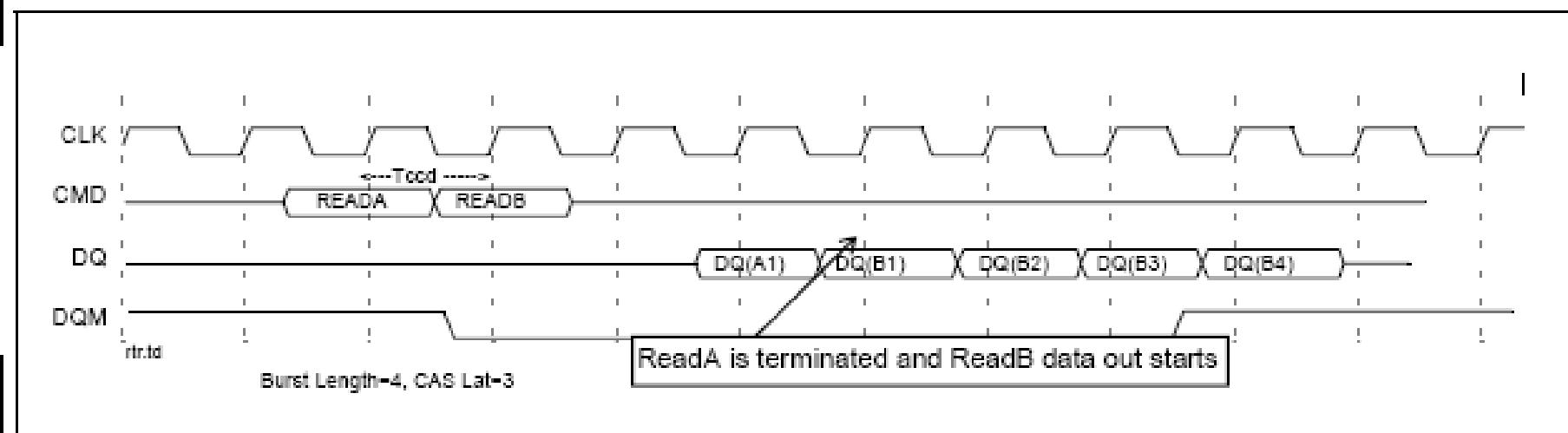
December 2005



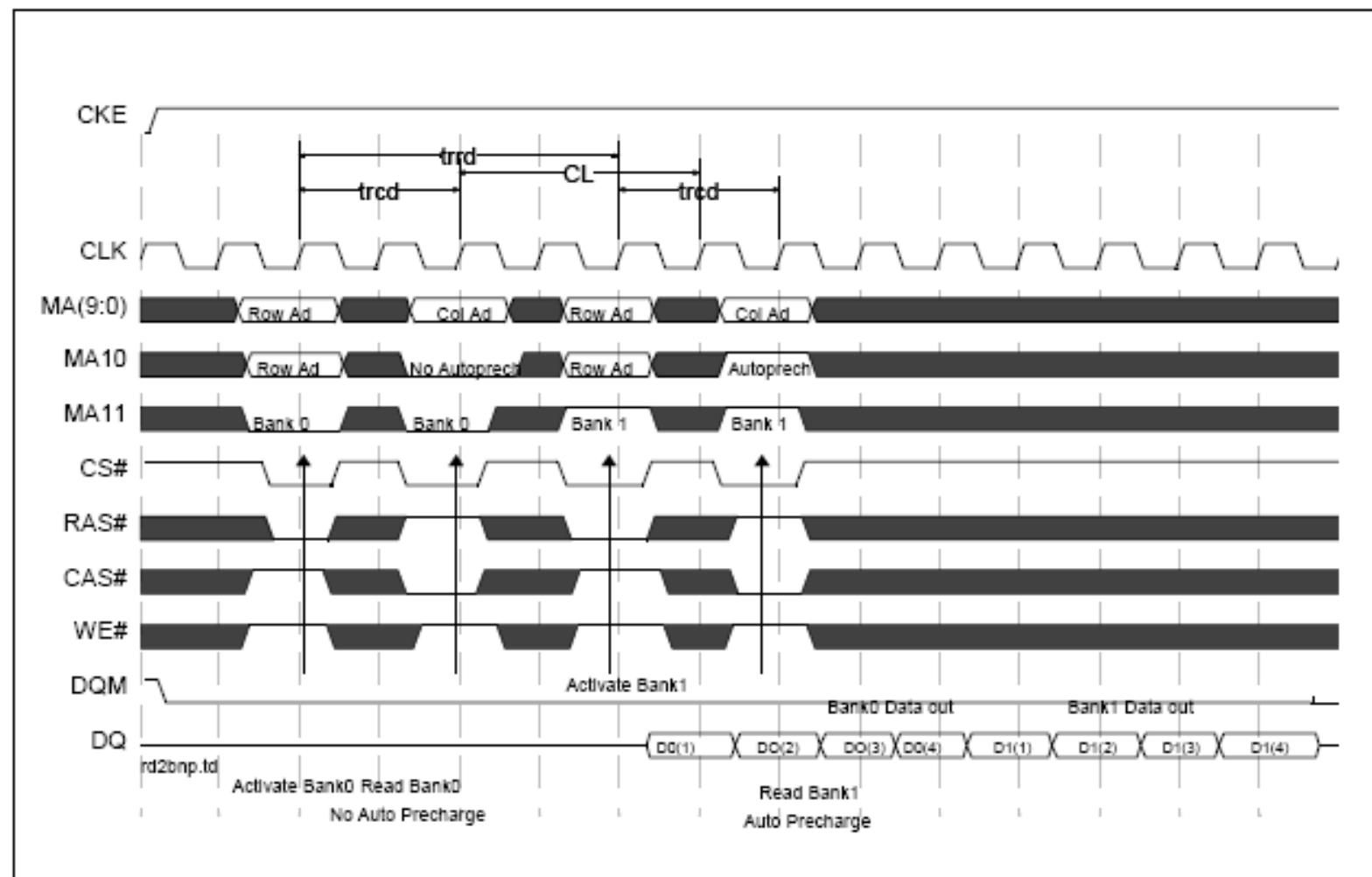
Read and Write Commands (Burst Length 4 Shown)



Read Terminated By Read



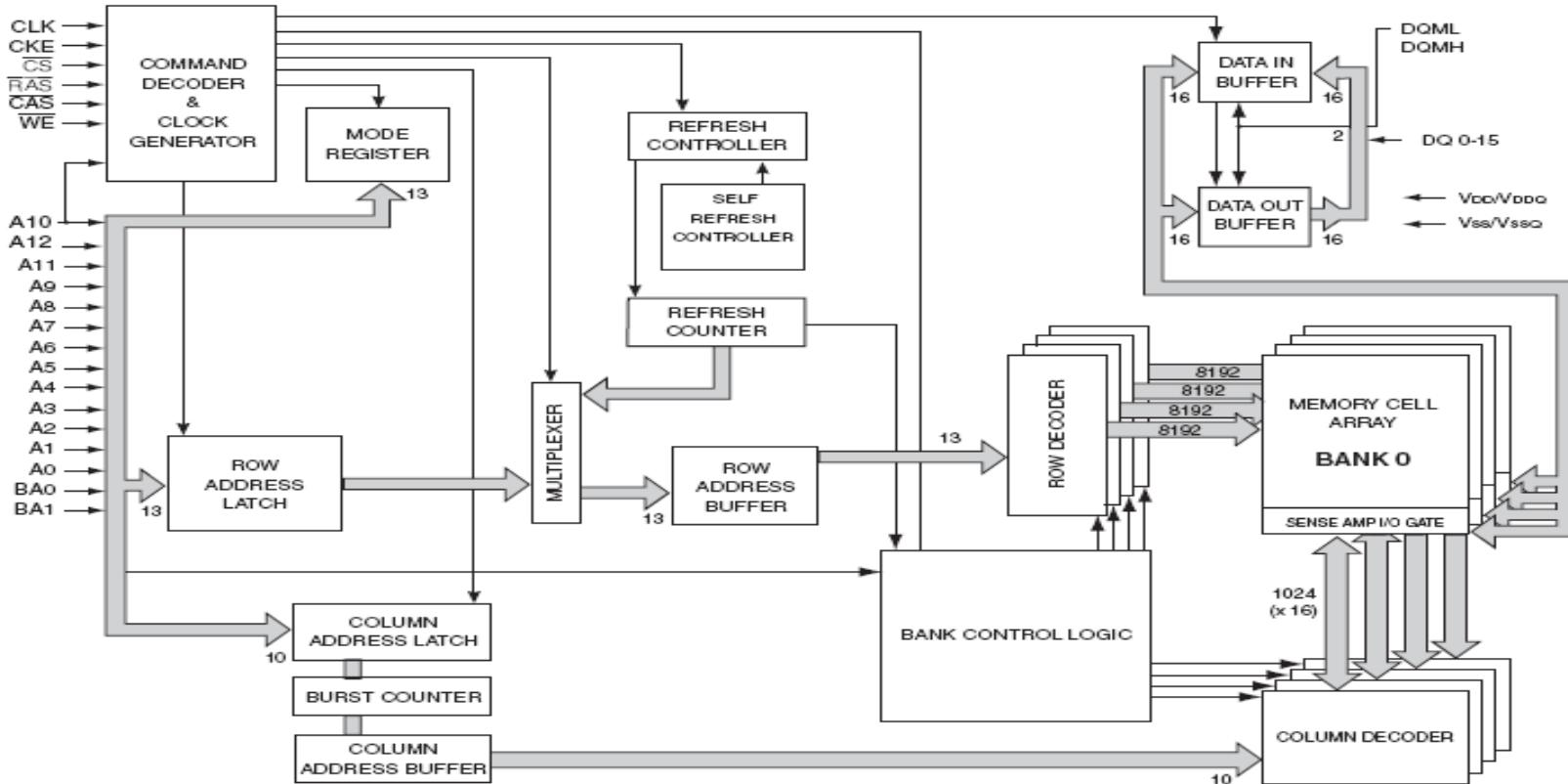
Two Bank Ping Pong Read



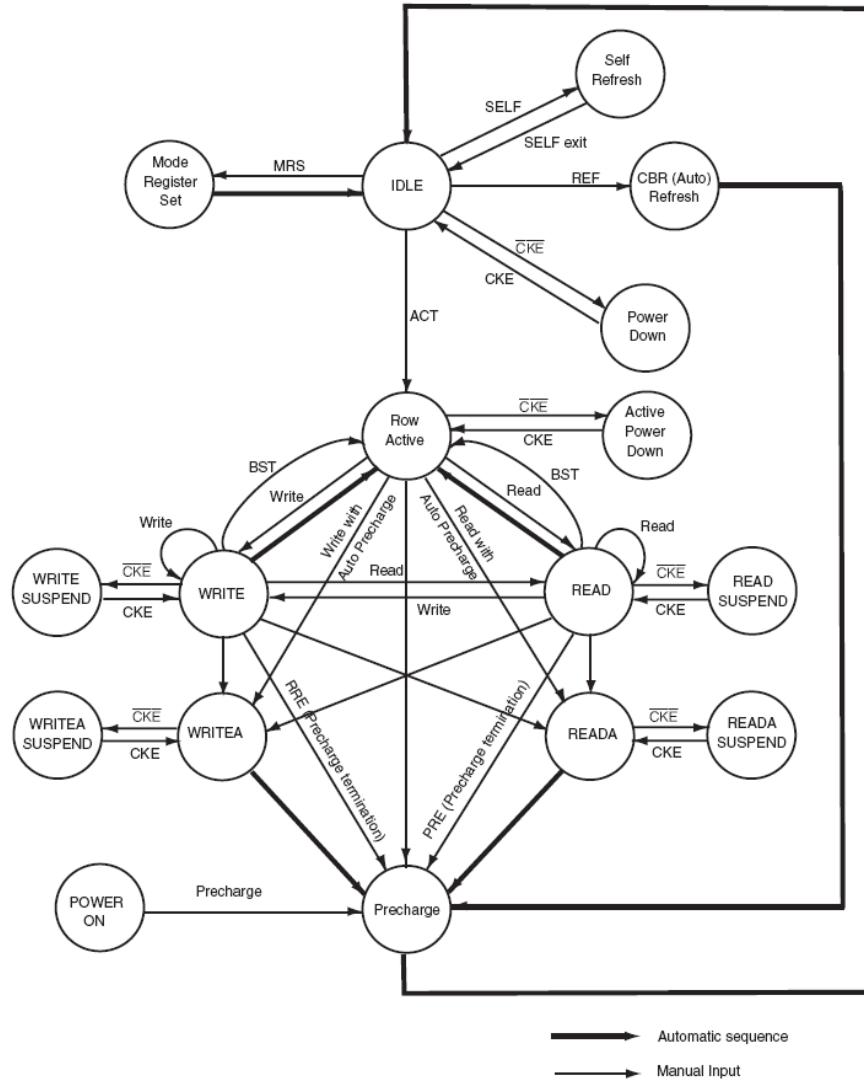
IS42S16320B

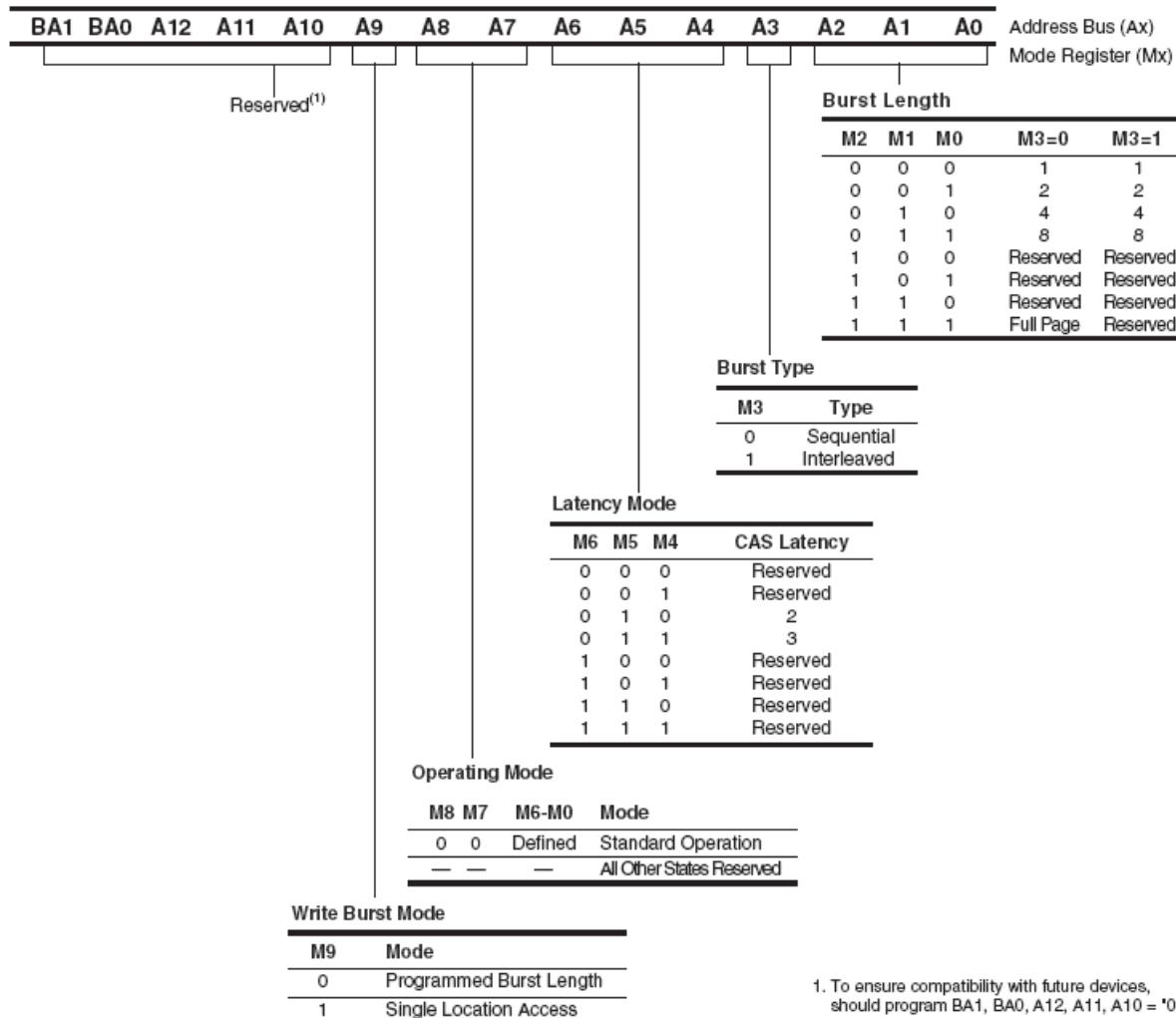
32Meg x 16
512-MBIT SYNCHRONOUS DRAM

PRELIMINARY INFORMATION
JULY 2007



STATE DIAGRAM





Timing Waveforms

Figure 1. AC Parameters for Read Timing (Burst Length =4)

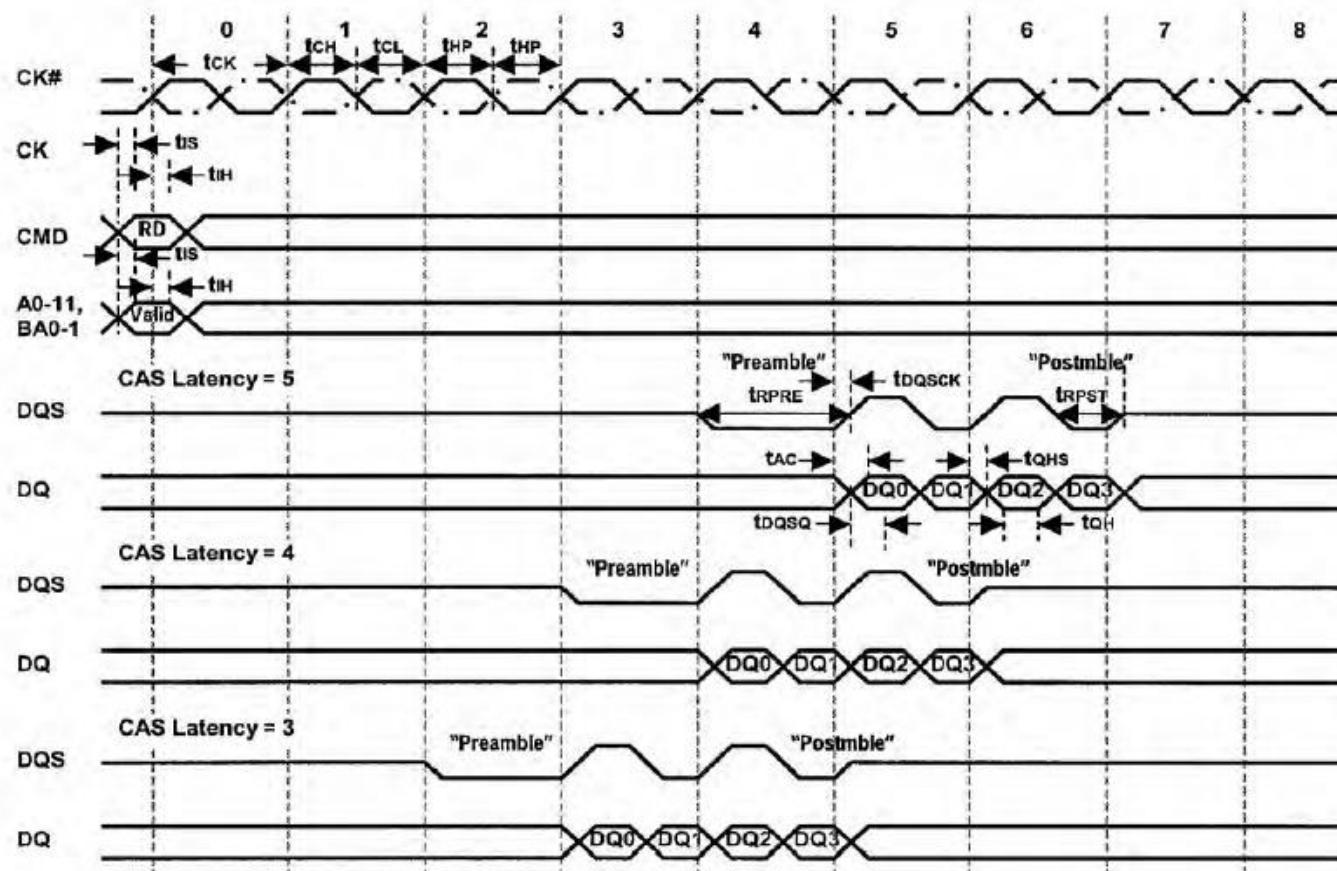
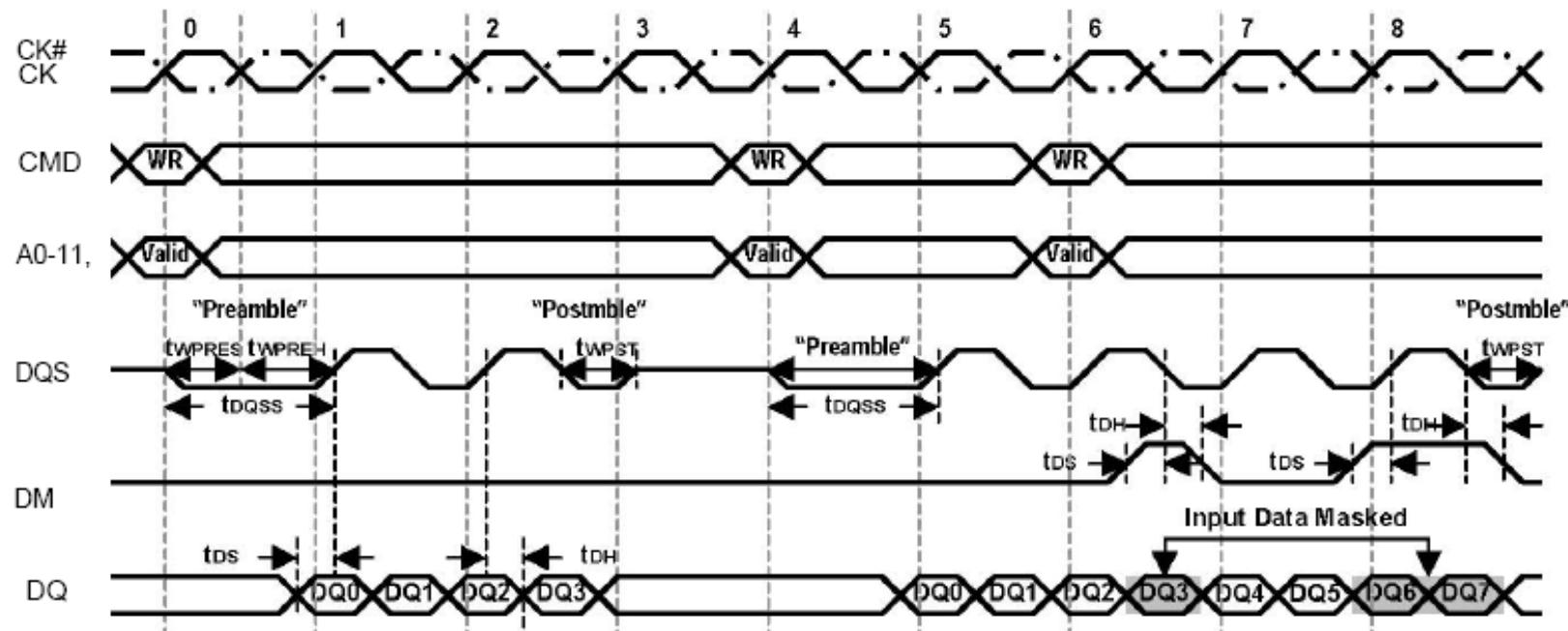


Figure 2. AC Parameters for Write Timing (Burst Length=4)



Постоянные запоминающие устройства

МПЗУ (MROM)

ППЗУ (PROM)

РПЗУ-УФ (EPROM)

ОПРПЗУ-УФ (EPROM-OTP)

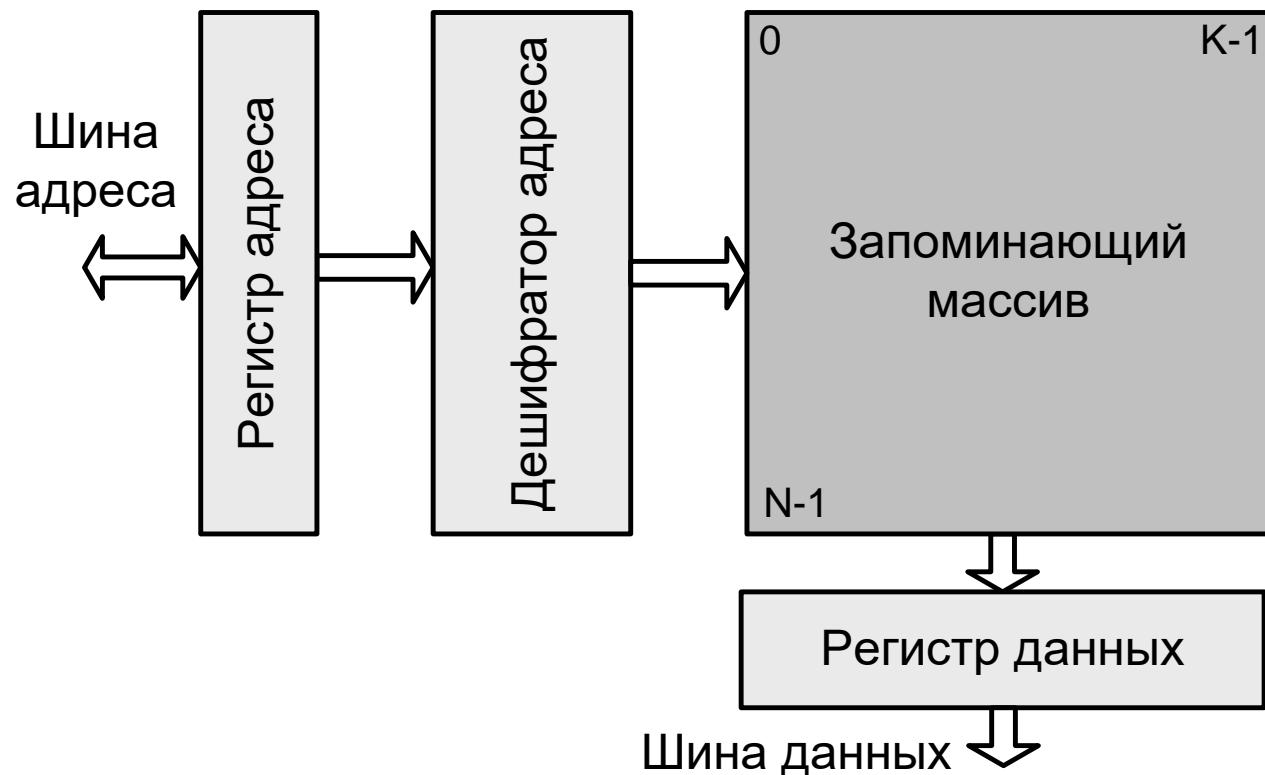
РПЗУ-ЭС (EEPROM)

FLASH

Преимущества ROM по сравнению RAM:

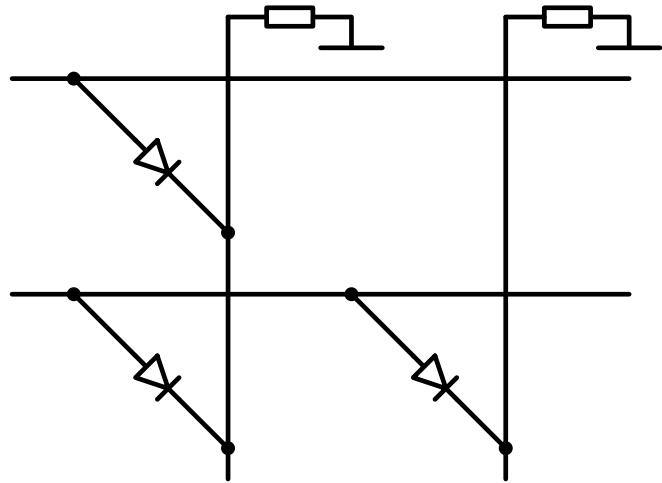
- Аппаратная простота.
- Высокая плотность размещения ЗЭ.
- Энергонезависимость.
- Большое быстродействие.

Структура ПЗУ (ROM)

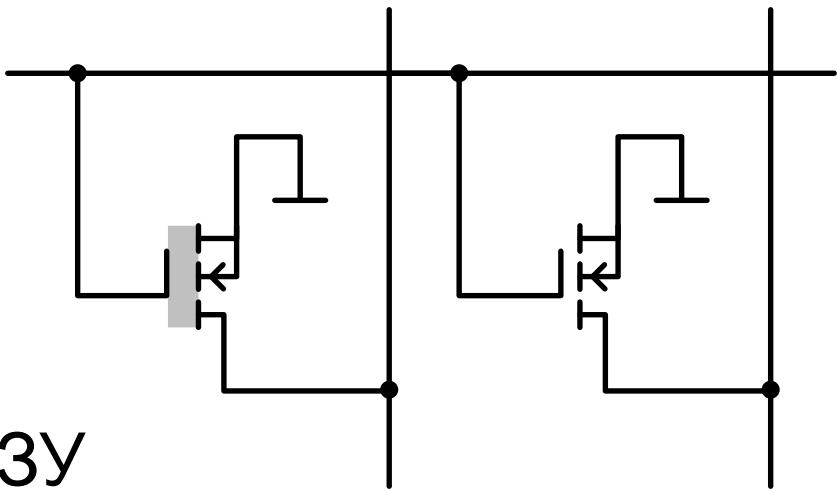


МПЗУ

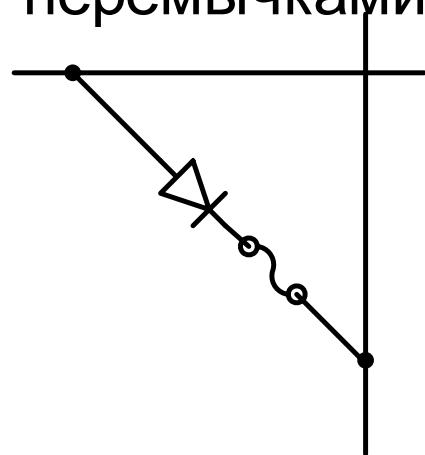
ЗЭ на диодах



ЗЭ на МОП транзисторах

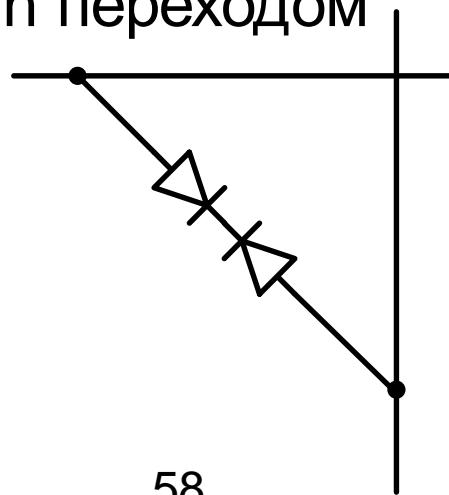


ППЗУ с плавкими
перемычками

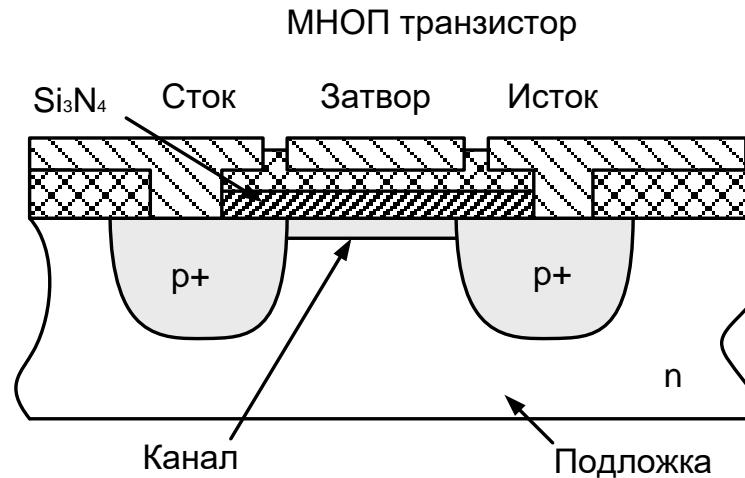
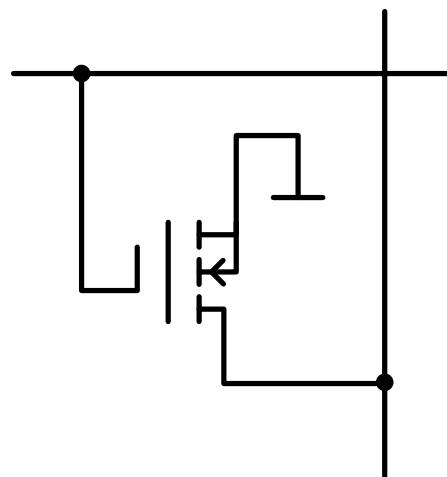


ППЗУ

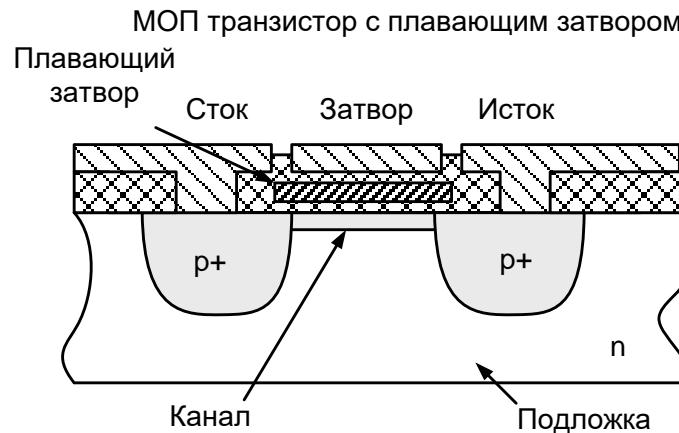
ППЗУ с переключаемым
р-n переходом



РПЗУ-УФ, ОПРРПЗУ-УФ (EPROM, EPROM-OTP)



РПЗУ-ЭС (EEPROM), FLASH





IS93C76A IS93C86A

8K-BIT/16K-BIT SERIAL ELECTRICALLY ERASABLE PROM

MAY 2007

FEATURES

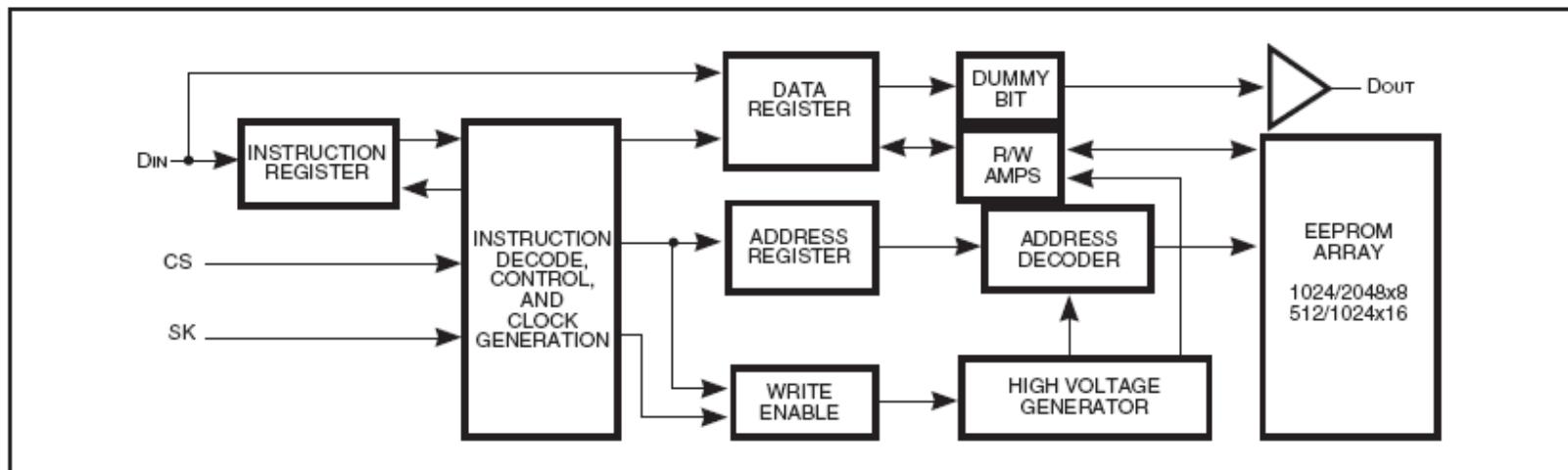
- Industry-standard Microwire Interface
 - Non-volatile data storage
 - Wide voltage operation:
 $V_{CC} = 1.8V$ to $5.5V$
 - Auto increment for efficient data dump
- User Configured Memory Organization
 - By 16-bit or by 8-bit
- Hardware and software write protection
 - Defaults to write-disabled state at power-up
 - Software instructions for write-enable/disable
- Enhanced low voltage CMOS E²PROM technology
- Versatile, easy-to-use Interface
 - Self-timed programming cycle
 - Automatic erase-before-write
 - Programming status indicator
 - Word and chip erasable
 - Chip select enables power savings
- Durable and reliable
 - 40-year data retention after 1M write cycles
 - 1 million write cycles
 - Unlimited read cycles
 - Schmitt-trigger Inputs
- Industrial and Automotive Temperature Grade
- Lead-free available

IS93C76A IS93C86A

8K-BIT/16K-BIT SERIAL ELECTRICALLY ERASABLE PROM

MAY 2007

FUNCTIONAL BLOCK DIAGRAM



IS93C76A IS93C86A

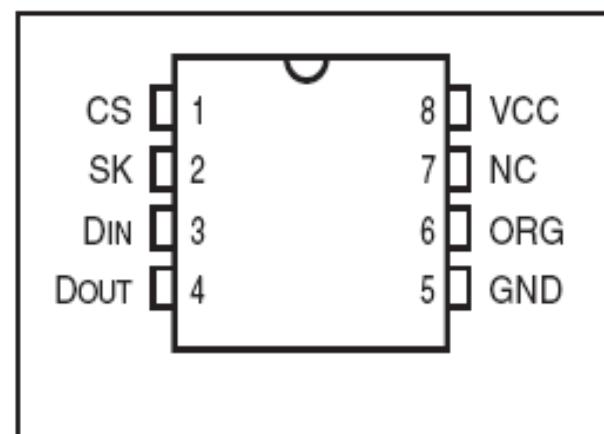
8K-BIT/16K-BIT SERIAL ELECTRICALLY ERASABLE PROM

MAY 2007

PIN DESCRIPTIONS

CS	Chip Select
SK	Serial Data Clock
D _{IN}	Serial Data Input
D _{OUT}	Serial Data Output
ORG	Organization Select
NC	Not Connected
Vcc	Power
GND	Ground

8-Pin DIP, 8-Pin TSSOP



IS93C76A IS93C86A**8K-BIT/16K-BIT SERIAL ELECTRICALLY
ERASABLE PROM**

MAY 2007

INSTRUCTION SET - IS93C86A (16kb)

Instruction ⁽²⁾	Start Bit	OP	Code	8-bit Organization (ORG = GND)		16-bit Organization (ORG = Vcc)	
				Address ⁽¹⁾	Input Data	Address ⁽¹⁾	Input Data
READ	1		10	(A10-A0)	—	(A9-A0)	—
WEN (Write Enable)	1		00	11x xxxx xxxx	—	11 xxxx xxxx	—
WRITE	1		01	(A10-A0)	(D7-Do)	(A9-A0)	(D15-Do)
WRALL (Write All Registers)	1		00	01x xxxx xxxx	(D7-Do)	01 xxxx xxxx	(D15-Do)
WDS (Write Disable)	1		00	00x xxxx xxxx	—	00 xxxx xxxx	—
ERASE	1		11	(A10-A0)	—	(A9-A0)	—
ERAL (Erase All Registers)	1		00	10x xxxx xxxx	—	10 xxxx xxxx	—

Notes:

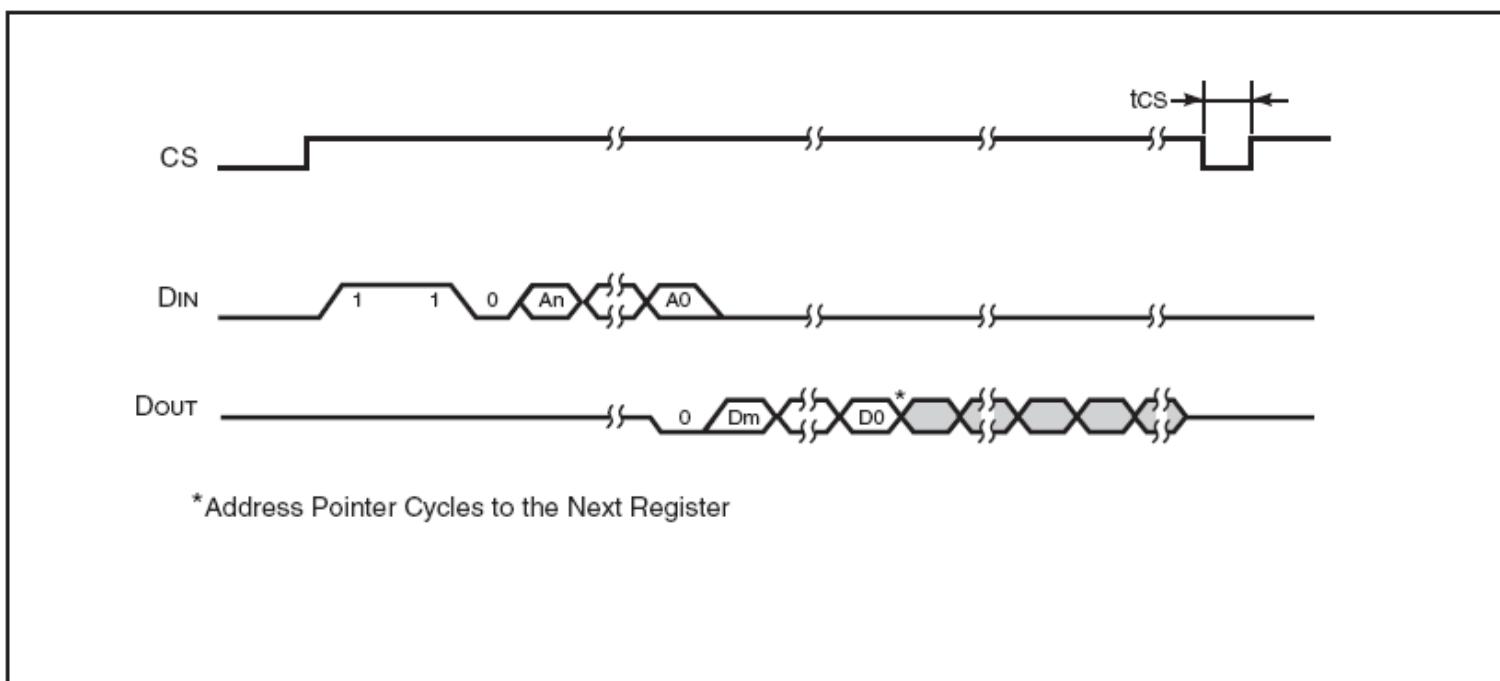
1. x = Don't care bit.
2. If the number of bits clocked-in does not match the number corresponding to a selected command, all extra trailing bits are ignored, and WRITE, WRALL, ERASE, ERAL, WEN, and WDS instructions are rejected, but READ is accepted.

IS93C76A IS93C86A

8K-BIT/16K-BIT SERIAL ELECTRICALLY ERASABLE PROM

MAY 2007

FIGURE 3. READ CYCLE TIMING





IS93C76A IS93C86A

**8K-BIT/16K-BIT SERIAL ELECTRICALLY
ERASABLE PROM**

MAY 2007



Features

- Single 4.5V - 5.5V Supply
- Serial Interface Architecture
- Page Program Operation
 - Single Cycle Reprogram (Erase and Program)
 - 1024 Pages (264 Bytes/Page) Main Memory
- Two 264-Byte SRAM Data Buffers – Allows Receiving of Data while Reprogramming of Nonvolatile Memory
- Internal Program and Control Timer
- Fast Page Program Time – 7 ms Typical
- 80 μ s Typical Page to Buffer Transfer Time
- Low Power Dissipation
 - 15 mA Active Read Current Typical
 - 15 μ A CMOS Standby Current Typical
- 10 MHz Max Clock Frequency
- Hardware Data Protection Feature
- Serial Peripheral Interface (SPI) Compatible – Modes 0 and 3
- CMOS and TTL Compatible Inputs and Outputs
- Commercial and Industrial Temperature Ranges

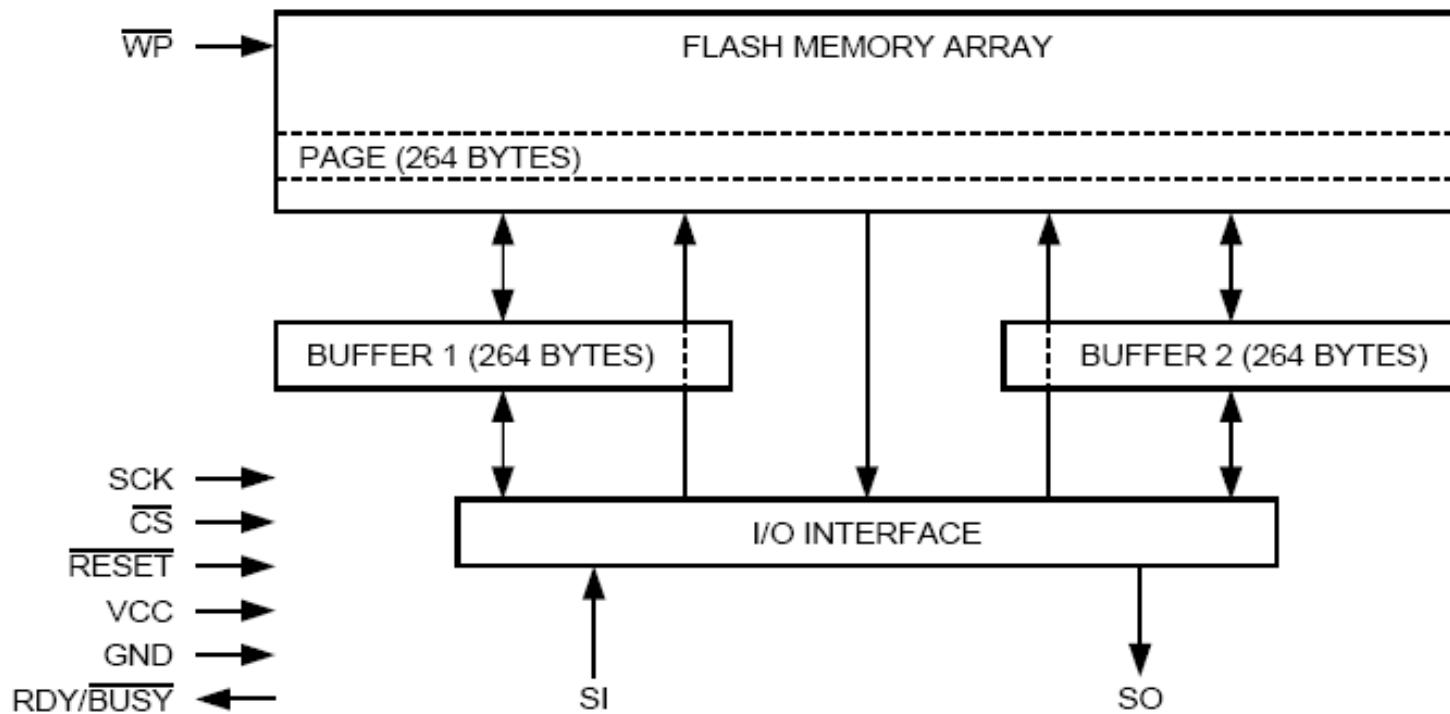
2-Megabit
5-volt Only
Serial
DataFlash®

AT45D021

Pin Configurations

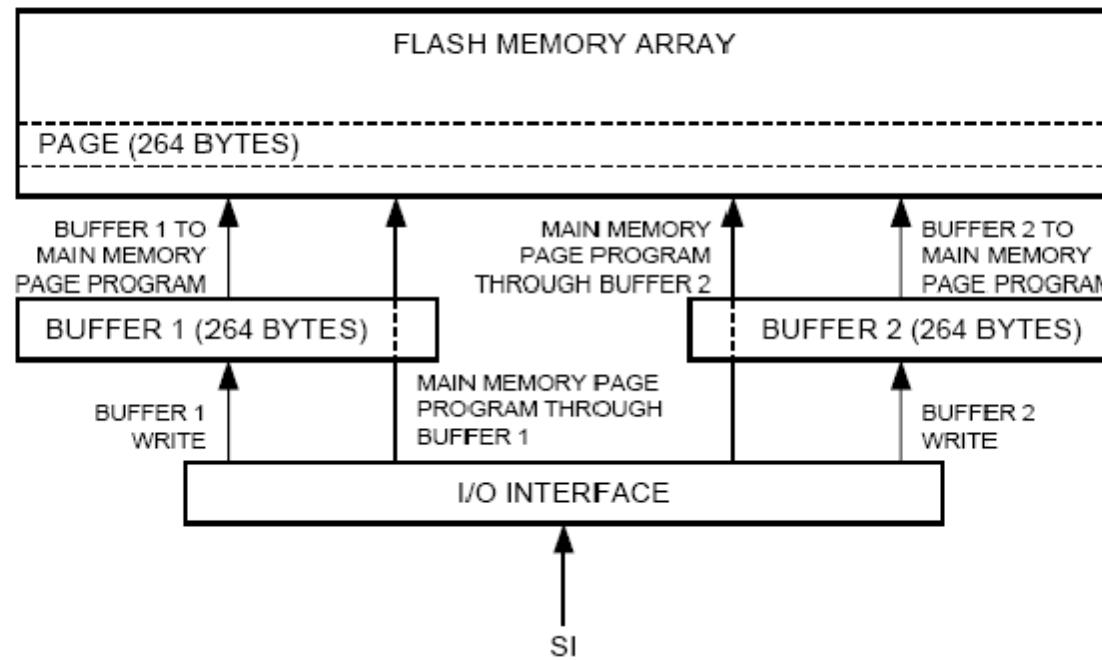
Pin Name	Function
CS	Chip Select
SCK	Serial Clock
SI	Serial Input
SO	Serial Output
WP	Hardware Page Write Protect Pin
RESET	Chip Reset
RDY/BUSY	Ready/Busy

Block Diagram

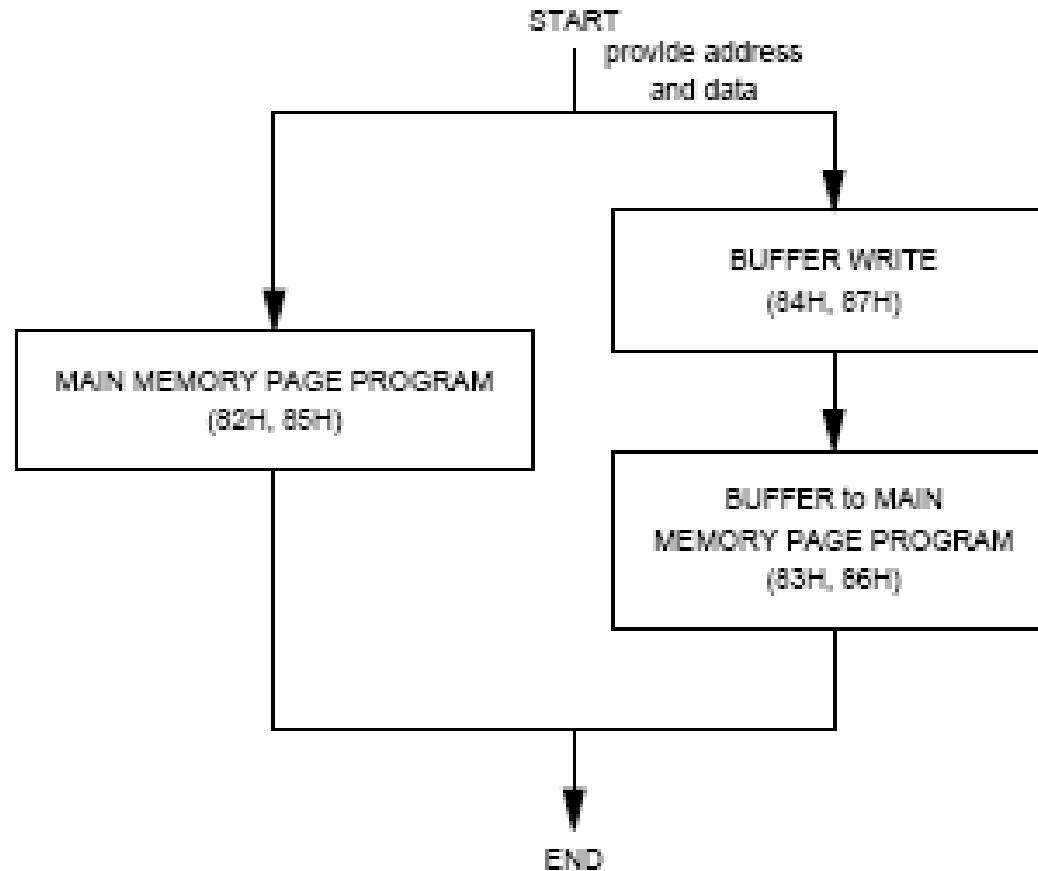


Write Operations

The following block diagram and waveforms illustrate the various write sequences available.



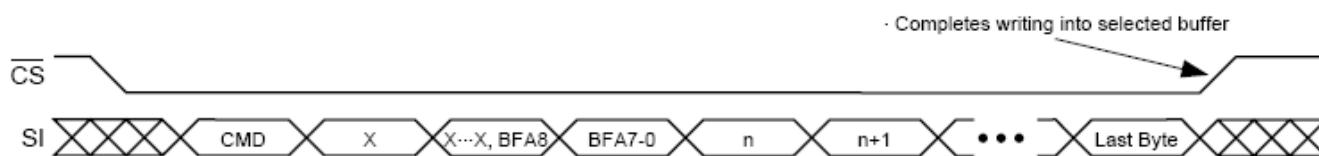
Algorithm for Programming or Reprogramming of the Entire Array Sequentially



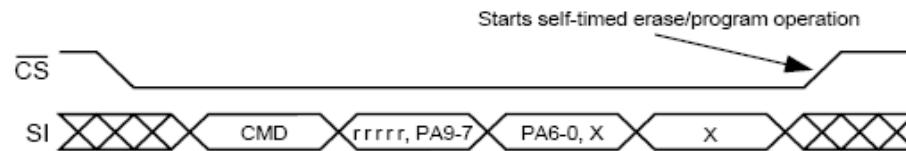
Main Memory Page Program through Buffers



Buffer Write



Buffer to Main Memory Page Program (Data from Buffer Programmed into Flash Page)

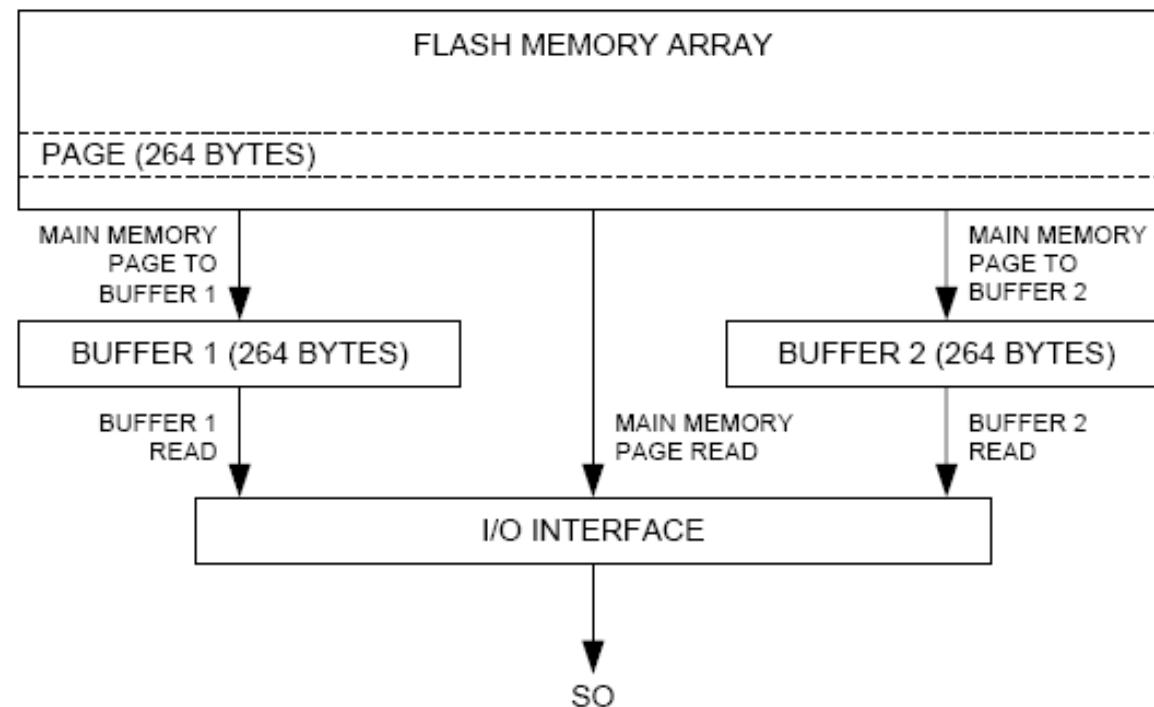


Each transition represents
8 bits and 8 clock cycles

n = 1st byte read
n+1 = 2nd byte read

Read Operations

The following block diagram and waveforms illustrate the various read sequences available.



Main Memory Page Read**Main Memory Page to Buffer Transfer (Data from Flash Page Read into Buffer)****Buffer Read**

Each transition represents
8 bits and 8 clock cycles

n = 1st byte written
n+1 = 2nd byte written

Методы повышение надежности ЗУ

Контроль по четности/нечетности

$$P_{\text{ч}} = d_0 \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_6 \oplus d_7, P_{\text{н}} = \bar{P}_{\text{ч}}$$

\oplus - операция сложения по модулю

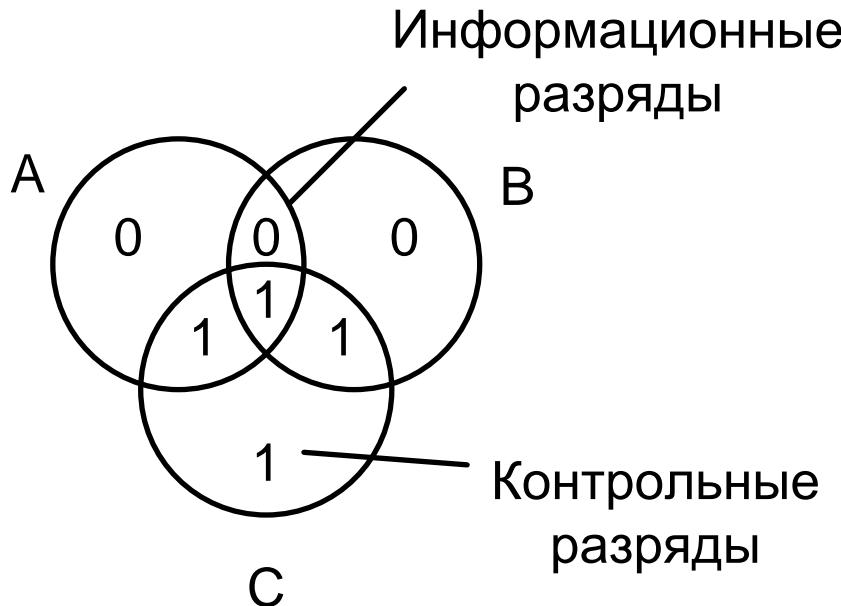
Пример: $D = 10010100$, количество единиц = 3,

$$P_{\text{ч}} = d_0 \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_6 \oplus d_7 = 1, P_{\text{н}} = \bar{P}_{\text{ч}} = 0$$

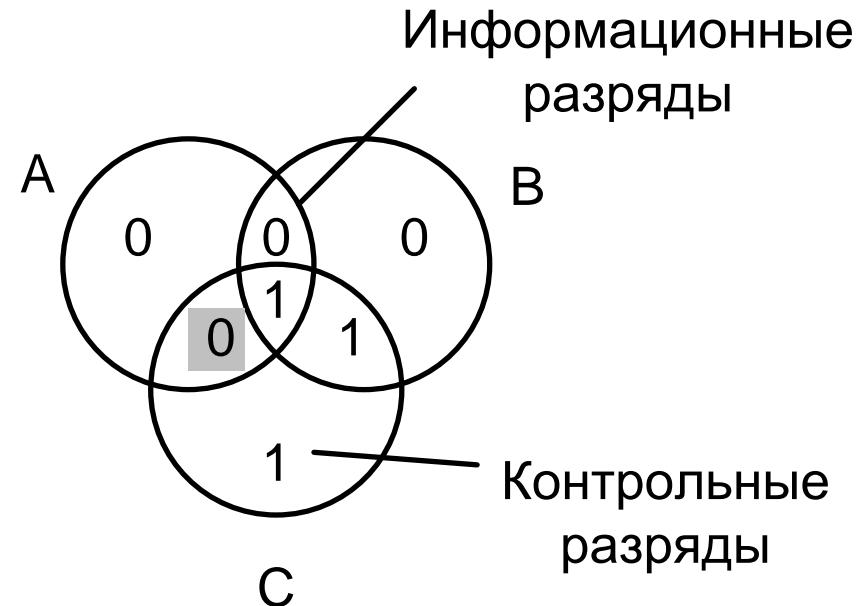
При чтении новое P' сравнивается P и если $P' \oplus P = 1$,
то обнаружена ошибка.

Код Хэмминга

Исходные данные



Ошибочные данные



$P'_A=1, P'_B=0, P'_C=0 \Rightarrow$ Нарушен информационный
бит $A \cap C / B$

Результат проверок по коду Хэмминга - синдром:

$$S = \{p_1 \oplus p'_1, p_2 \oplus p'_2, p_3 \oplus p'_3\}$$

Код Хэмминга позволяет обнаружить и исправить единичную ошибку и обнаружить двойную.

- Если $S = 0$, то ошибок не обнаружено.
- Если в синдроме одна единица, то ошибка в одном корректирующем разряде (не исправляется).
- Если в синдроме несколько единиц, то он указывает на ошибочный информационный разряд.
- При добавлении общего контрольного разряда ($P = d_0 \oplus d_1 \oplus d_2 \oplus d_3 \oplus p_0 \oplus p_1 \oplus p_2$) можно обнаружить двойную ошибку (не исправляется)

Пример для 4-х разрядных информационных слов

Корректирующие разряды размещены в позициях 2^i и контролируют разряды с двоичным номером, содержащим 2^i .

P	d3	d2	d1	p2	d0	p1	p0
8	7	6	5	4	3	2	1

Исходное слово

0	0	1	0	1	1	0	1
8	7	6	5	4	3	2	1

Ошибкающее слово

0	0	1	0	1	0	0	1
8	7	6	5	4	3	2	1

Синдром: $S = \{p_2 \oplus p'_2, p_1 \oplus p'_1, p_0 \oplus p'_0\} = 011_2 = 3_{10}$

$$p_0 = d_0 \oplus d_1 \oplus d_3,$$

$$p_1 = d_0 \oplus d_2 \oplus d_3,$$

$$p_2 = d_1 \oplus d_2 \oplus d_3$$

$$p_0 = d_0 \oplus d_1 \oplus d_3 = 1,$$

$$p_1 = d_0 \oplus d_2 \oplus d_3 = 0,$$

$$p_2 = d_1 \oplus d_2 \oplus d_3 = 1$$

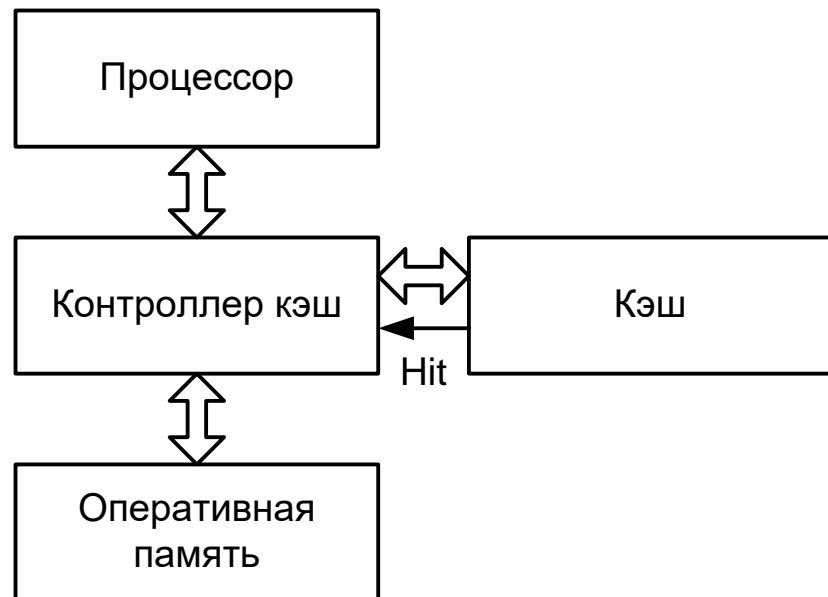
$$p'_0 = d_0 \oplus d_1 \oplus d_3 = 0,$$

$$p'_1 = d_0 \oplus d_2 \oplus d_3 = 1,$$

$$p'_2 = d_1 \oplus d_2 \oplus d_3 = 1$$

Принципы построения кэш-памяти

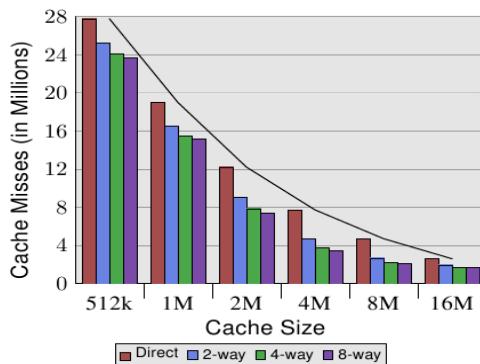
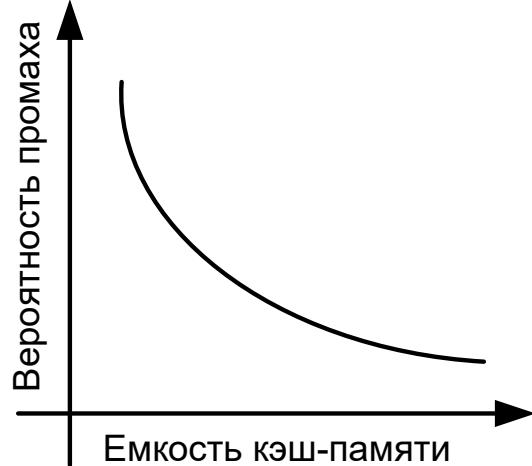
Кэш-память – ассоциативное ЗУ, позволяющее сгладить разрыв в производительности процессора и оперативной памяти. Выборка из кэш-памяти осуществляется по физическому адресу ОП.



Эффективность кэш-памяти зависит от:

- Емкости кэш-памяти.
- Размера строки.
- Способа отображения ОП в кэш.
- Алгоритма замещения информации в кэш.
- Алгоритма согласования ОП и кэш.
- Числа уровней кэш.

Емкость кэш-памяти



Размер линейки

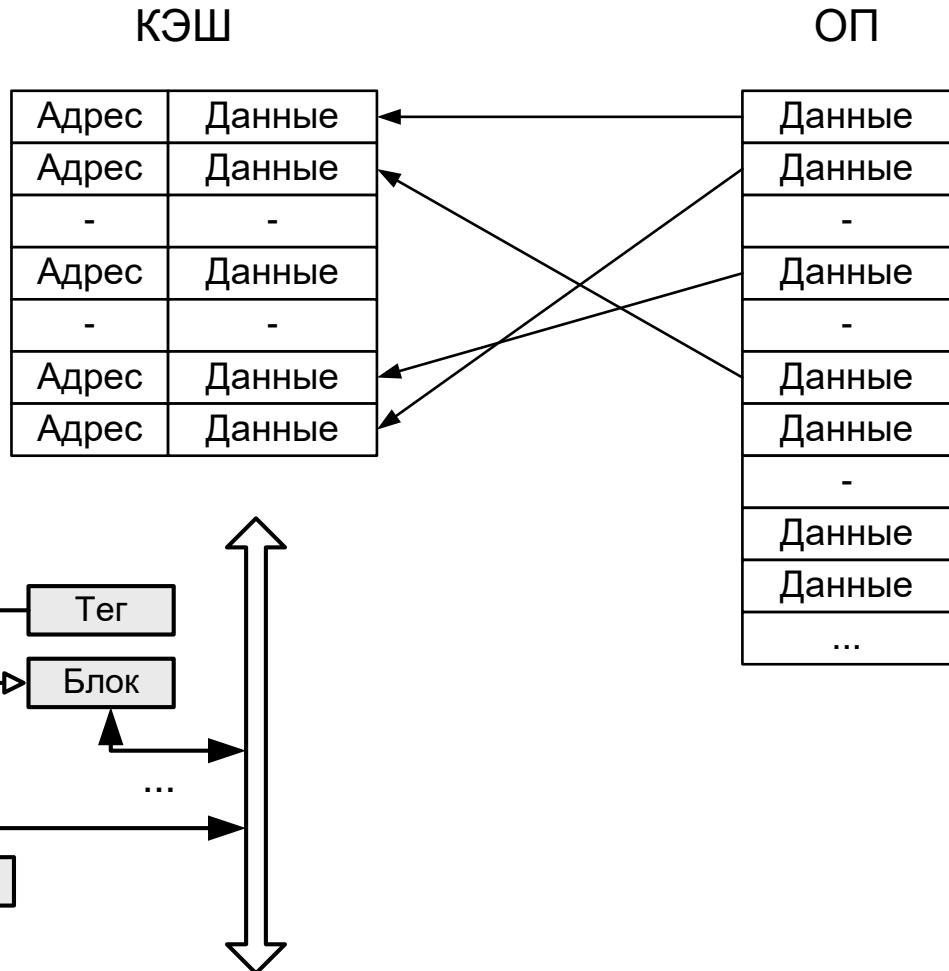
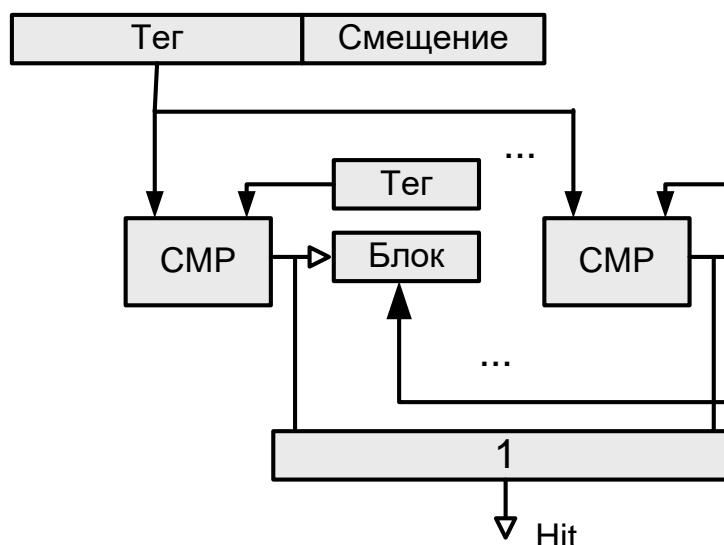


Способы отображения ОП в кэш:

- Произвольная загрузка.
- Прямое размещение.
- Наборно-ассоциативный способ отображения.

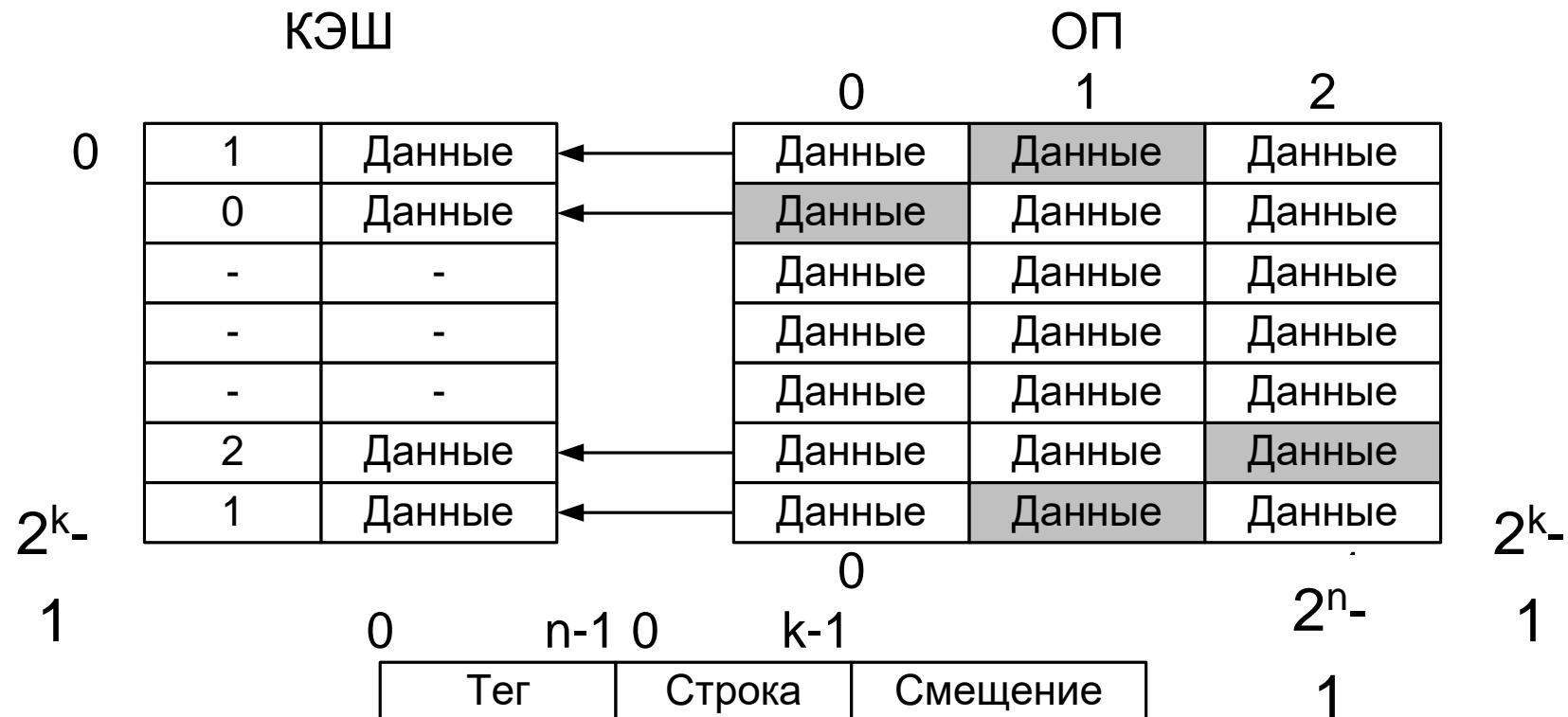
Произвольная загрузка (Fully associated cache memory, FACM).

Адрес строки FACM
определяется из условия
формирования наиболее
представительной выборки

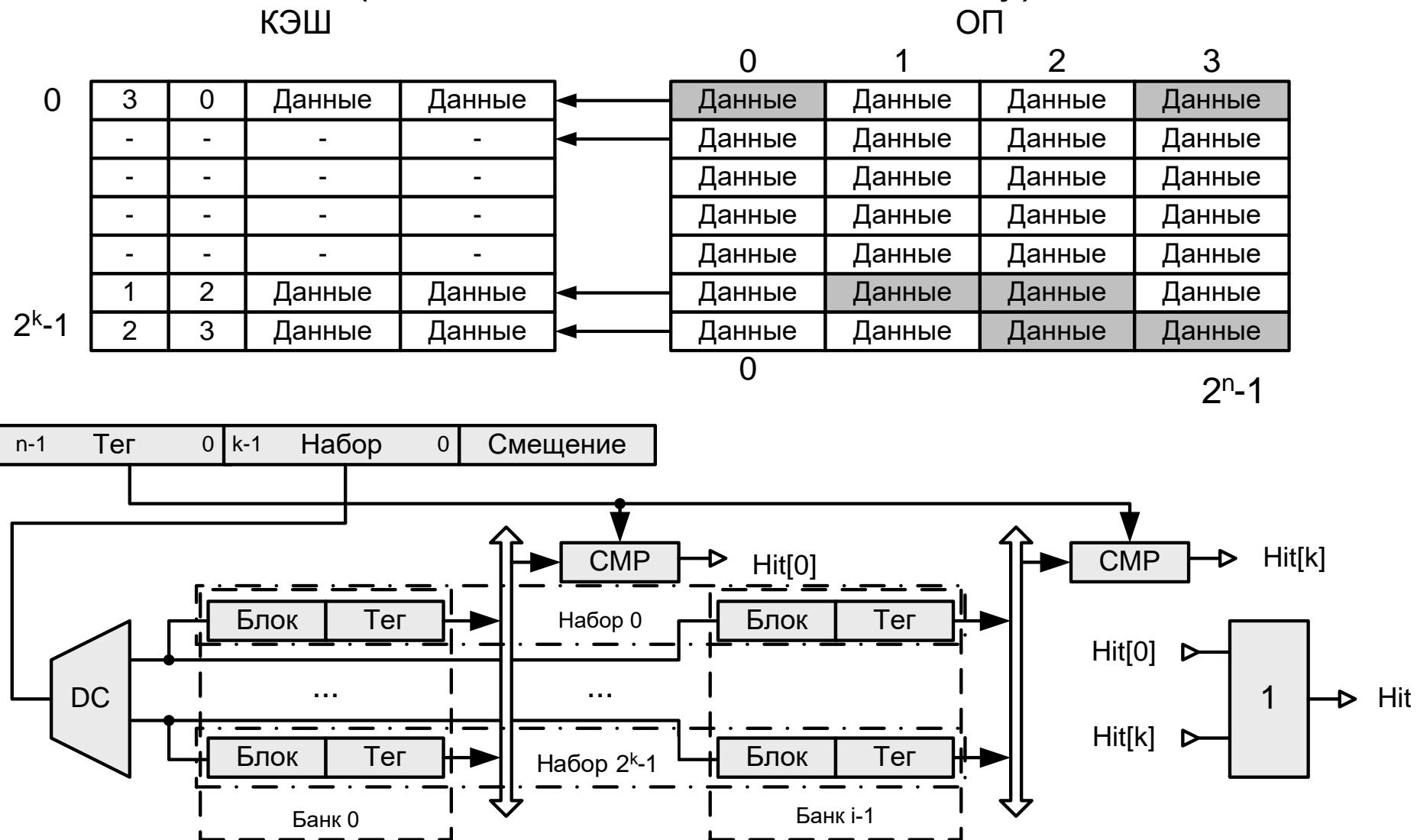


Прямое размещение.

Адрес строки однозначно определяется по тегу ($i = t \bmod k$).



Наборно-ассоциативная кэш-память (Set associated cache memory)



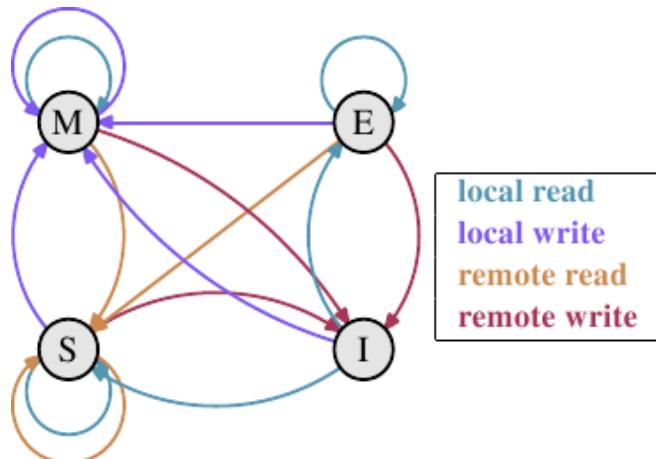
Алгоритмы замещения

- Замещение немодифицированных данных.
- Рандомизированный алгоритм.
- Замещение наименее используемого (Least Recently Used, LRU)

Согласование ОП и кэш

- Метод сквозной записи (Write True).
- Метод сквозной записи с буферизацией (Write Combining).
- Метод обратной записи (Write Back).

Протокол MESI

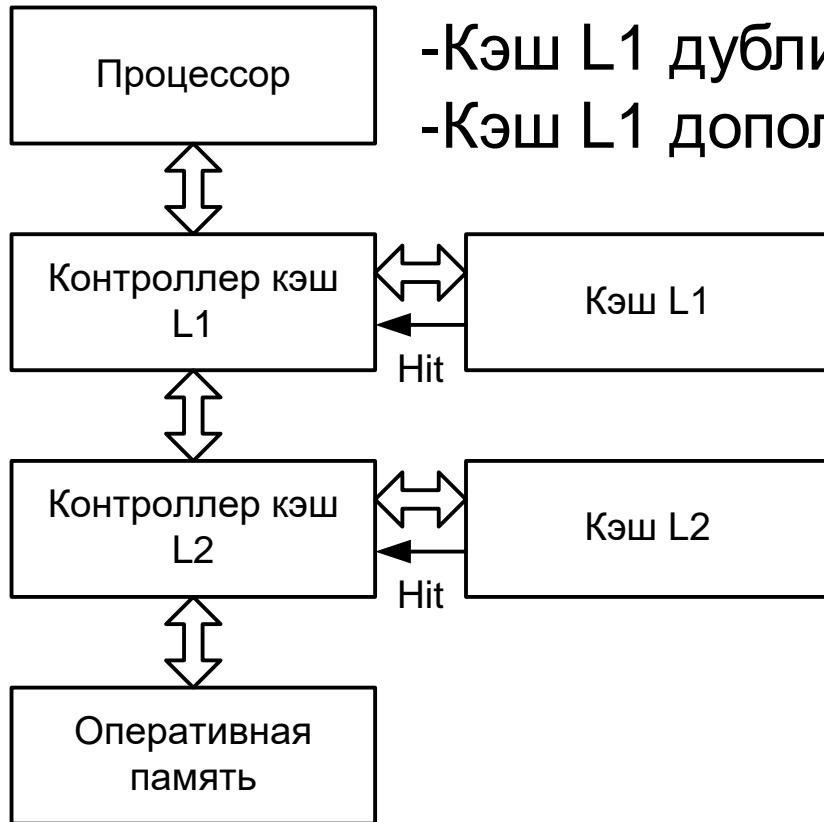


Modified
Exclusive
Shared
Invalid

- Признак несогласованных данных.
- Признак согласованных данных.
- Признак согласованных данных в ВС.
- Признак отсутствия данных.

* - <http://lwn.net/Articles/252125/>

Разделение кэш-памяти

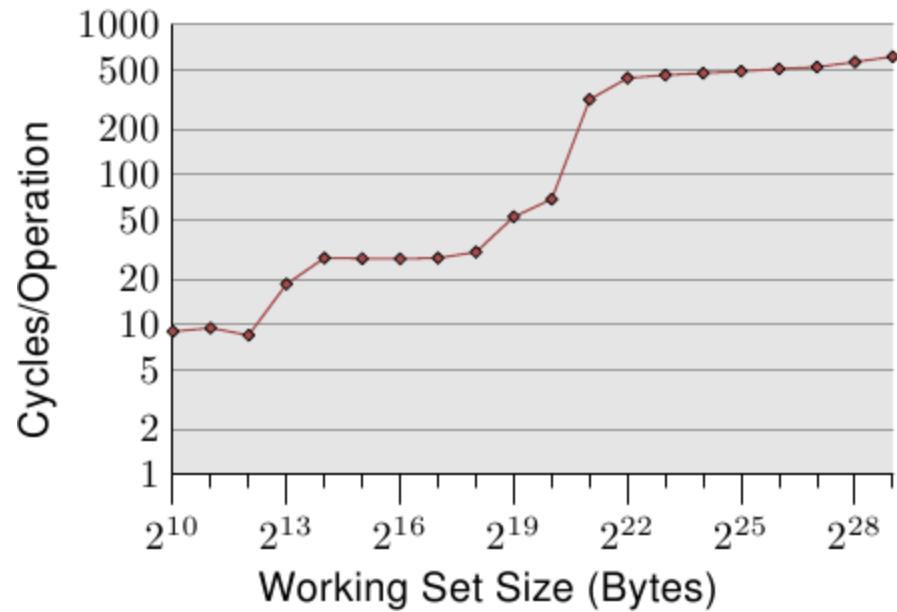


- Кэш L1 дублирует L2 (inclusive).
- Кэш L1 дополняет L2 (exclusive).

Доступ к массивам данным по
случайным адресам

L1D — 2^{13} байт

L2D — 2^{21} байт



Виртуальная память

Механизм виртуализации адресного пространства позволяет:

- Увеличить объем адресуемой памяти.
- Использовать физическую память различного объема.
- Возложить на аппаратную составляющую механизмы доступа к ВЗУ
- Сгладить разрыв в производительности ОП и ВЗУ.
- Ускоряет доступ к данным по последовательным адресам.
- Способствует реализации защиты памяти.

Виртуальные системы строятся по трем принципам:

- Системы с блоками различного размера (сегментная организация).
- Системы с блоками одинакового размера (страничная организация).
- Смешанные системы (сегментно-страничная организация).

Страницная организация

Программа отображается в память равными блоками – страницами. Преобразование логического адреса в физический осуществляется с помощью таблицы страниц.

Преобразование логического адреса в физический реализуется в устройстве управления памятью (Memory Manage Unit), который определяет, находится ли страница в физической памяти (попадение).

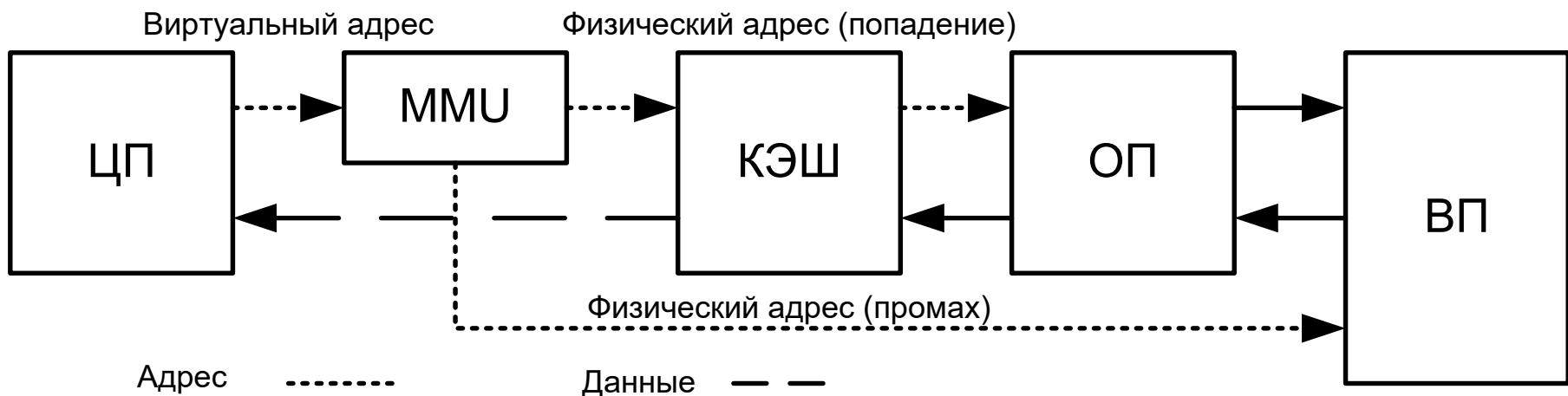
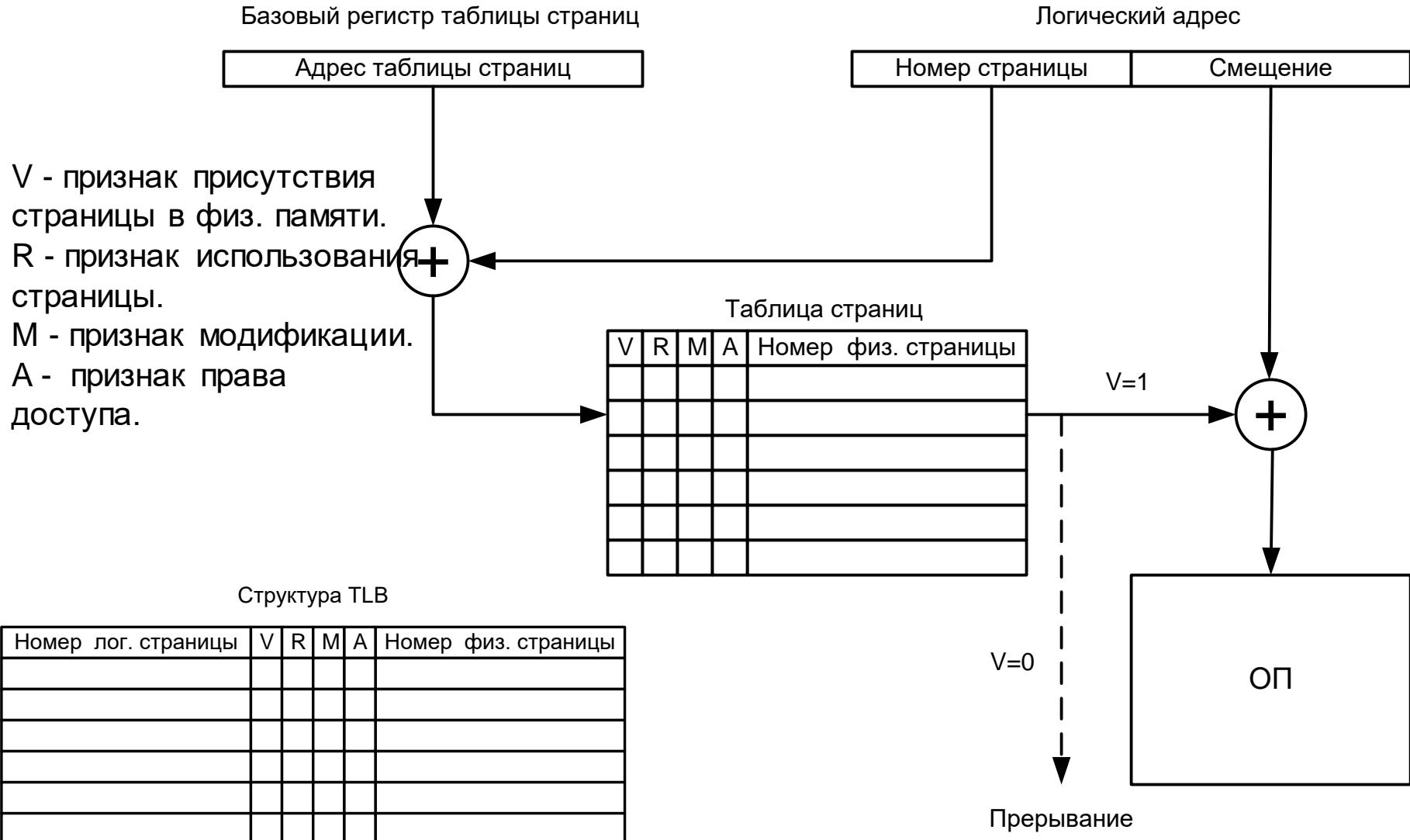
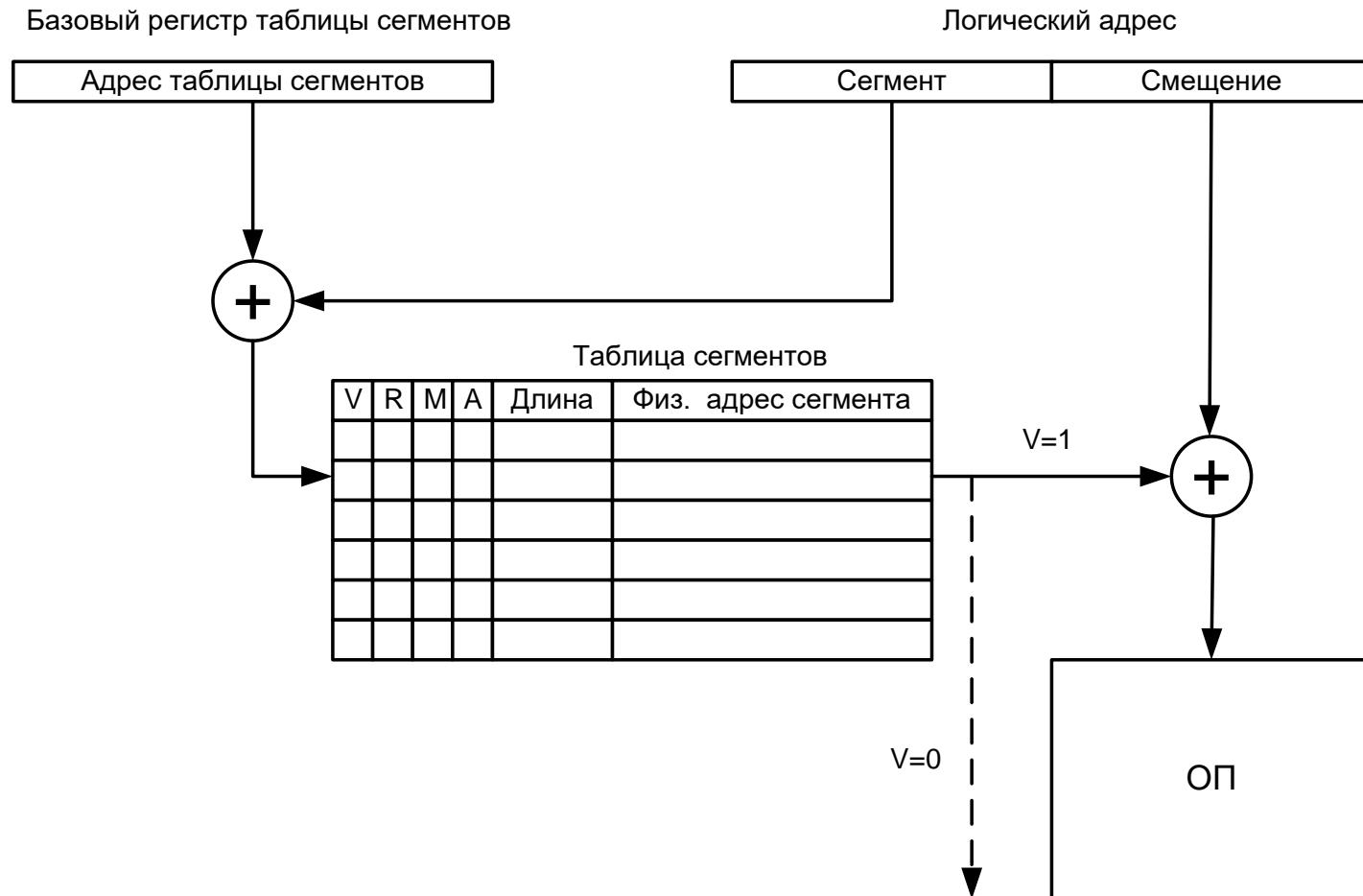


Схема страничного преобразования



Сегментная организация

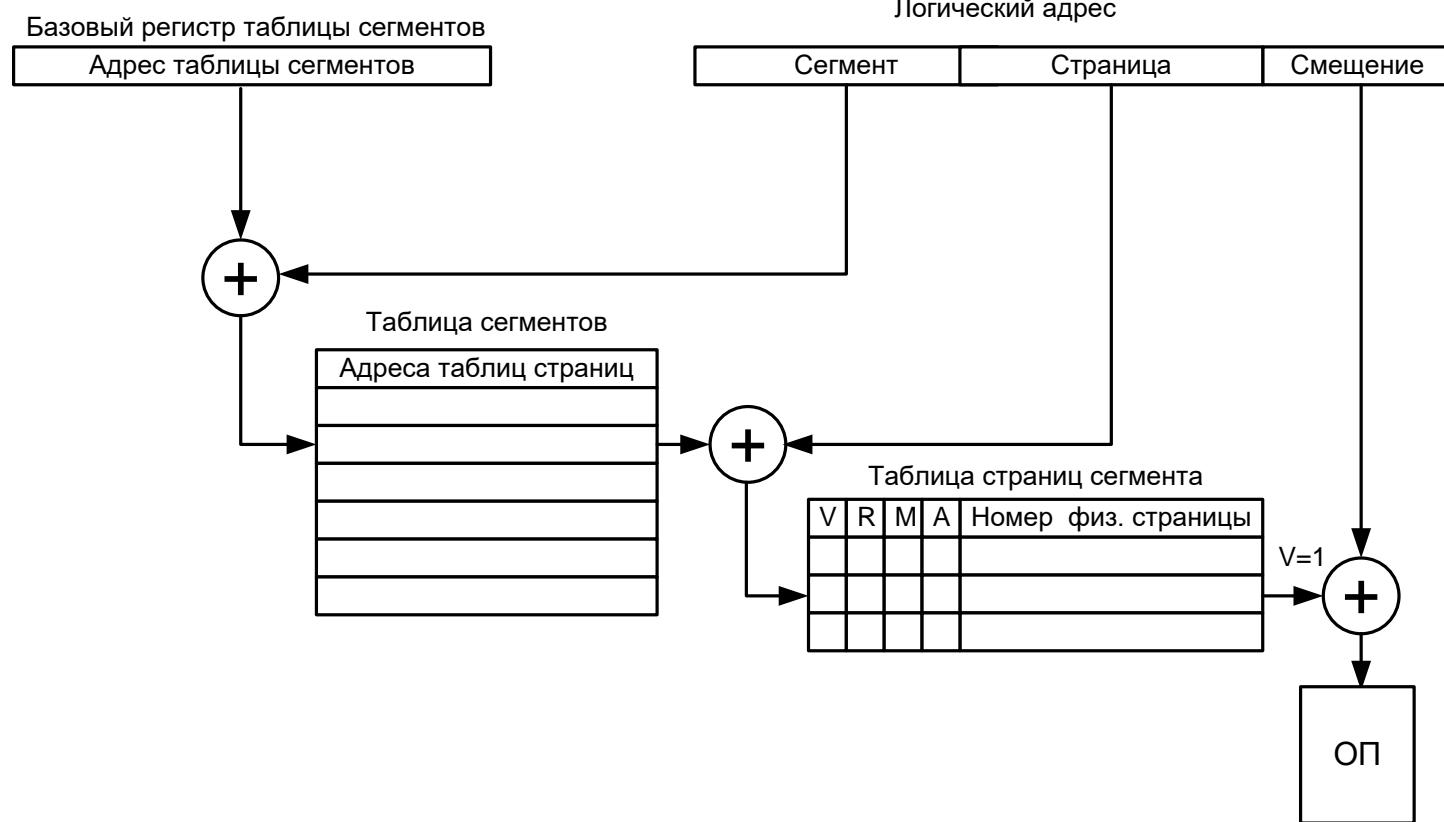
Программа отображается в память блоками различного размера – сегментами. Преобразование логического адреса в физический осуществляется с помощью таблицы сегментов.



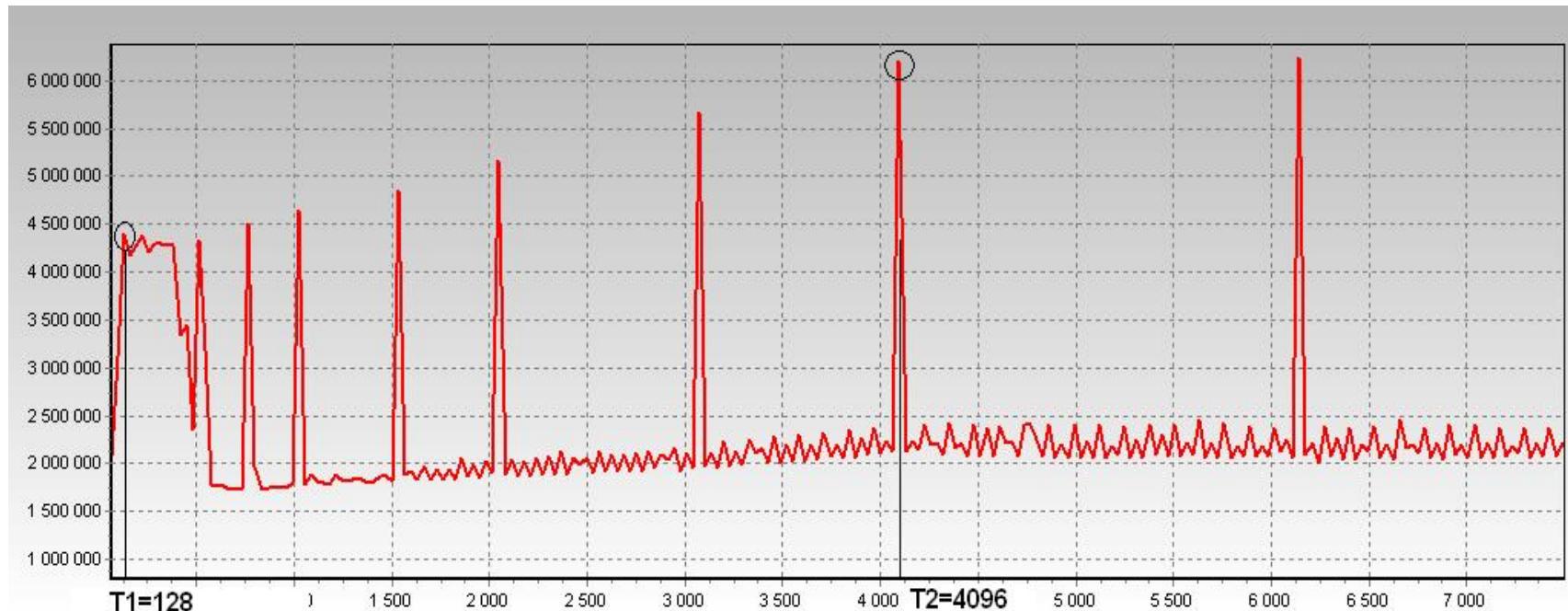
Сегментно-страничная организация памяти

Программа отображается в память блоками различного размера – сегментами, каждый из которых целое число страниц.

Преобразование логического адреса в физический осуществляется с помощью таблицы сегментов и таблицы страниц сегмента.



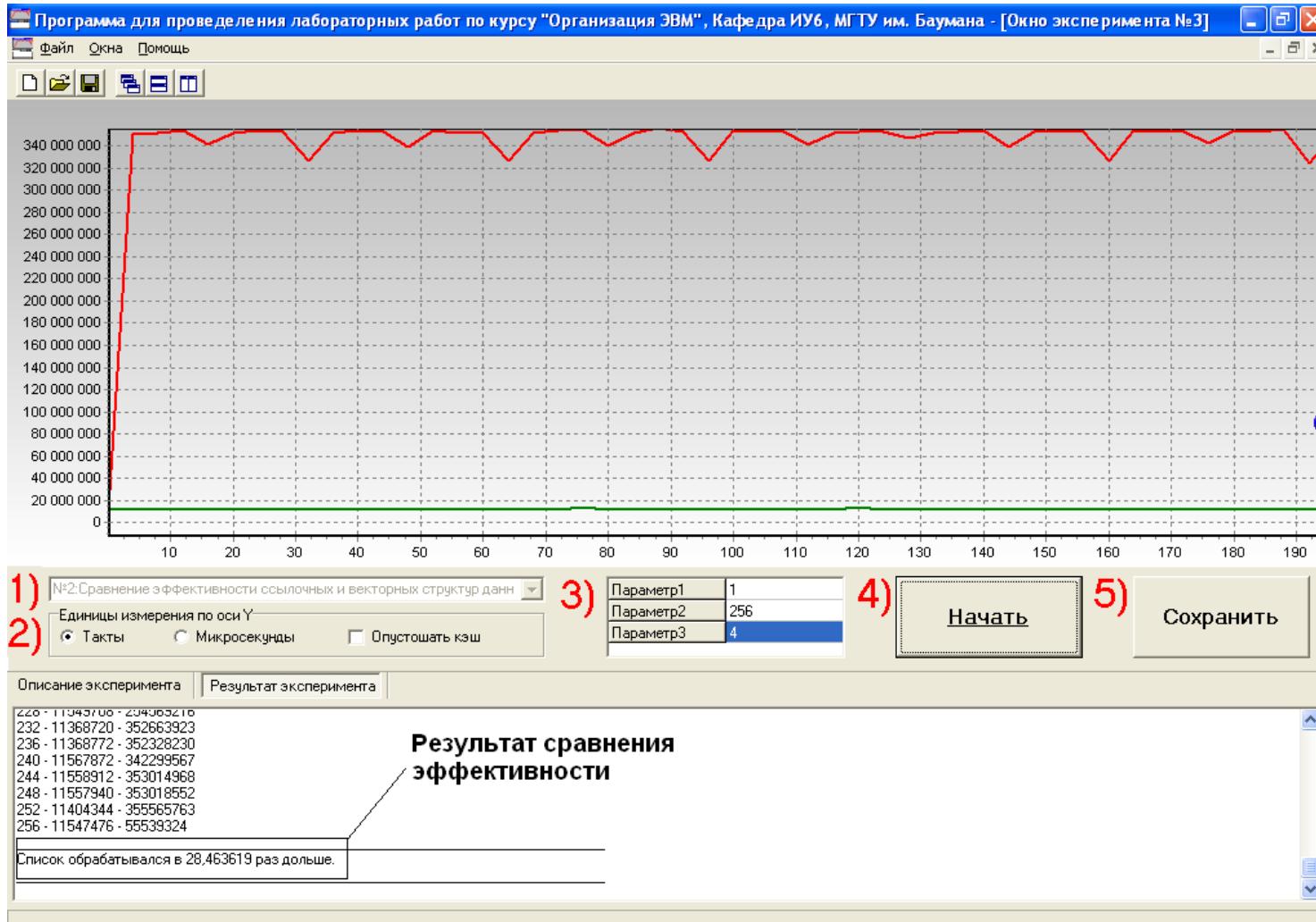
Исследование расслоения динамической памяти.



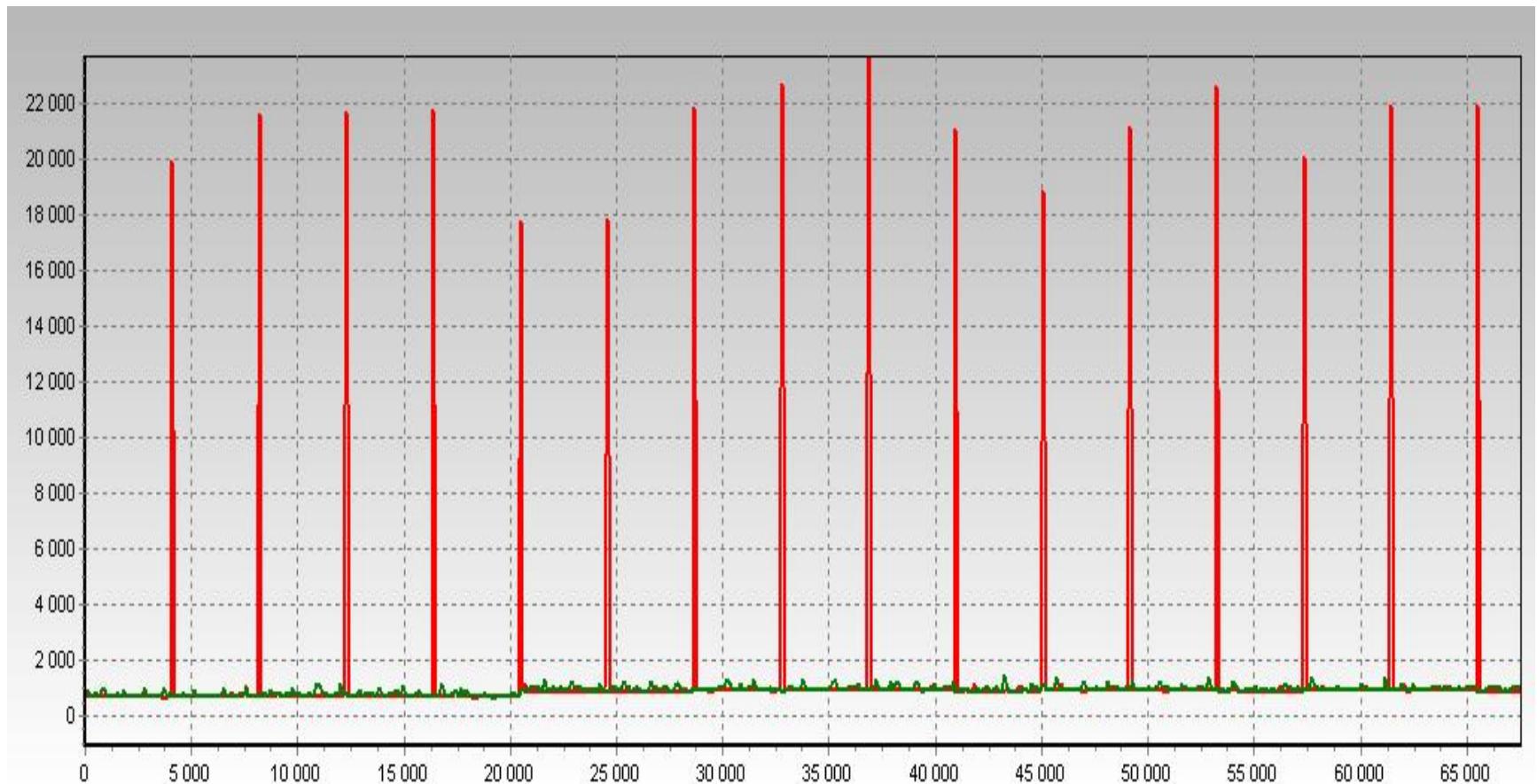
Код профилируемой программы на языке С.

```
// ВЫДЕЛЕНИЕ ПАМЯТИ
p = (int*)_malloc64(Param_[3]); // АДРЕС КРАТЕН 64
for (int pg_size = Param_[2]; pg_size <= Param_[1]; pg_size += Param_[2])
{
    Start_Count(); // Начало замера времени
    volatile int x = 0;
    for (int b = 0; b < pg_size; b += Param_[2])
        for (int a = b; a < Param_[3]; a += pg_size)
            x += *(int *) (int(p) + a);
    Finish_Count(); // Конец замера времени
}
```

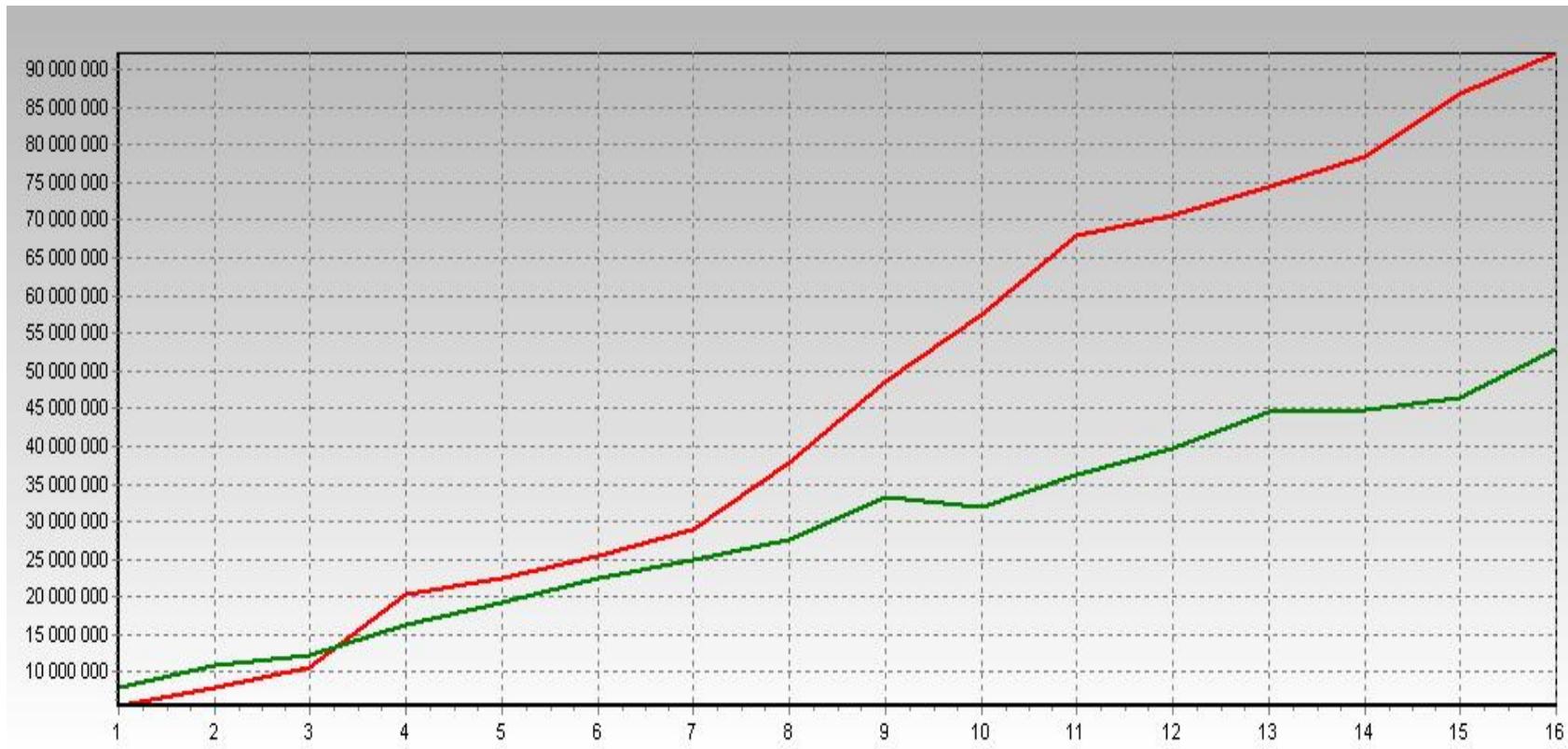
Сравнение эффективности ссылочных и векторных структур



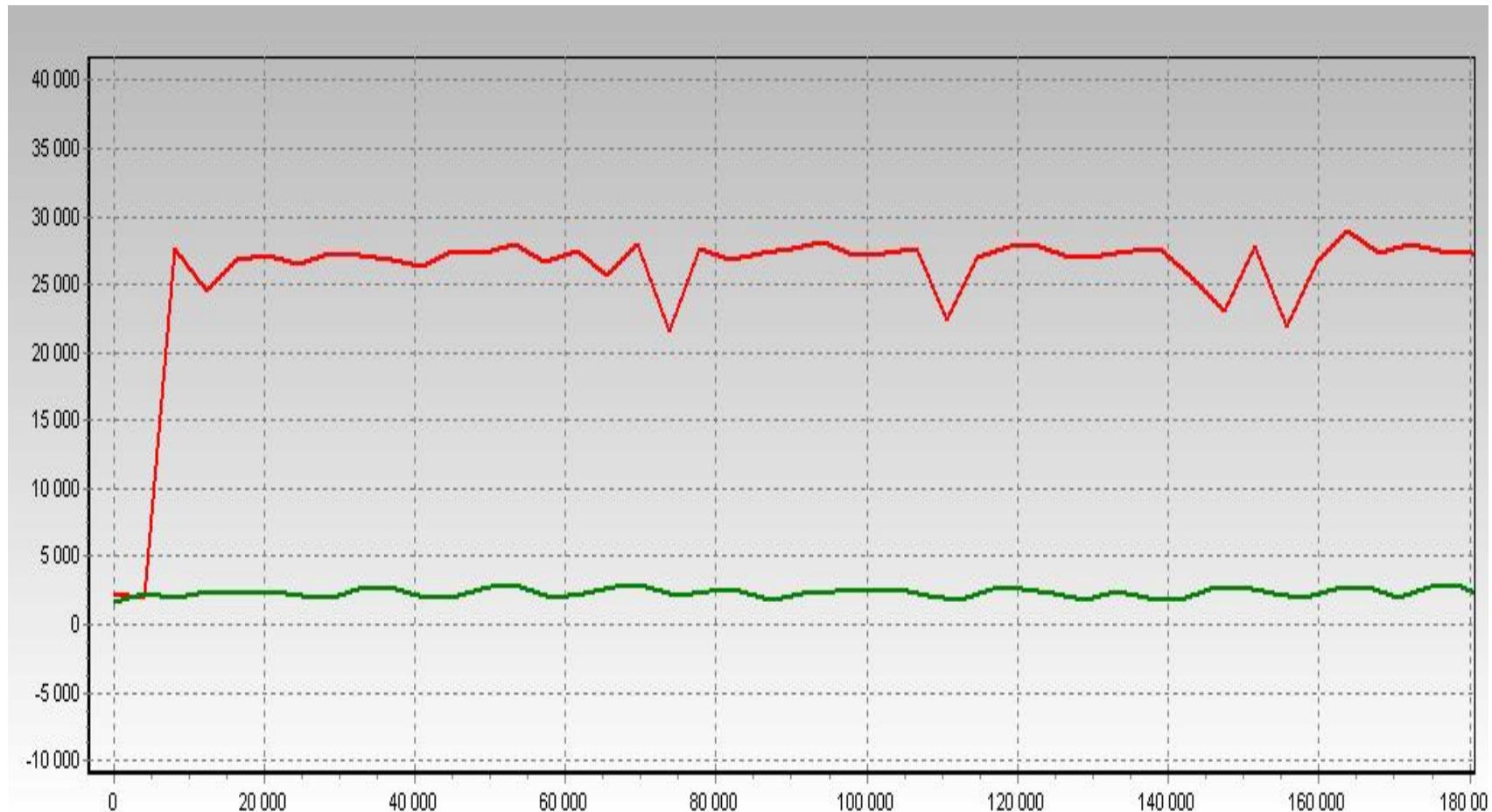
Исследование эффективности предвыборки в TLB



Использование оптимизирующих структур данных



Конфликты в кэш-памяти



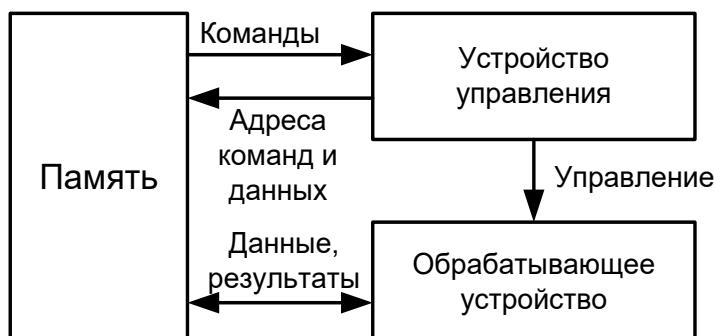
IX. Принципы построения и архитектура ЭВМ

Общие принципы построения современных ЭВМ

Принципы Фон-Неймана

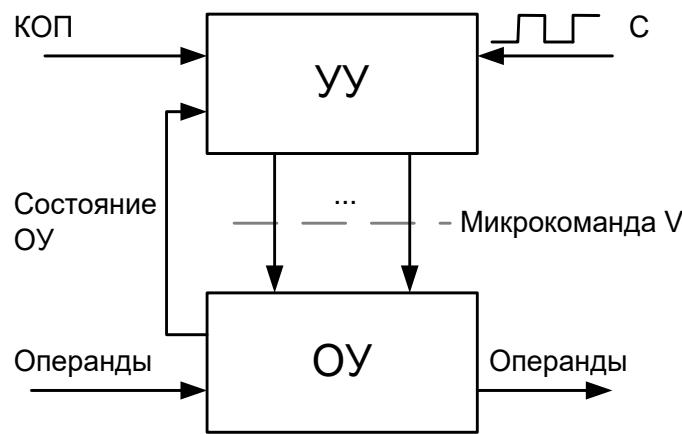
- Двоичное кодирование информации
- Программное управление
- Адресность памяти
- Однородность памяти*

ОКОД, SISD



- Гарвардская архитектура
(ОП для хранения команд и ОП для хранения данных)
- Принстонская архитектура
(ОП для хранения команд и данных)

Принципы микропрограммного управления



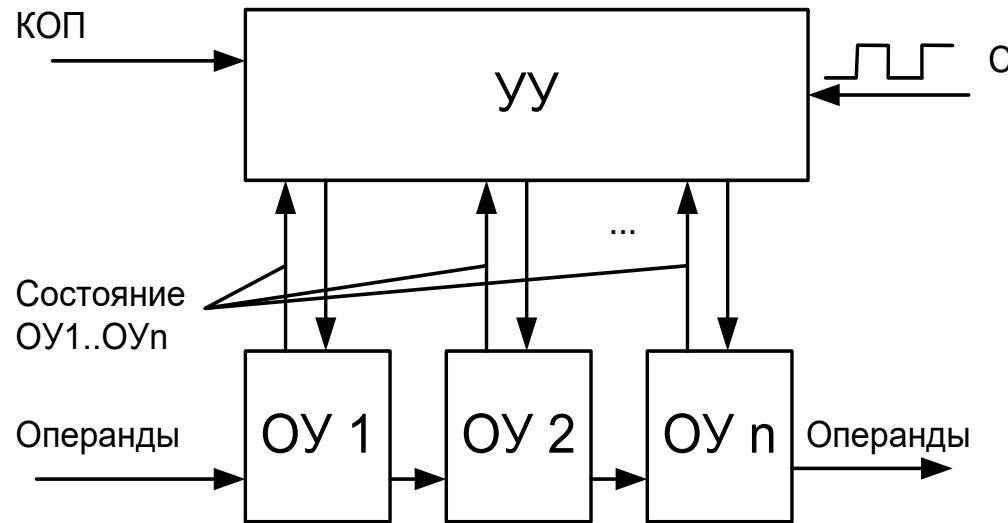
Любое цифровое устройство можно рассматривать, как совокупность операционного и управляемого блока.

Любая команда или последовательность команд реализуется в операционном блоке за несколько тактов

Последовательность сигналов управления должна выдаваться устройством управления в соответствии с поступающей на вход командой и текущим состоянием операционного блока

Состояние линий управления в каждом такте задает микрокоманду. Совокупность микрокоманд, необходимых для реализации команды называется микропрограммой.

Принцип конвейерной обработки

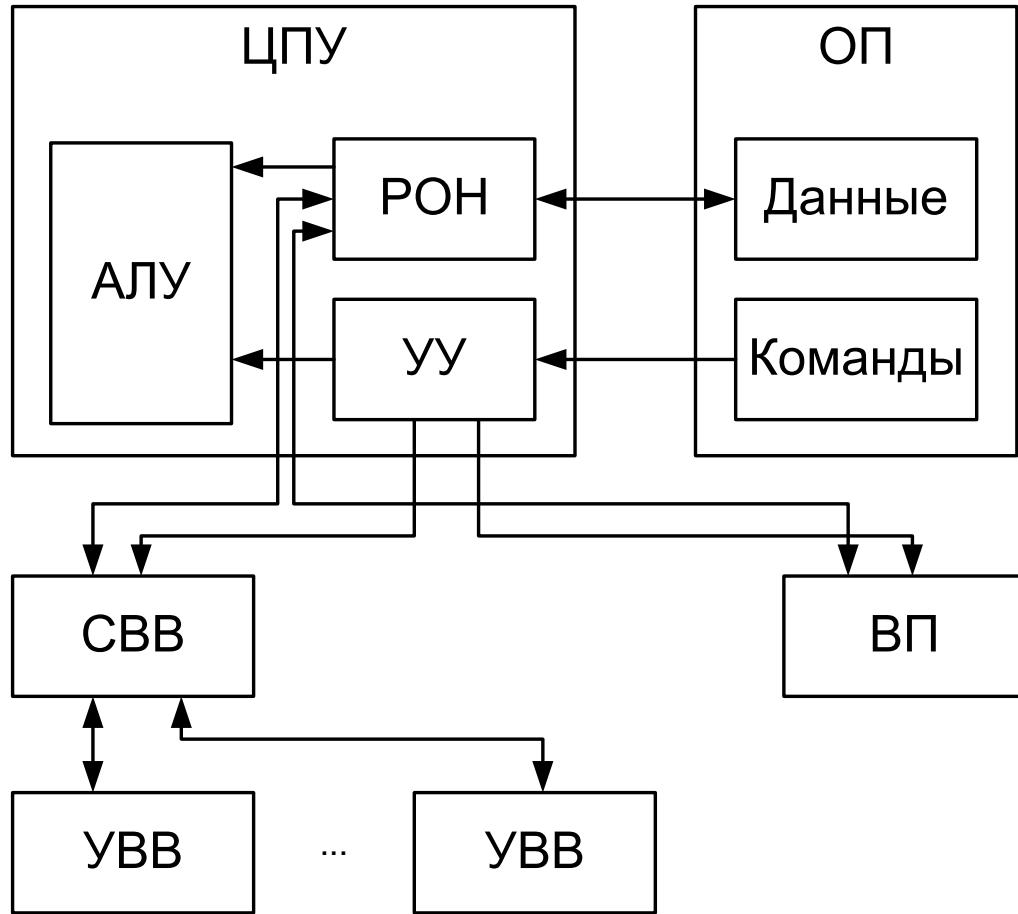


Конвейерная обработка представляет собой процесс, при котором сложные действия разделяются на более короткие стадии. Их параллельное выполнение для последовательности действий позволяет более полно использовать обрабатывающие ресурсы конвейера.

Структура современных ЭВМ

- Центральное процессорное устройство (ЦПУ).
 - Арифметико-логическое устройство (АЛУ)
 - Устройство управления (УУ)
 - Регистры общего назначения (РОН)
- Основная память
- Система ввода-вывода
- Управление внешними устройствами
- Внешняя память
- Система передачи информации
- Система синхронизации
- Система прерываний
- Система прямого доступа к памяти
- Система подвода питания/земли и система энергосбережения
- Система контроля и повышения отказоустойчивости

ЭВМ с непосредственными связями

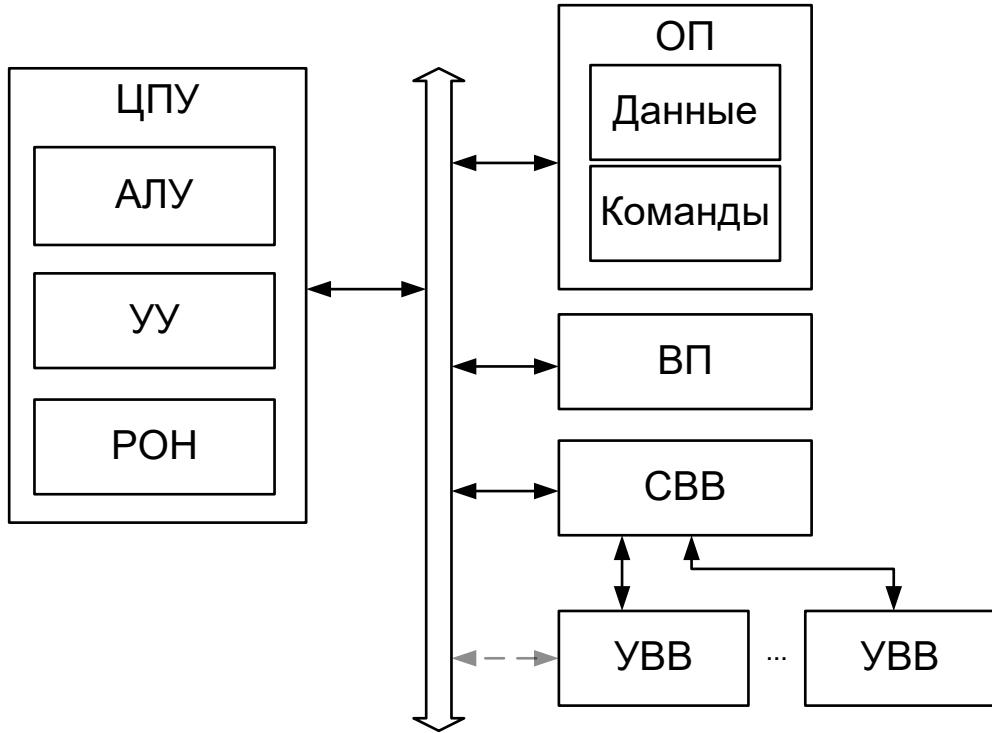


-(+)
При построении
оптимальных линий
связи вычислительная
машина обладает
максимальным
быстродействием.

(-)
Ограничение на
количество выводов
микросхем не позволяет
организовать широкие
шины.

(-)
Реконфигурация
системы требует
изменения характеристик
линий связи.

ЭВМ с магистральной структурой

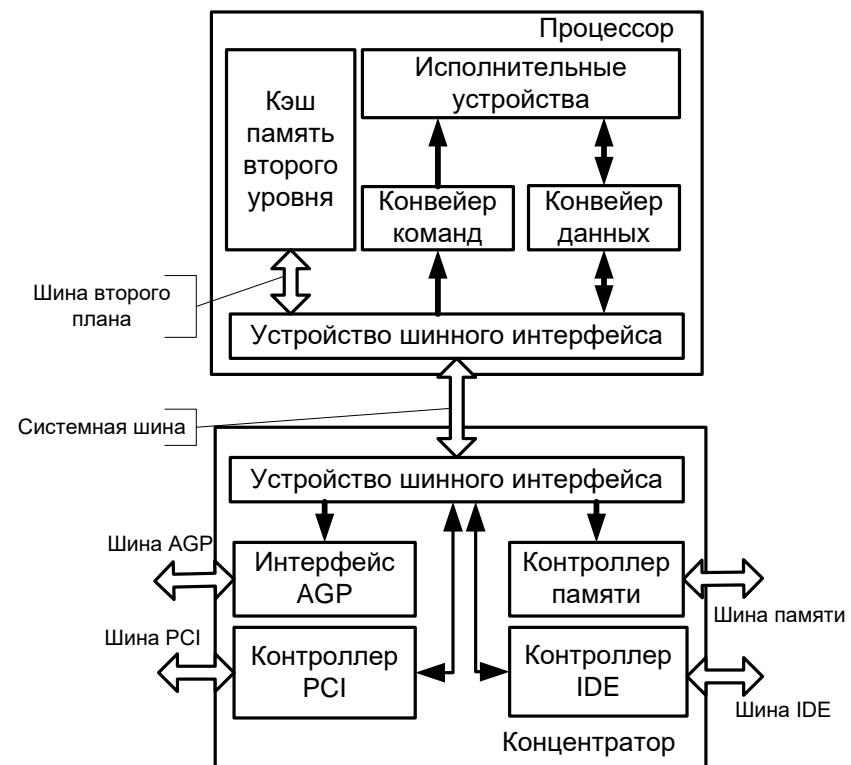
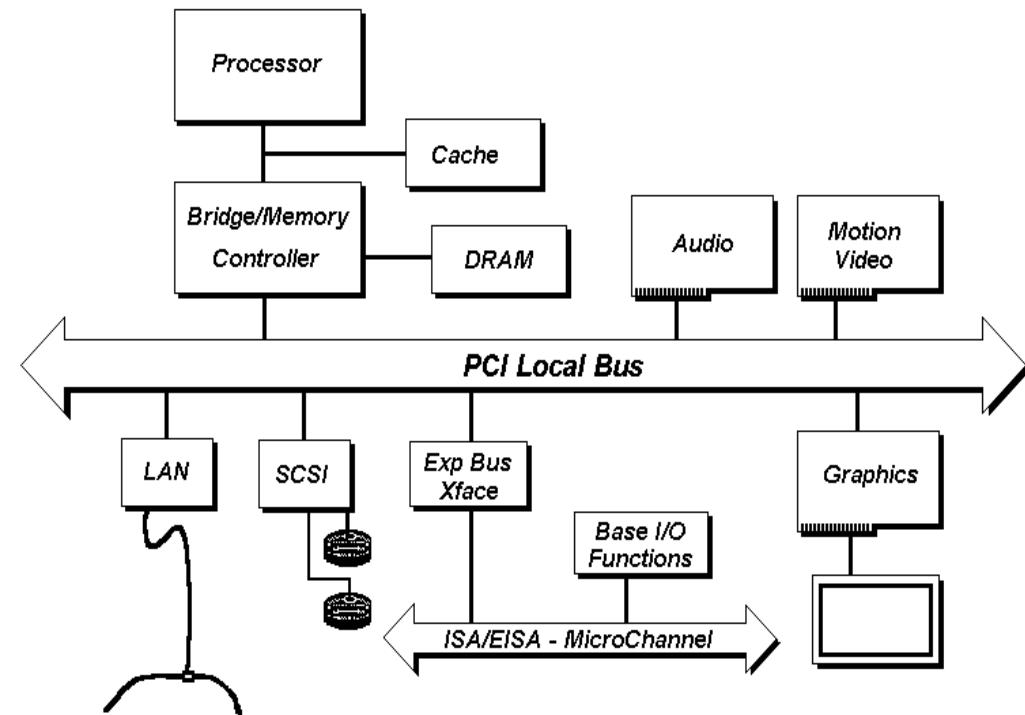


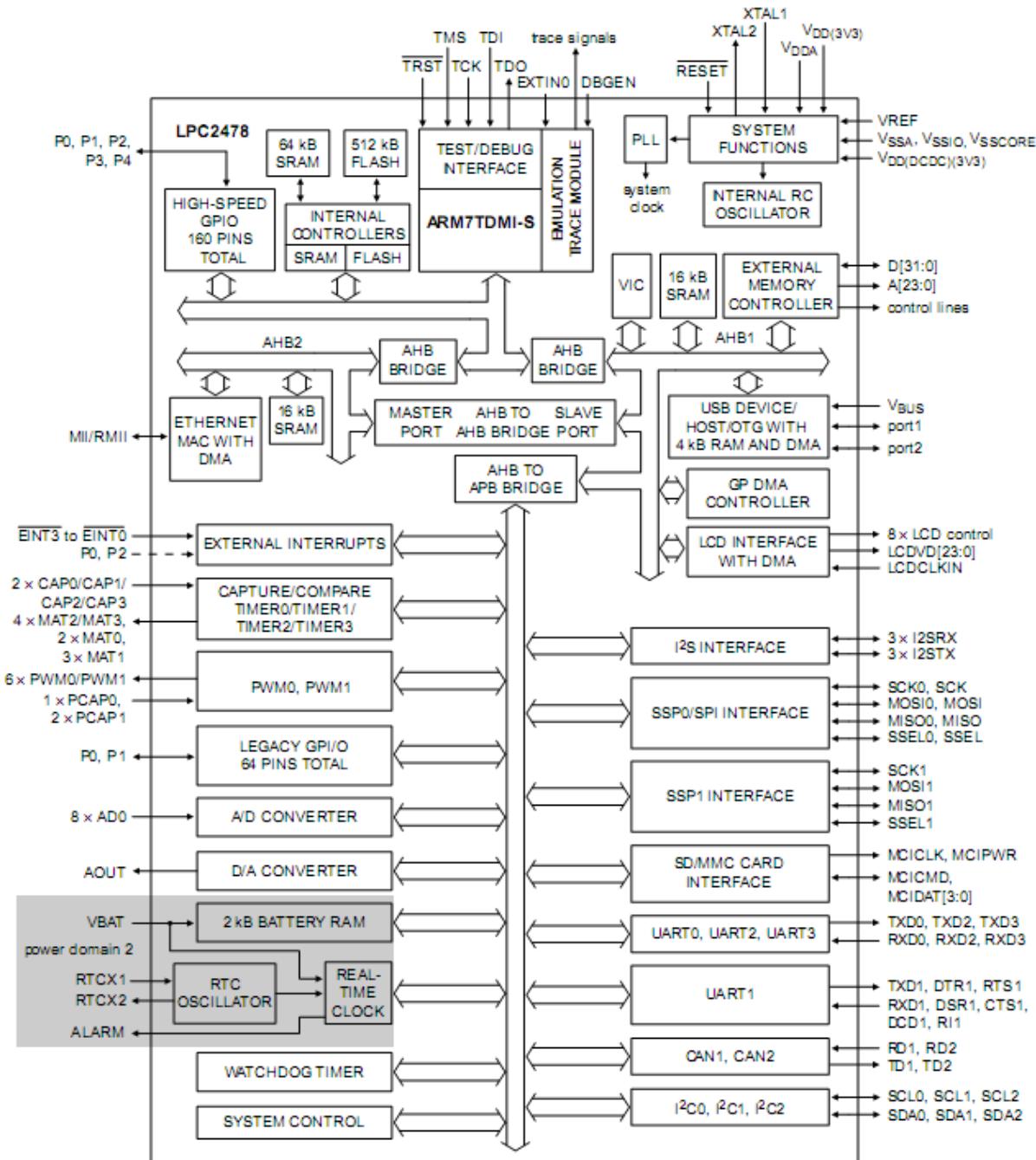
(+) Общая шина позволяет легко реконфигурировать систему.

(-) Шина является узким местом.

- Шина, используемая всеми устройствами системы для передачи данных называется системной.
- Для разгрузки системной шины используют иерархию шин.
- По назначению, разделяют шины адреса, шины данных и шины управления.

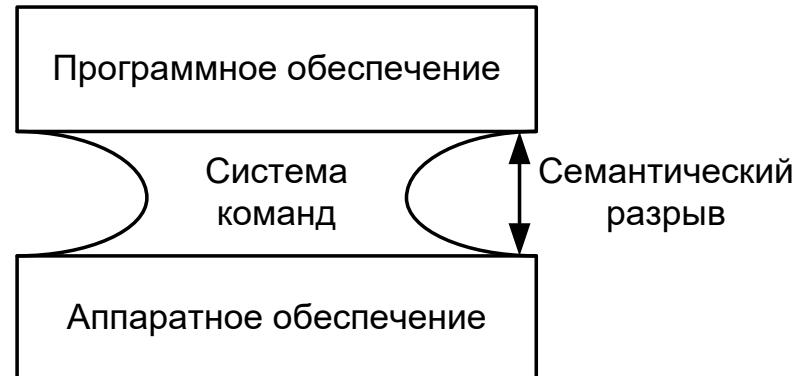
Примеры построения ЭВМ с иерархией шин





Основные тенденции развития ЭВМ

- Повышение степени интеграции элементной базы
 - Увеличение набора команд
 - Увеличение степени аппаратной поддержки.
- Обратная совместимость
- Наличие семантического разрыва



Проблема семантического разрыва

Технология программирования непрерывно развивается, что позволяет увеличивать функциональность программ и сокращать время их разработки. Создание проблемно-ориентированных языков высокого уровня усугубляет принципиальное отличие языка машинных команд, реализуемого компьютером, от языков, используемых при написании программ. Данная проблема носит название "семантического разрыва" и выражается в неоправданном падении производительности вычислительной системы.

Архитектура системы команд

В команде указывается, какую операцию выполнять (КОП), над какими операндами выполнять операцию, а также куда поместить operand.



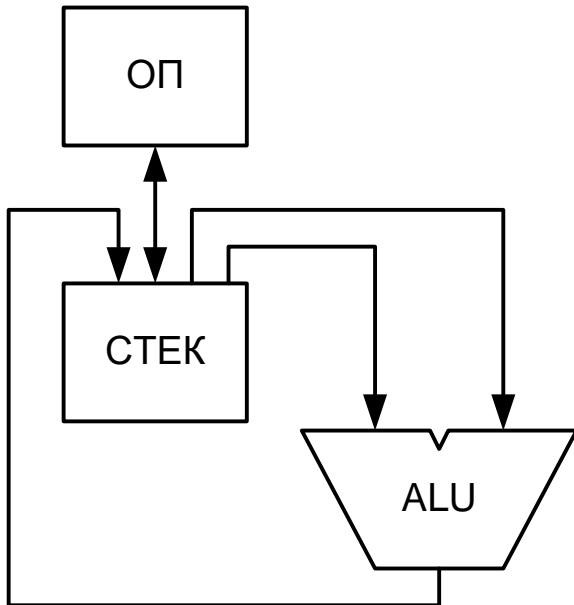
RISC – Reduced Instruction Set Computer; CISC – Complex Instruction Set Computer;
VLIW – Very Long Instruction Word; ROSC - Removed Operand Set Computer

Сравнение CISC, RISC и VLIW архитектур СК

Характеристика	CISC	RISC	VLIW
Длина команды	Различная	Однаковая	Однаковая
Расположение полей в командах	Различное	Однаковое	Однаковое
Количество регистров	Малое. Регистры специализированные	Большое. Регистры универсальные	Большое. Регистры универсальные
Доступ к памяти	Кодируется в команде. Выполняется по микрокоманде	Выполняется по специальной команде	Выполняется по специальной команде
Длительность выполнения команд	Различная	Однаковая (для большинства команд)	Различная

Стековая архитектура СК

(+) При размещении операндов в стековой памяти (LIFO) архитектура команд упрощается (большое количество действий выполняется аппаратно)



Операции:

- занесение в стек (PUSH);
- извлечение из стека (POP);
- выполнение действий на стеком
(извлечение operandов из вершины стека,
выполнение действий, помещение
результата в вершину стека)

Для выполнение арифметических операций
их преобразуют к постфиксной форме
(Польской записи).

Пример: $a = a + b * (c - d)$; Постфиксная форма: abcd-*+;

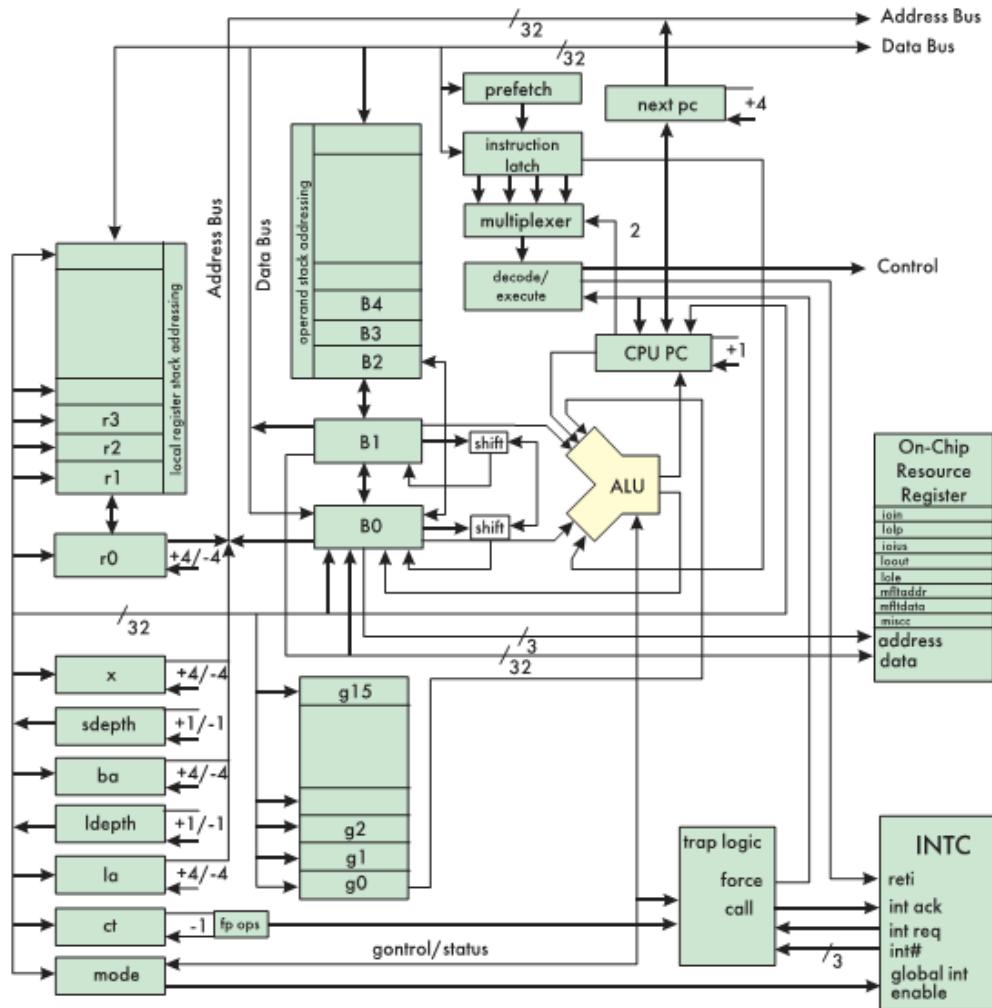
Действия: PUSH a; PUSH b; PUSH c; PUSH d; SUB; MUL; ADD; POP a.

(-) Отсутствие прямого доступа к памяти ограничивает область применения.

(-) Сложность организации параллельной обработки.

Стековые процессоры (Форт-процессоры)

Блок-схема микропроцессора IGNITE



Сравнение выполнения программы на RISC-процессоре и на стековом микропроцессоре

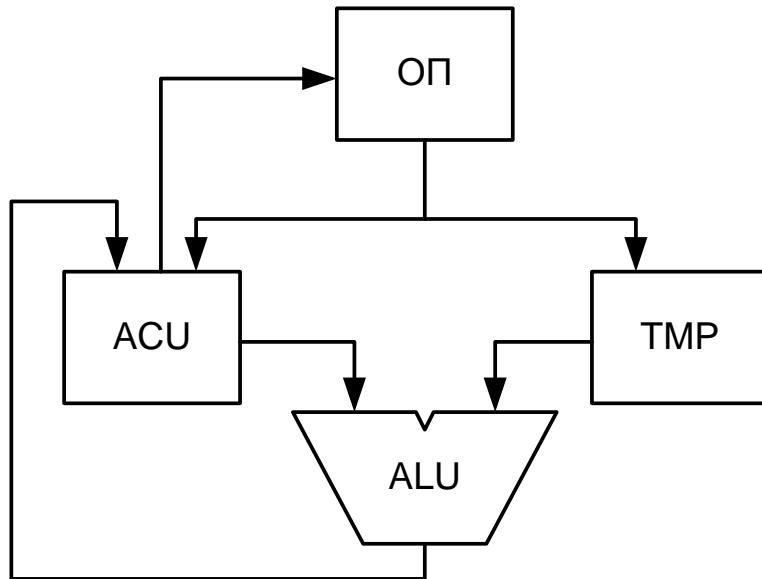
Номер команды	RISC MPU	IGNITE
1	add #1, g2, g5	push g1 push g2 inc #1
2	sub g1, g5, g5	sub
3	add g5, g3, g5	push g3 add
4	shl g4, #1, temp	push g4 shl #1
5	sub g5, temp, g5	sub pop g5
	Всего 20 байт	Всего 10 байт

Набор микросхем TDS9092 FORTH CHIPS



Аккумуляторная архитектура СК

Один из операндов должен обязательно находиться в специальном регистре-аккумуляторе. Результат также сохраняется в аккумулятору.



Операции:

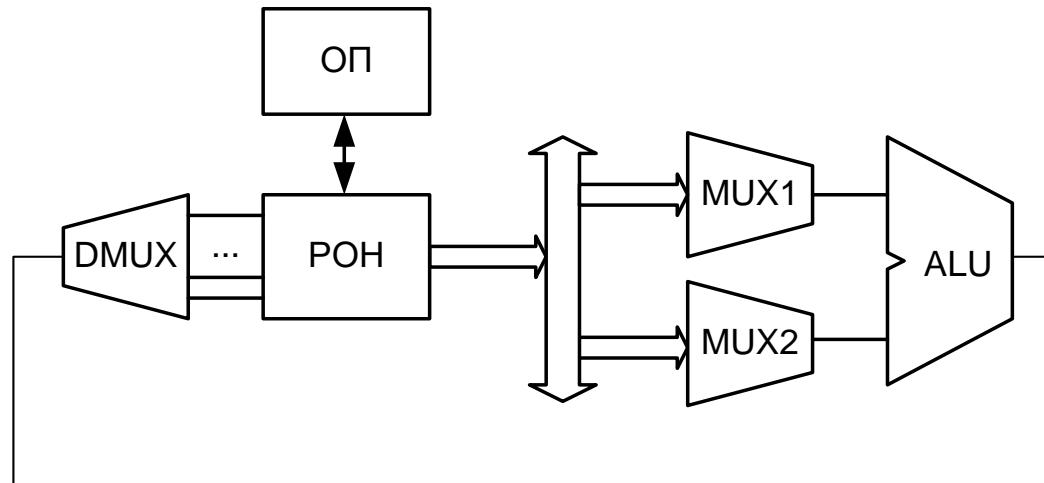
- занесение в аккумулятор (LOAD);
- извлечение из аккумулятора (STORE);
- выполнение действий над operandами (извлечение первого операнда из аккумулятора, извлечение второго операнда из ОП и помещение во временный теневой регистр TMP, выполнение действий, помещение результата в аккумулятор).

Пример: $a = a + b * (c - d)$; Определение троек: $T1=c-d$; $T2=b*T1$; $T3=a+T2$;
Действия: LOAD c; SUB D; MUL b; ADD a; STORE a.

- (+) В команде необходимо указывать только адрес второго операнда.
- (+) Ускоряются длинные вычисления ($a*b/c+d-e$).
- (-) Наличие одного аккумулятора является узким местом, т.к. временно ненужный результат необходимо перезаписывать в другой регистр или ОП.

Регистровая архитектура СК

В состав процессора входит большое количество однотипных регистров. В команде необходимо указать номера регистров, хранящих операнды, а также номер регистра операнда.



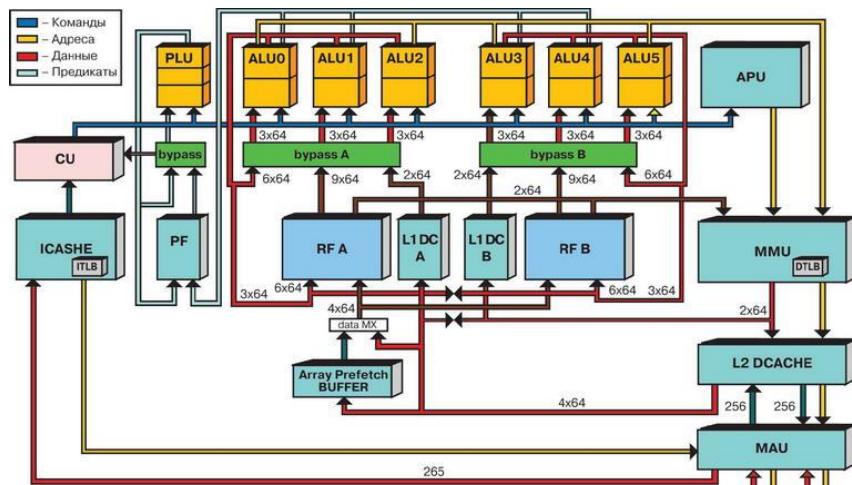
Для данной архитектуры возможны варианты размещения operandов: оба операнда в памяти; один операнд в памяти и один в РОН; оба операнда в РОН.

Для уменьшения размерности команд и для упрощения декодирования накладывают ограничения на размещение operandов.

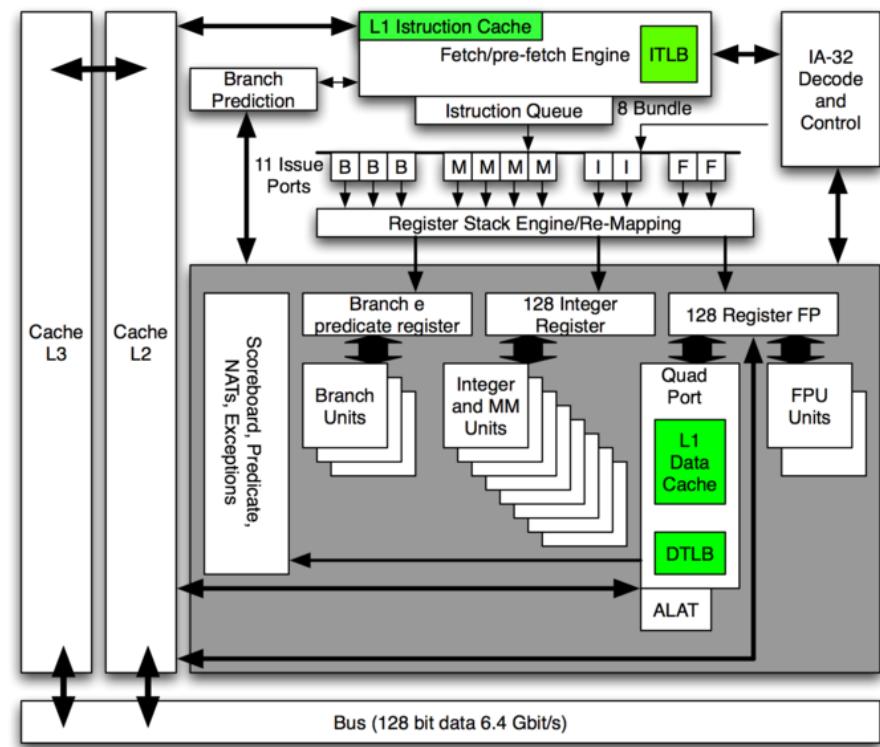
Архитектура VLIW – Very Long Instruction Word

В процессорах VLIW задача распределения решается во время компиляции и в инструкциях явно указано, какое вычислительное устройство должно выполнять какую команду.

Эльбрус-3 и его микропроцессорное исполнение Эльбрус 2000 (E2K) также являются VLIW процессорами.



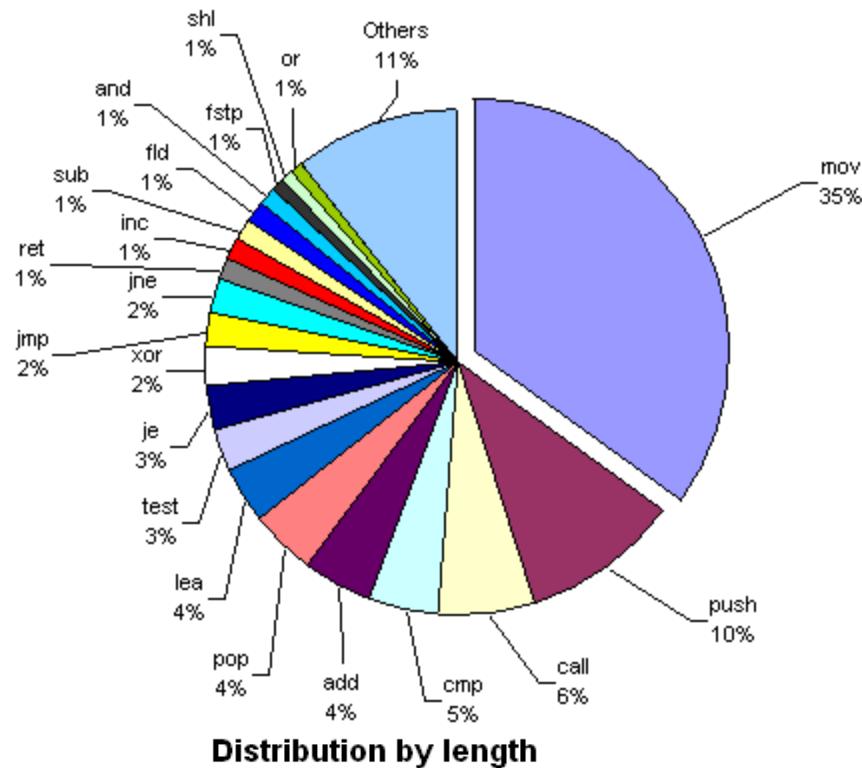
Микропроцессор Intel Itanium имеет как традиционную систему команд IA-32, так и систему команд «с явным параллелизмом» (англ. Explicitly Parallel Instruction Computing, EPIC), исполняемую VLIW-ядром



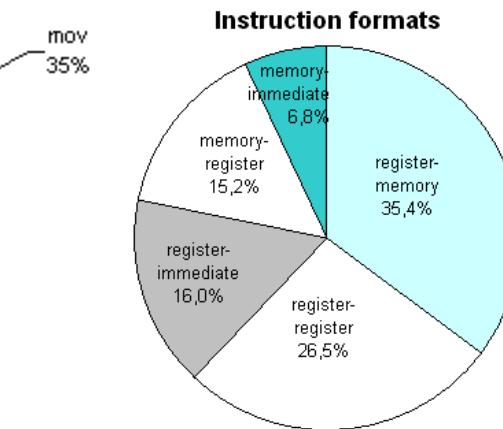
Вариант	(+)	(-)
Оба операнда находятся в регистрах	Простота аппаратной реализации. Простота параллельной обработки.	Избыточность в команде из-за сложности кодирования с кратностью 8 бит
Один operand находится в регистре, а один в памяти	Код компактен. Данные поступают в ALU без промежуточного хранения в РОН	Наличие адреса в команде усложняет дешифрацию и сокращает возможное кол-во РОН, адресуемых в команде.
Оба операнда находятся в памяти	Код наиболее компактен. Возможность выполнения простых действий наиболее быстро без занесения в РОН	Команды имеют максимальную длину. Из-за наличия коротких и длинных команд трудно оптимизировать тракты передачи данных и декодеры инструкций

Статистические данные для x86 команд

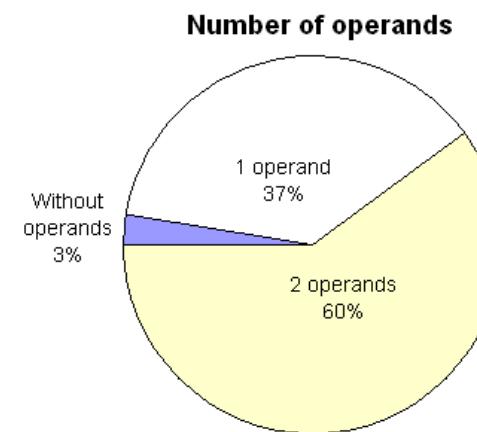
Top 20 instructions of x86 architecture



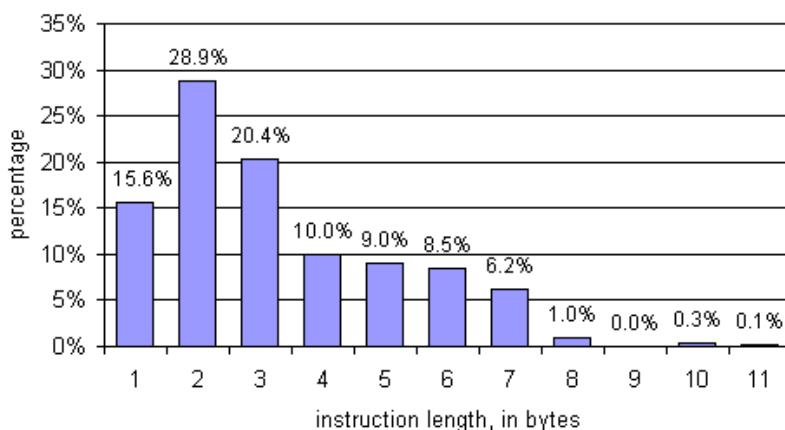
Distribution by length



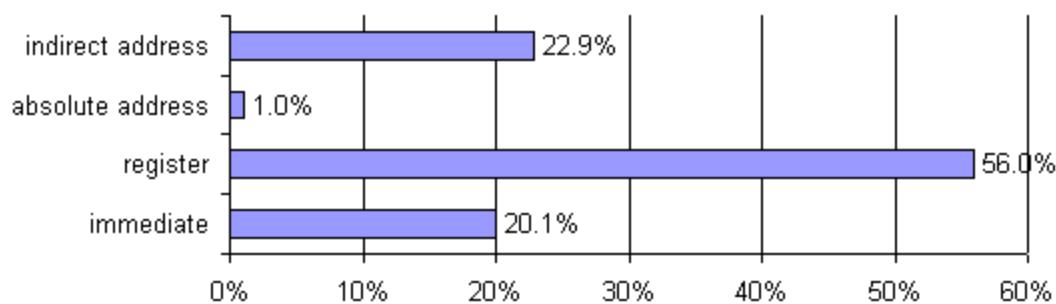
Instruction formats



Number of operands



Operand types in x86 architecture



Типы команд.

- Команды пересылки данных.
 - регистр-регистр
 - регистр-память
 - память-память
- Команды арифметической и логической обработки (сложение, вычитание, умножение, деление, инкремент, декремент, сравнение, операции над ЧПЗ, логические операции, операции сдвига).
Сдвиг: логический, арифметический, циклический, циклический через дополнительным разряд.
- Команды работы со строками (могут быть реализованы набором других команд, однако удобны при работе с символьной информацией).
- Команды векторной обработки (позволяет выполнять однотипные действия над большим количеством однородных данных). Пример арифметики с насыщением:
$$\begin{array}{r} 1011\ 0111\ 1010 \\ +\ \underline{0001\ 1001\ 1000} \\ \hline 1100\ 1111\ 1111 \end{array}$$
- Команды преобразования: служат для табличного преобразования данных из одной системы кодов в другую (2-10 <-> 2)

- Команды ввода/вывода. Служат для управления, проверки состояния и обмена данными с периферийными устройствами.

- Команды вывода в порт

- Команды ввода из порта.

- Команды управления потоком команд. Данные команды служат для указания очередности выполняемых команд.

Вычисление адреса очередной команды может выполняться несколькими способами:

- увеличением адреса на длину исполненной (естественный порядок).

- изменением адреса на длину следующей (перешагивание)

- изменением адреса на значение, указанное в текущей команде (короткий переход).

- непосредственное указание следующей команды (длинный переход).

Перечисленные команды могут выполняться лишь по некоторому условию (условные переходы).

Команды условного перехода составляют 80% команд управления.

Команды безусловного перехода: вызовы и возвраты из процедур, и.т.д.

Форматы команд.

Операционная часть	Адресная часть
--------------------	----------------

1. Четырехадресная команда.

КОП	1 операнд	2 операнд	результат	Адр след ком.
-----	-----------	-----------	-----------	---------------

2. Трехадресная команда

КОП	1 операнд	2 операнд	результат
-----	-----------	-----------	-----------

3. Двухадресная команда.

КОП	1 операнд	2 оп-д/результат
-----	-----------	------------------

Характерна для CISC-архитектуры

4. Аккумуляторная архитектура

КОП	1 операнд
-----	-----------

Второй operand хранится в аккумуляторе.
Данный формат команд характерен для RISC-архитектур.

5. Нульоперандная команда.

КОП

Способы адресации

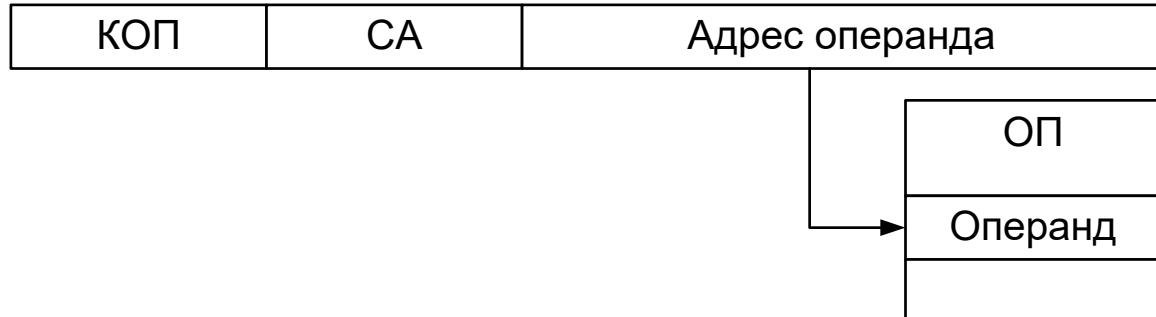
Адресная часть		
Способ адресации (СА)	Адрес/операнд	
<u>Непосредственная адресация</u>	КОП	СА

Вместо адреса команда содержит непосредственно операнд.

(+) команда выполняется быстро

(-) непосредственный операнд может не войти в команду

Прямая адресация



Адрес в команде является адресом операнда

(+) если operand находится в памяти, то это самый быстрый способ указать на него

(-) заранее определенный адрес влияет на переносимость программы.

(-) Адрес занимает много места

Неявная адресация

КОП	СА
-----	----

Операнд подразумевается (следует из КОП).

(+) Команда занимает мало места

(-) только такие командах нельзя использовать для построение всей системы команд.

Регистровая адресация

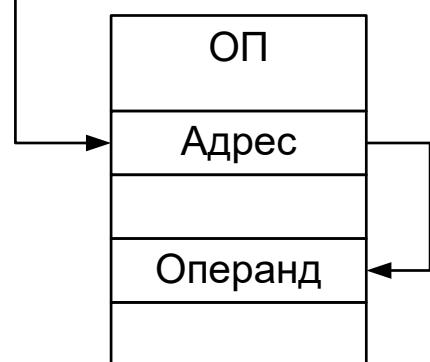
Адрес в команде указывает не на ячейку ОП, а на регистр.

(+) Быстрее прямой адресации

(-) Количество регистров ограничено

Косвенная адресация

КОП	СА	Адрес операнда
-----	----	----------------



Адрес в команде указывает на ячейку памяти, в которой находится адрес операнда.

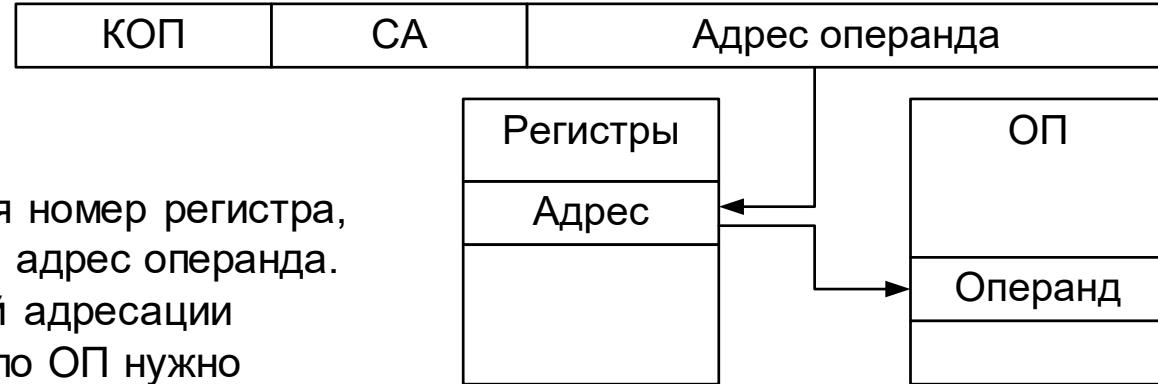
(+) удобна для обработки структурных типов данных.

(-) приходится осуществлять много обращений к ОП.

Косвенная регистровая адресация

В команде содержится номер регистра, в котором содержится адрес операнда.

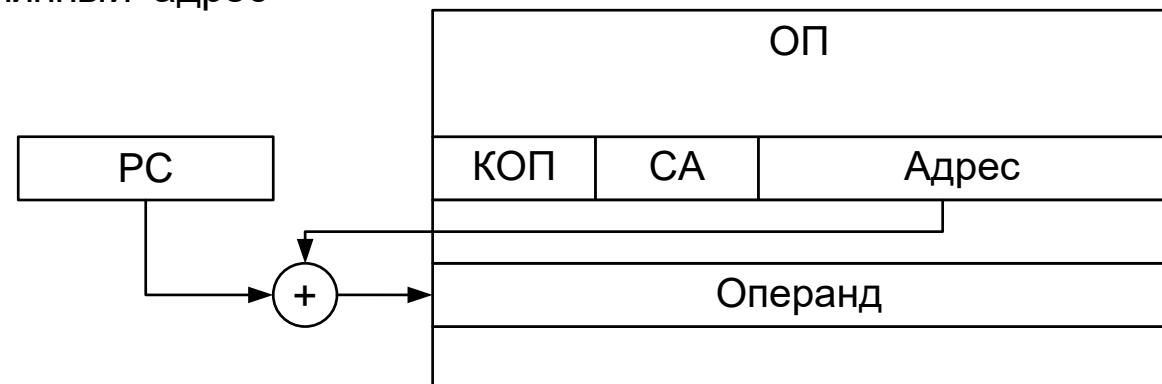
- (+) быстрее косвенной адресации
- (-) для перемещения по ОП нужно менять содержимое регистра



Относительная адресация

Адрес вычисляется относительно счётчика команд

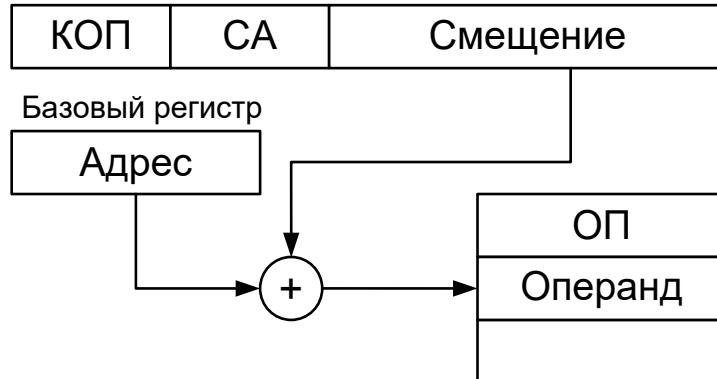
- (+) Код переносим, команды занимают мало места
- (-) Может понадобиться длинный адрес



Базовая регистровая адресация

Адрес в команде представляет собой смещение, которое складывается со значением в базовом регистре для получения адреса операнда

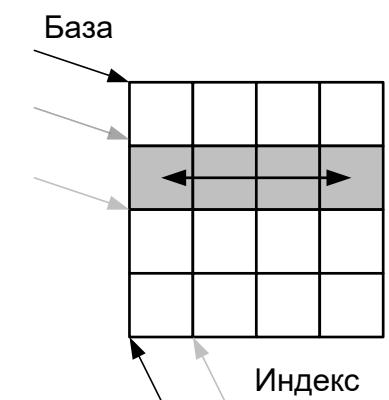
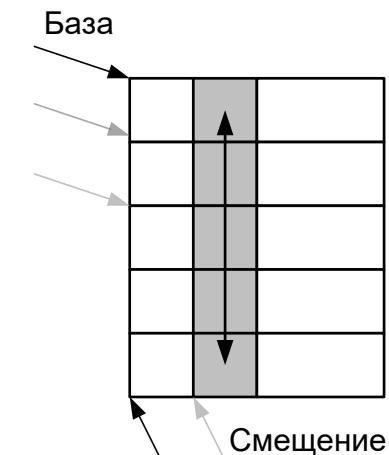
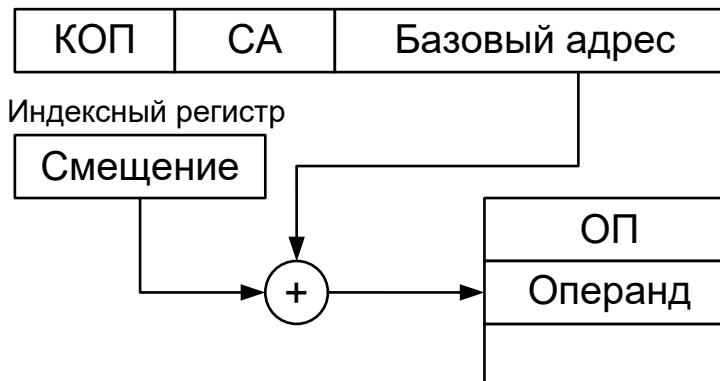
- (+) Удобна для работы со структурами данных, размещаемых динамически.
- (-) Переносимость меньше, чем у относительной адресации



Индексная регистровая адресация

В поле адреса команды содержится базовый адрес, складываемый со значением смещения в индексном регистре.

- (+) Удобна для работы со структурами данных, размещаемых динамически.
- (-) Переносимость меньше, чем у относительной адресации



Автоинкрементная/автодекрементная адресация

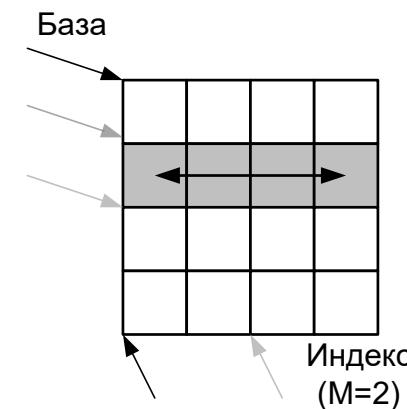
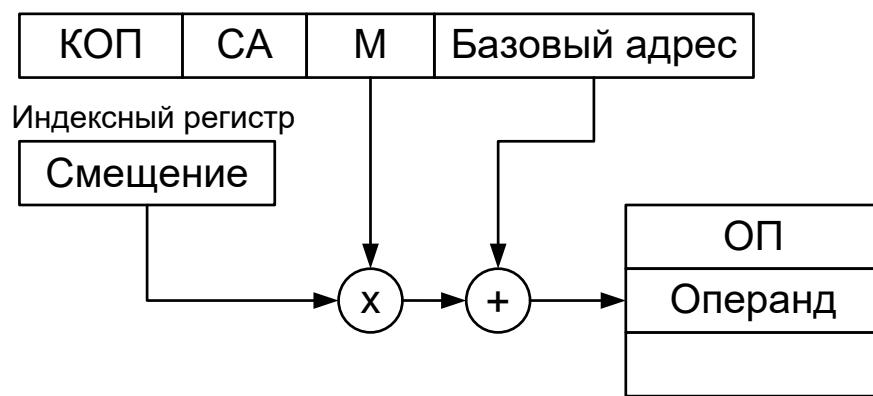
Разновидность регистровой индексной или базовой адресации. До или после выполнения команды значение базового или индексного регистра увеличивается/уменьшается на единицу или масштаб.

- (+) Способ адресации удобен для команд обработки строк.
- (-) Автоматическое изменение часто требуется выполнять на величину, большую единицы.

Индексная адресация с масштабированием

Индексный регистр умножается на масштаб M и суммируется с базовым адресом из команды.

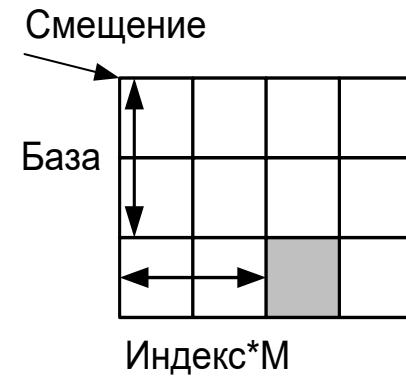
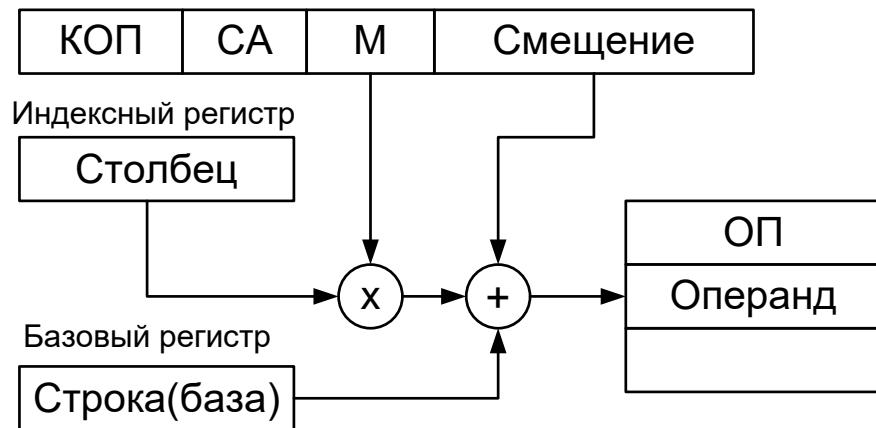
- (+) Удобен для модификации адреса на величину M .
- (-) Вычисление адреса замедляется, т.к. требуется выполнять умножение.



Базовая индексная адресация с масштабированием

Адрес определяется по формуле Адрес=Индекс*Масштаб+База+Смещение.

- (+) Базовая индексная адресация с масштабированием часто используется при обращении к системным таблицам, находящимся в ОП (таблица дескрипторов, таблицы страниц, таблица векторов прерываний и т.д.)
- (-) Ограничено на величину M (M=1,2,4,8).



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.

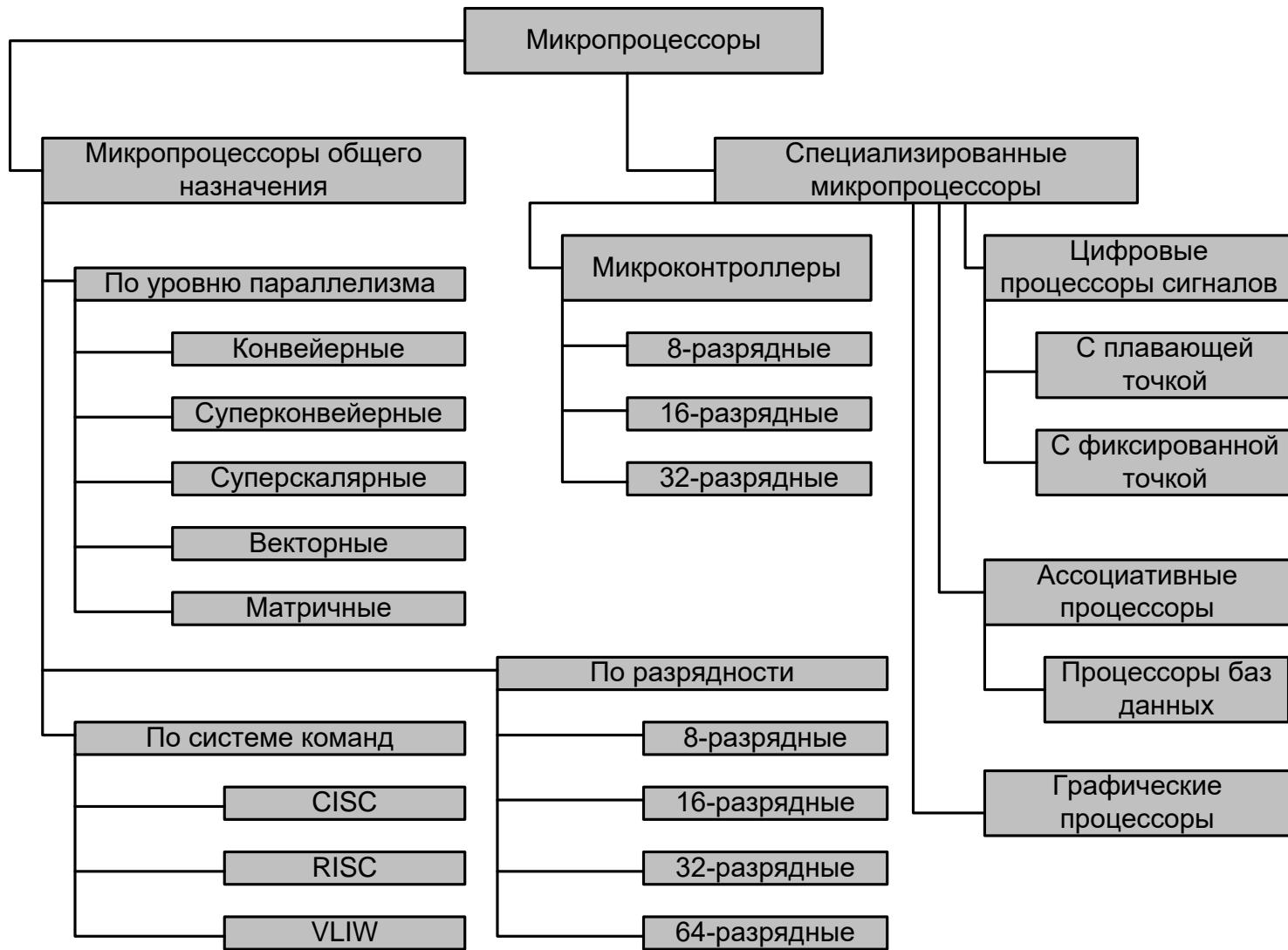
Процессором (процессорным ядром) называется устройство ЭВМ, непосредственно осуществляющее процесс переработки информации и управление им в соответствии с заданным алгоритмом, который, как правило, представлен программой.

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 временными, электрическим и конструктивным параметрам, совместное использование которых позволяет реализовать основные функции ЭВМ.

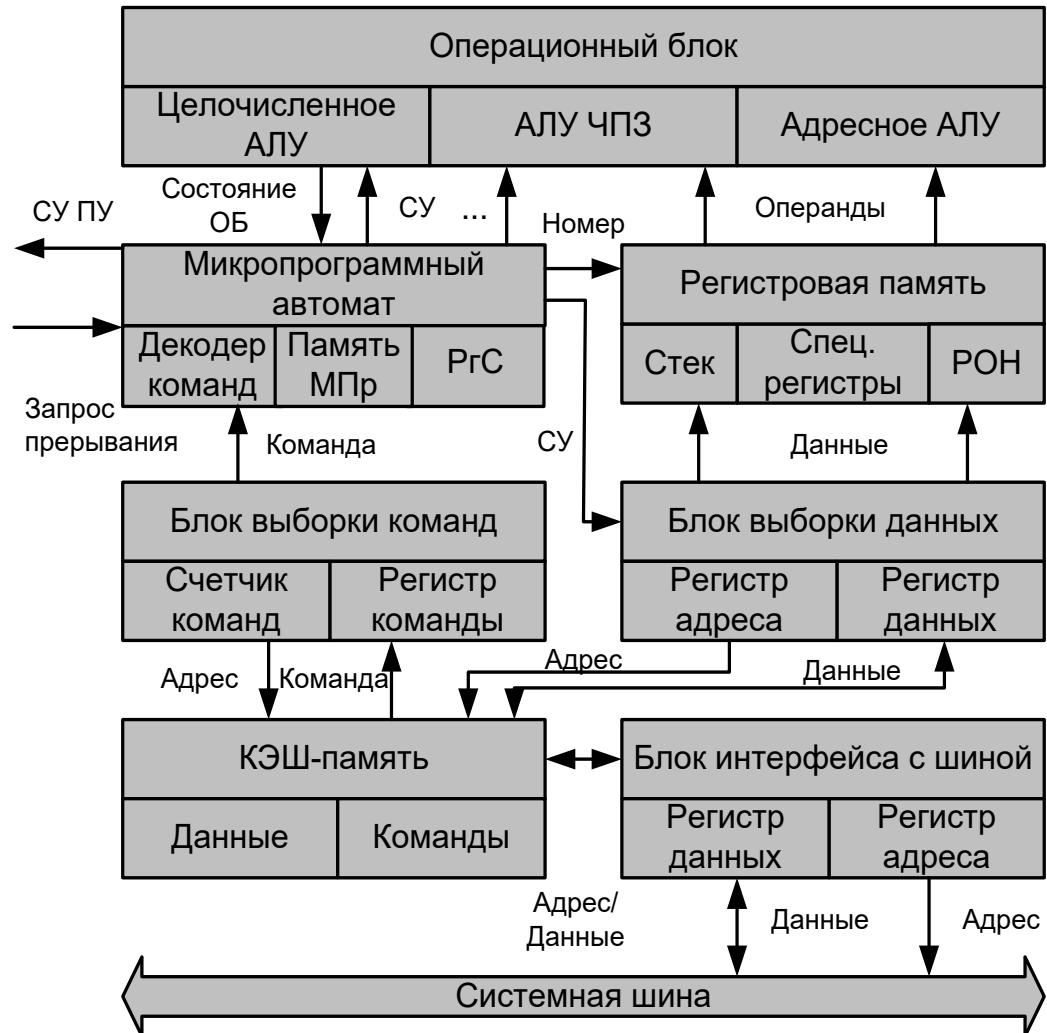
Классификация процессорных устройств



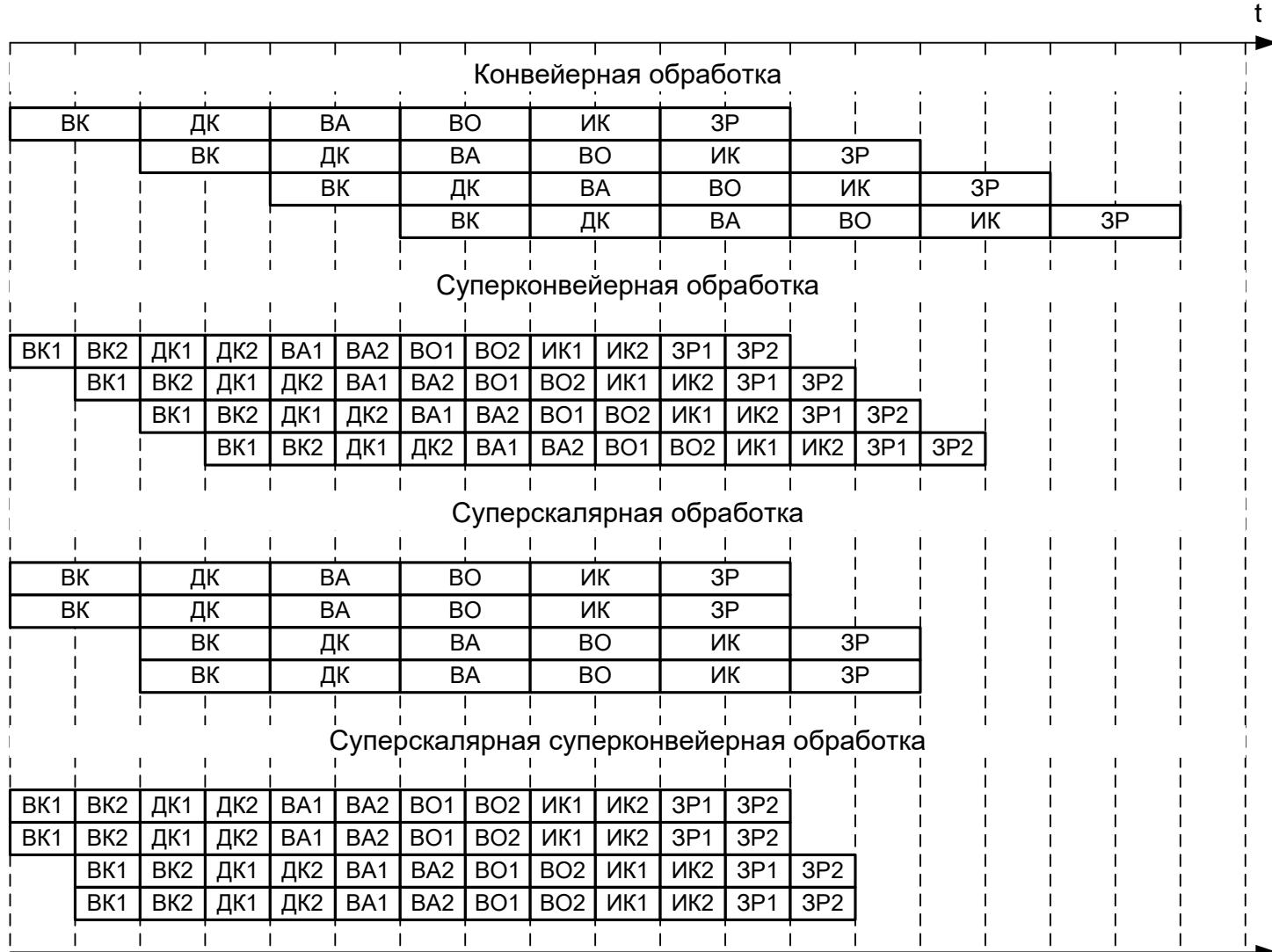
Обобщенная структура универсального процессорного устройства

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).

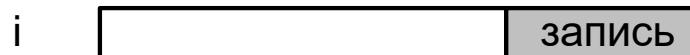
2. Риск по данным

Команды имеют зависимость по данным.

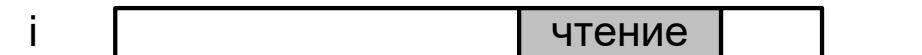
$O(i)$ – множество ячеек, изменяемых командой i ;

$I(j)$ – множество ячеек, читаемых командой j .

А) Чтение после записи (ЧПЗ).



Б) Запись после чтения (ЗПЧ).



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

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

В) Запись после записи (ЗПЗ).



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

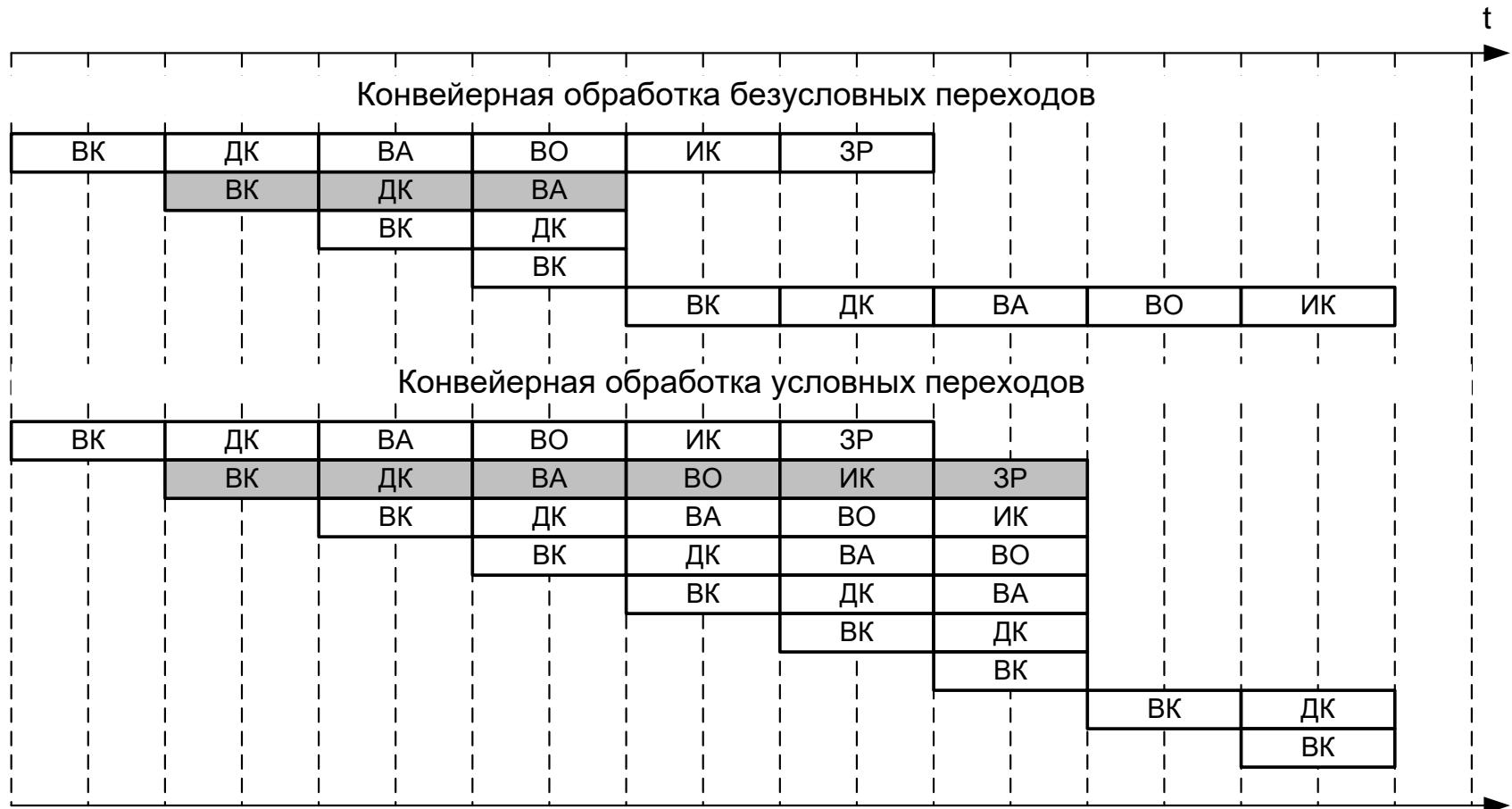
3. Риск по управлению.

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 operandов и исполнения команды ветвления.

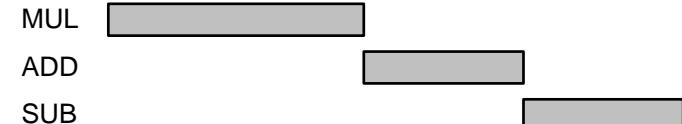
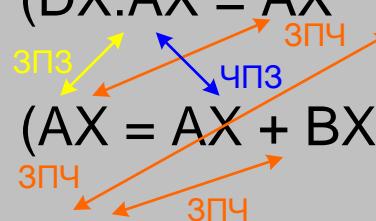
Временные потери при обработке команд переходов



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)

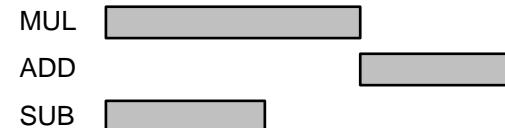


Правило:

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



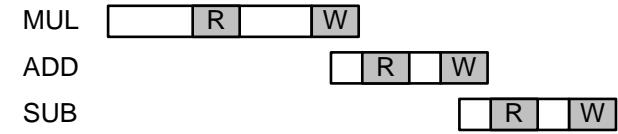
Конфликт типа ЧПЗ по данным, находящимся в регистрах, может быть устранен с помощью бита достоверности

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)

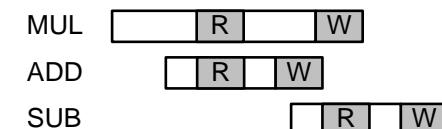
ЗПЧ
ЗПЗ
ЧПЗ



Правило:

При обнаружении конфликтов по данным, находящимся в ОП, запросы на запись результатов в память выполняются 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 (Pattern 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%).

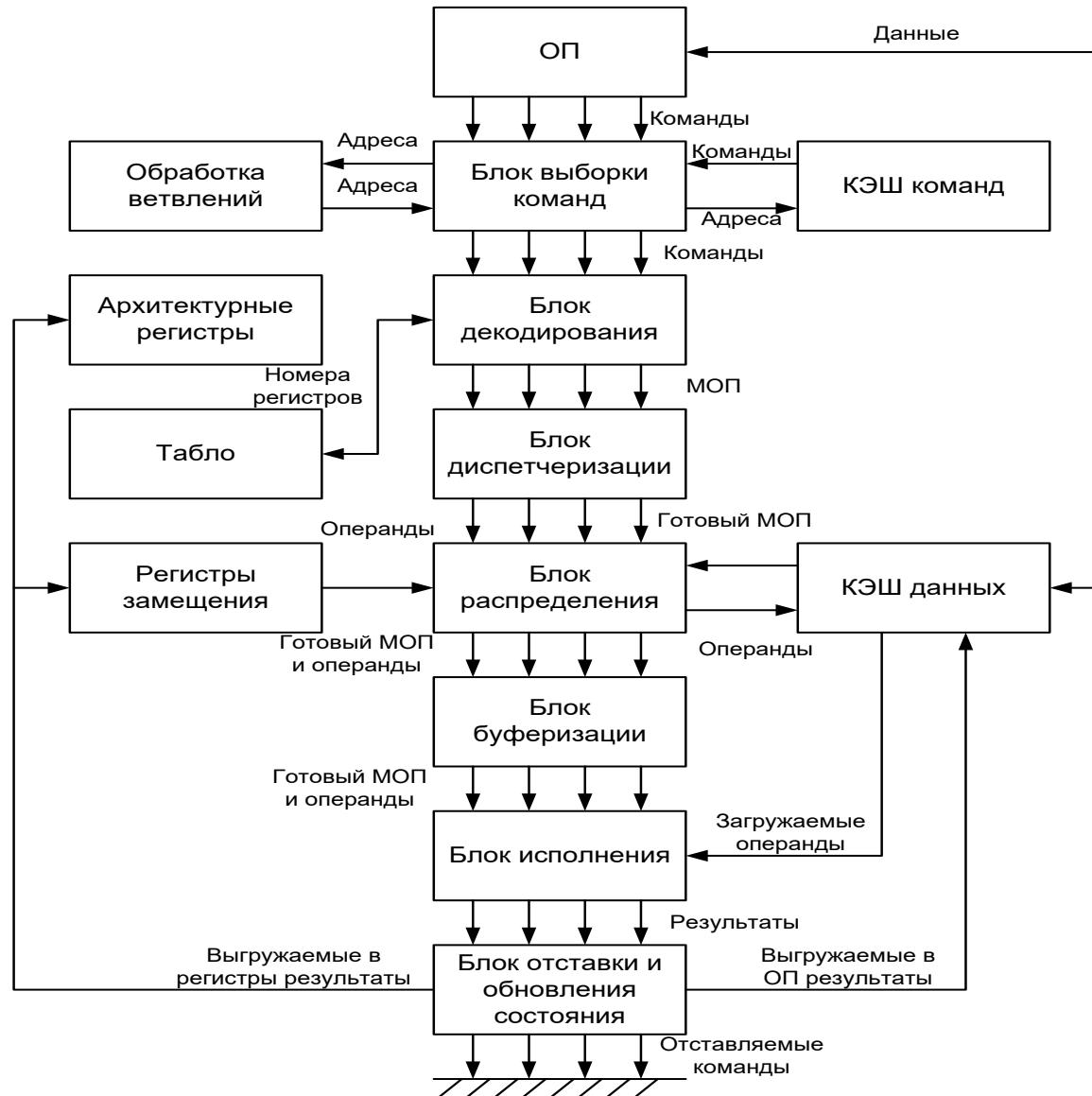
При хранении двух бит учитывается переход для двух последних исполнений команды (точность ~82%).

- Таблица меток перехода (Branch Target Buffer)

-Двухуровневое предсказание.

-Гибридное предсказание

Обобщенная схема суперскалярного суперконвейерного процессора



Структура процессора Р6

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. Страницчная 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.

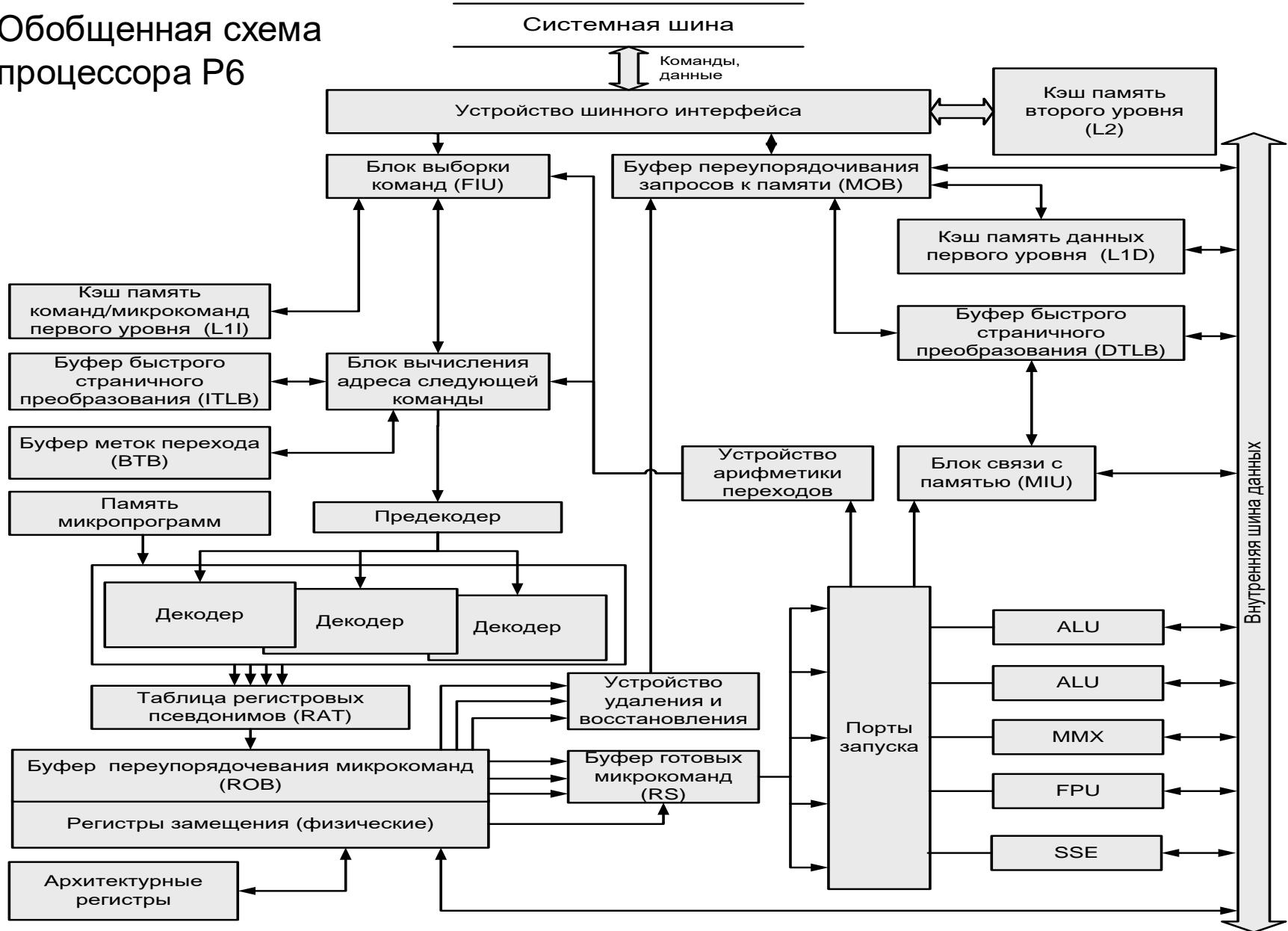
История процессоров с архитектурой IA32 (продолжение)

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 ветвлений. 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 выполнение команд (анализ зависимостей по данным, неупорядоченное и спекулятивное выполнение, предсказание ветвлений). Интегрированный смешанный кэш второго уровня (до 2 Мб). SSE (128 разрядов)
Семейство NetBurst (Pentium 4, Xeon, Pentium Ex. Ed.)	2000	Быстрое выполнение на удвоенной скорости. Гиперконвейерная суперскалярная организация. Глубокая предвыборка. Кэш трасс. SSE2 и SSE3. Hyper-Threading, Intel Virtualization Technology, Intel 64 Architecture, Dual Core

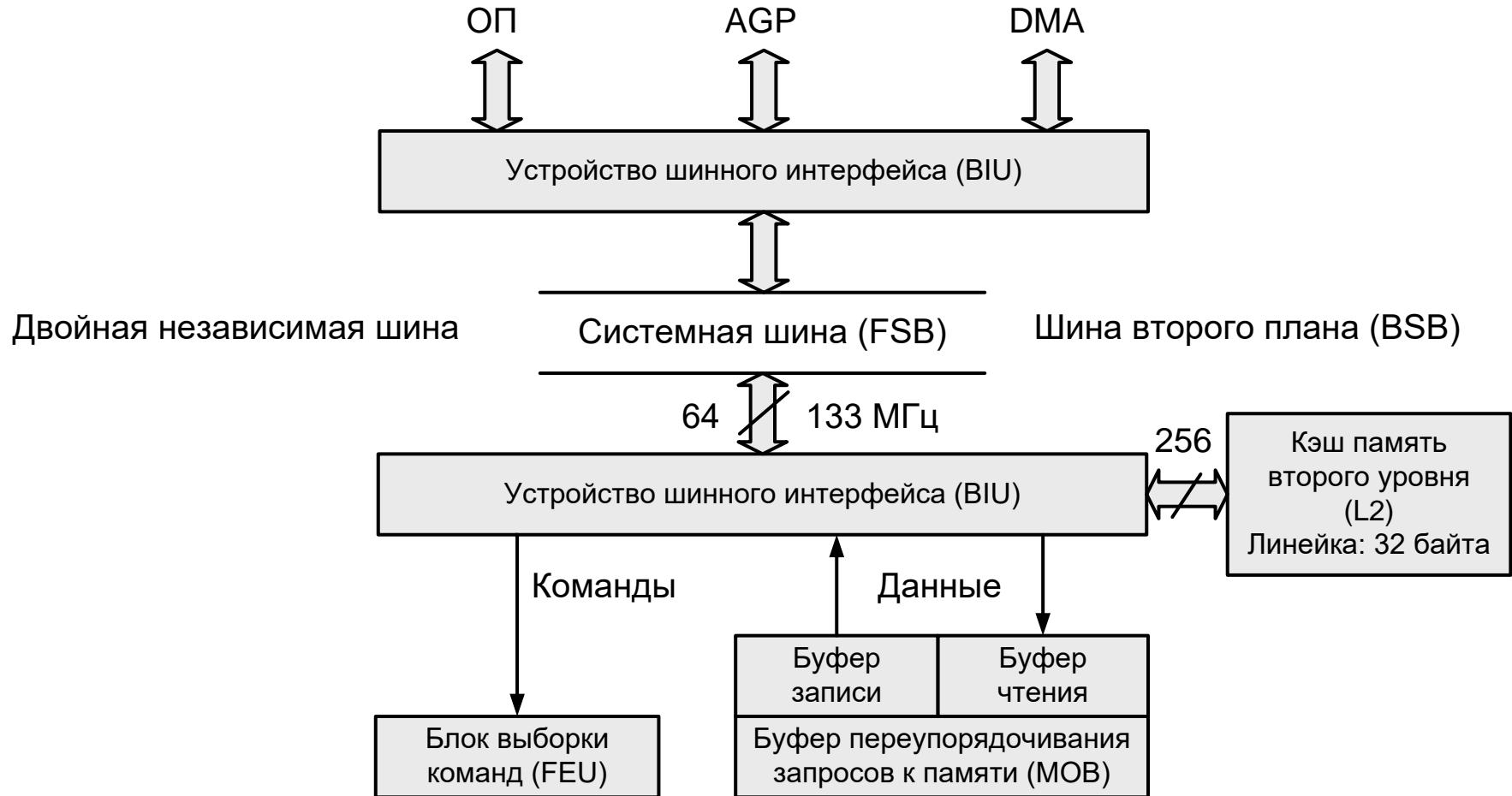
История процессоров с архитектурой IA32 (окончание)

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 Р6



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

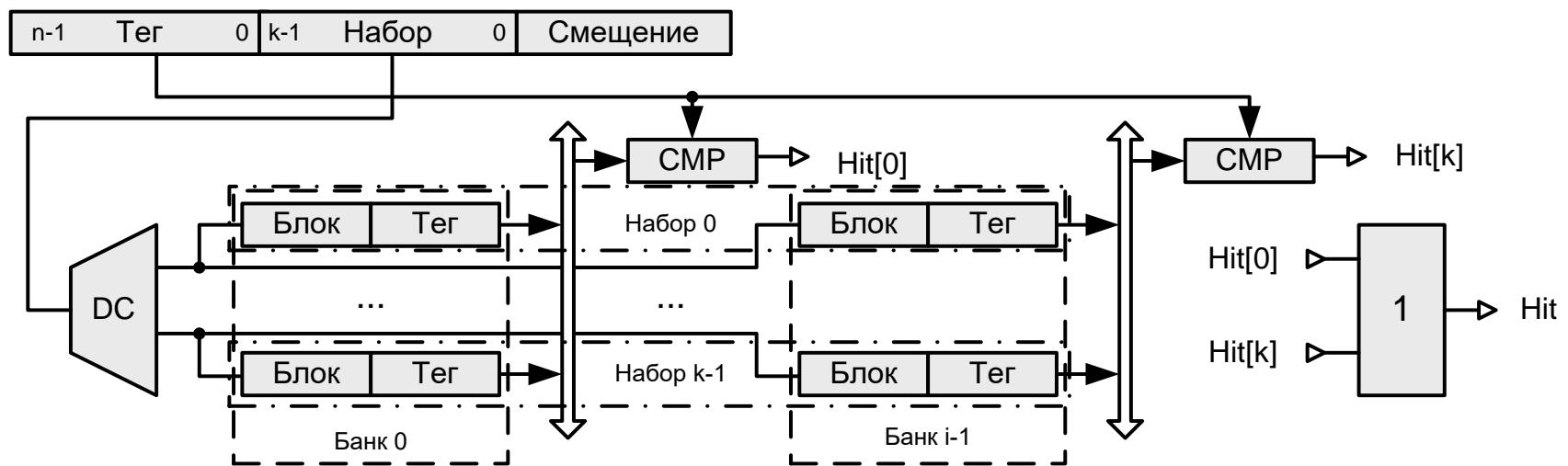
Размер линейки: 32 байта.

Ассоциативность: 4

Политика записи: Write Back

Алгоритм замещения: LRU

Протокол: MESI



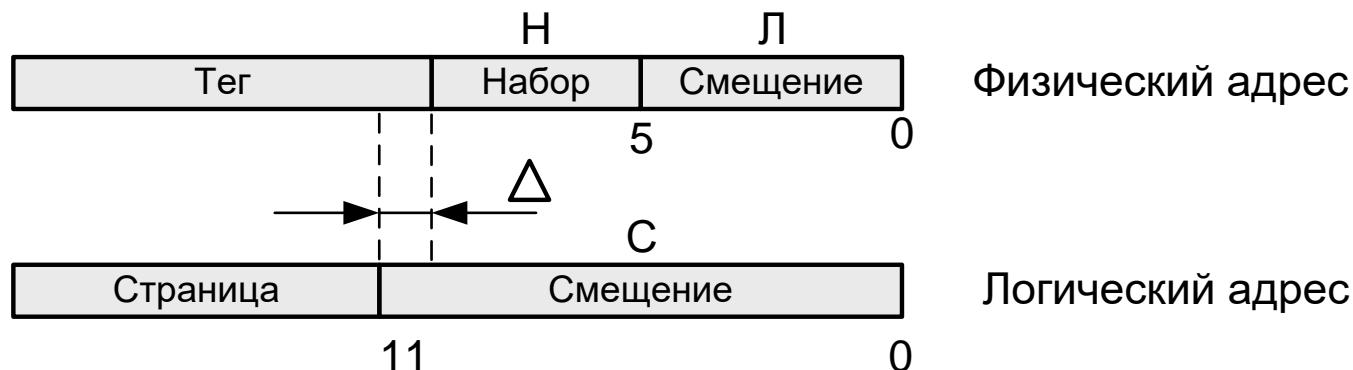
Проблема алиасинга

$$N \cdot L = (\text{Размер кэш}) / (\text{Ассоциативность})$$

При размещении в кэш данных по адресам с шагом $N \cdot L$ 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.

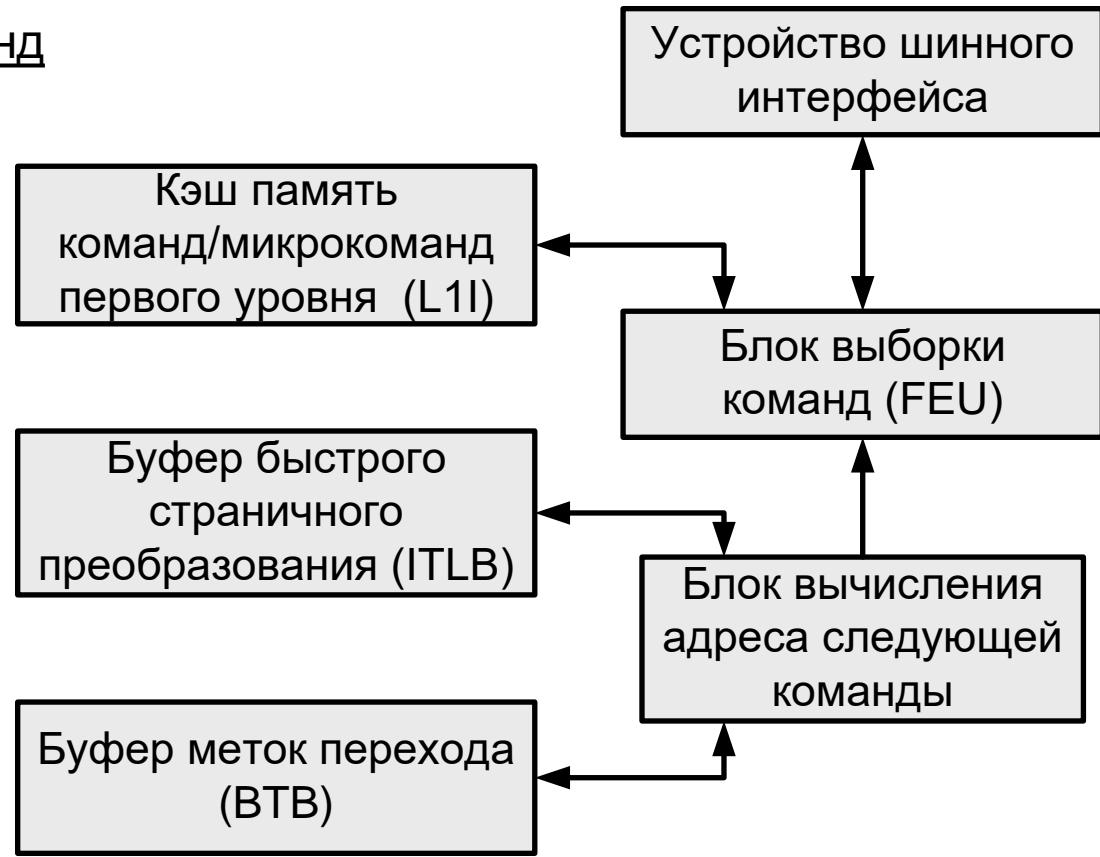


При обнаружении логического адреса в команде с учетом того, что ($C >= N \cdot L$) уже известен набор, а возможно и часть тега.

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

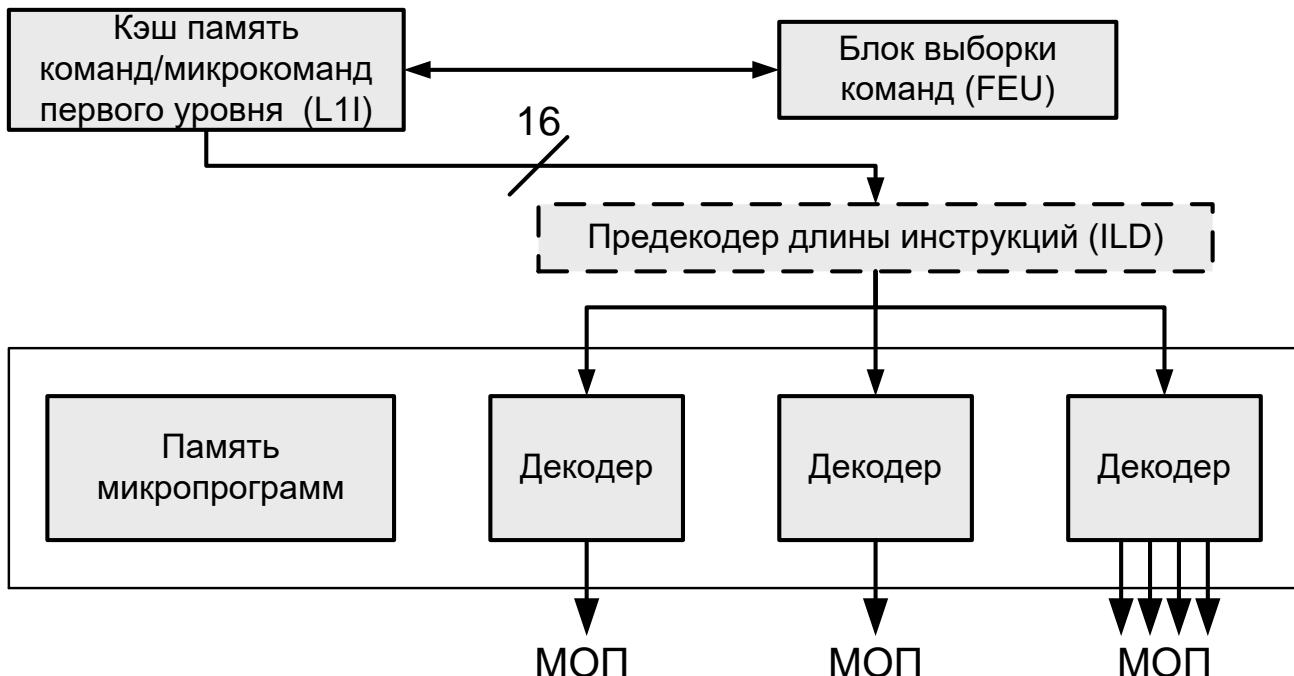
Информация, записываемая в TLB, не подлежит кэшированию.

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

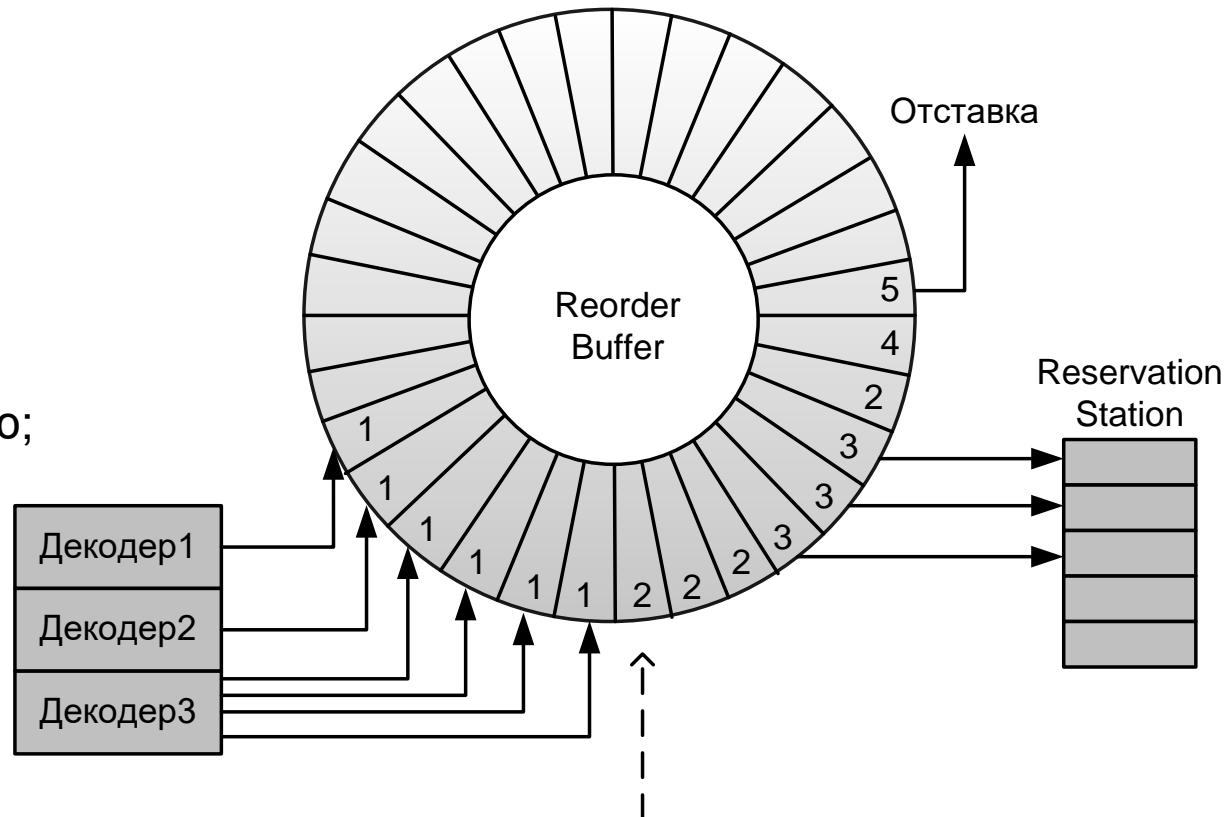
Буфер переупорядочивания микрокоманд

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

Микрокоманды в ROB

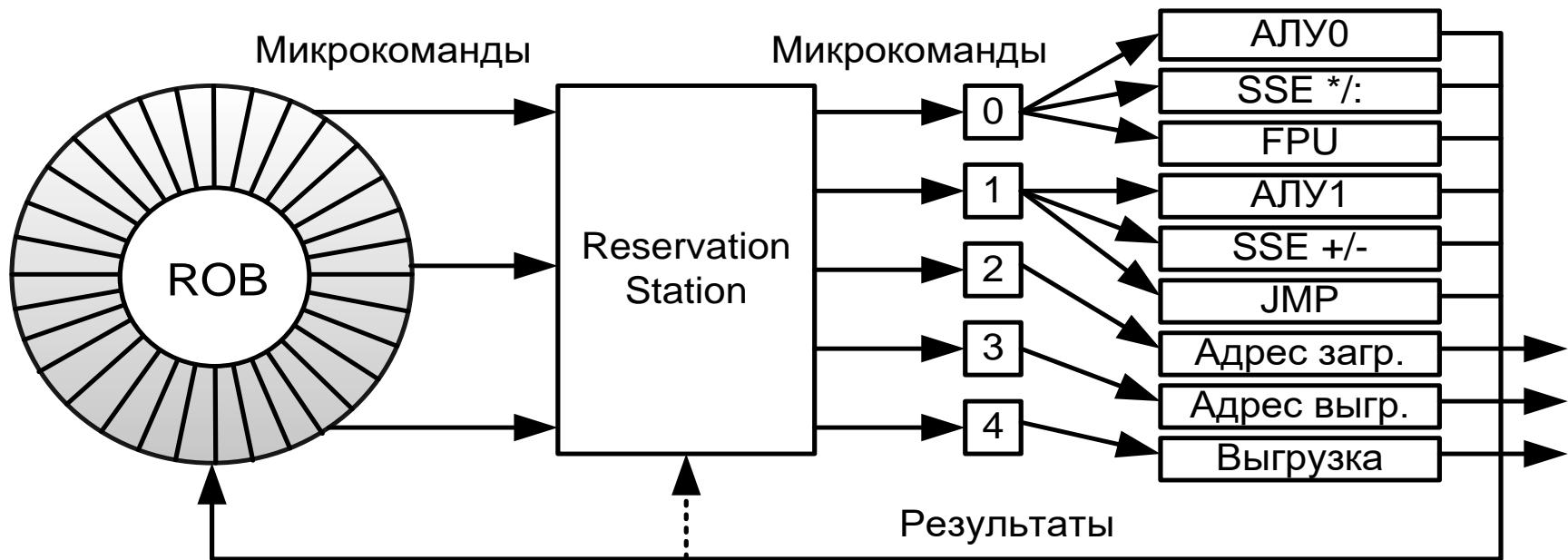
могут находиться в
одном из следующих
состояний:

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 ВИУ.

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 Р6

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

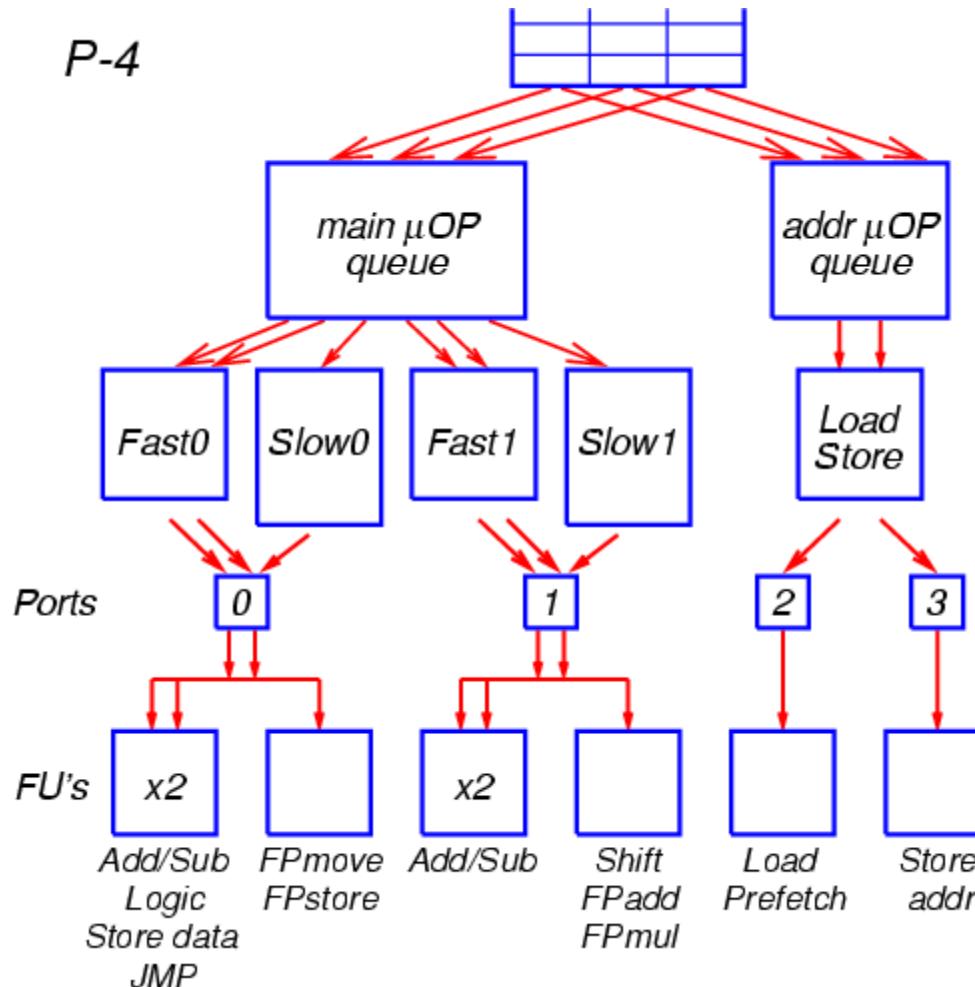
Недостатки Р6

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

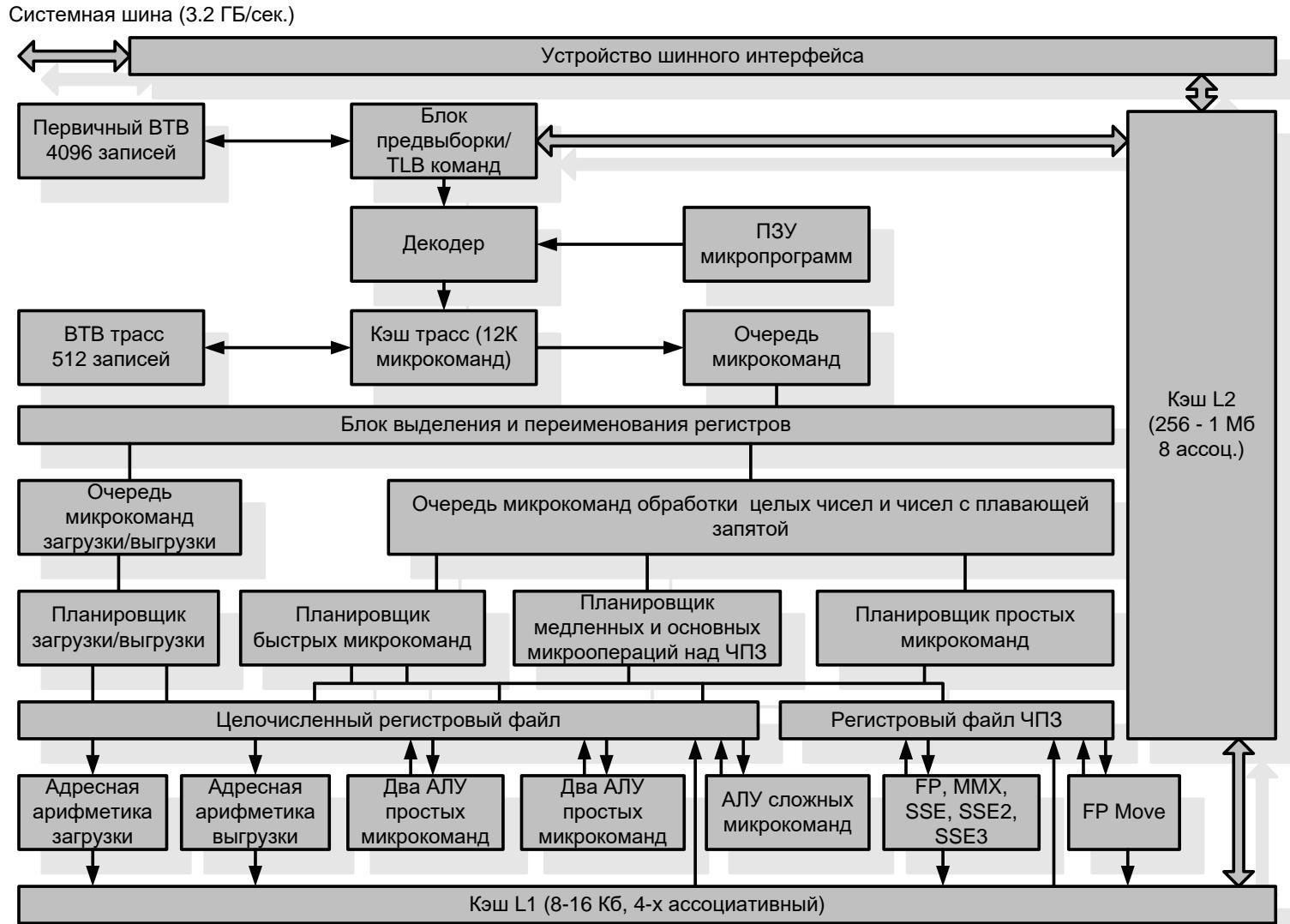
Отличие архитектуры NetBurst от P6

- 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 TBWB 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



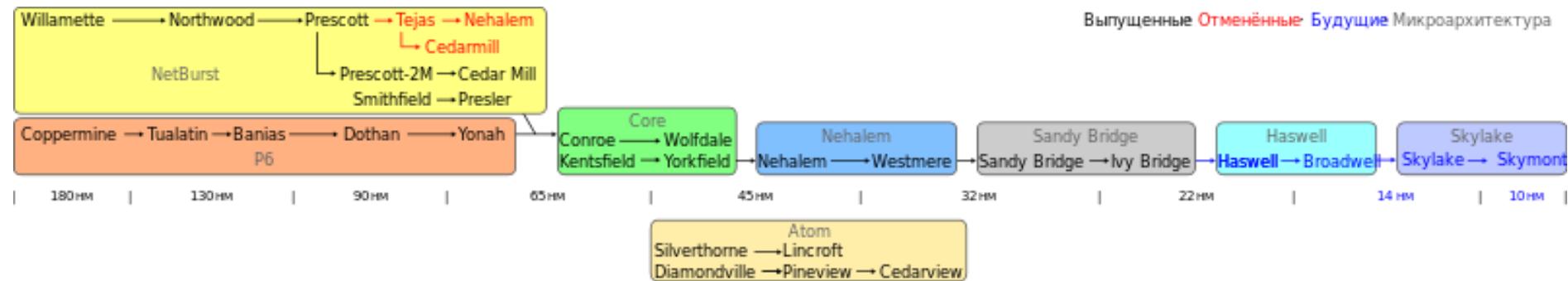
Особенности микроархитектуры NetBurst



Конвейеры микропроцессоров Intel

Микроархитектура	Количество стадий конвейера
<u>486</u> (80486)	3
<u>P5</u> (Pentium)	5
<u>P6</u> (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)
<u>NetBurst</u> (Willamette)	20
NetBurst (Northwood)	20
NetBurst (Prescott)	31
NetBurst (Cedar Mill)	31
<u>Core</u> (Merom/Conroe/Woodcrest)	12 (14 с выборкой и отставкой)
<u>Nehalem</u>	20
<u>Sandy Bridge</u>	14 (16 с выборкой и отставкой)

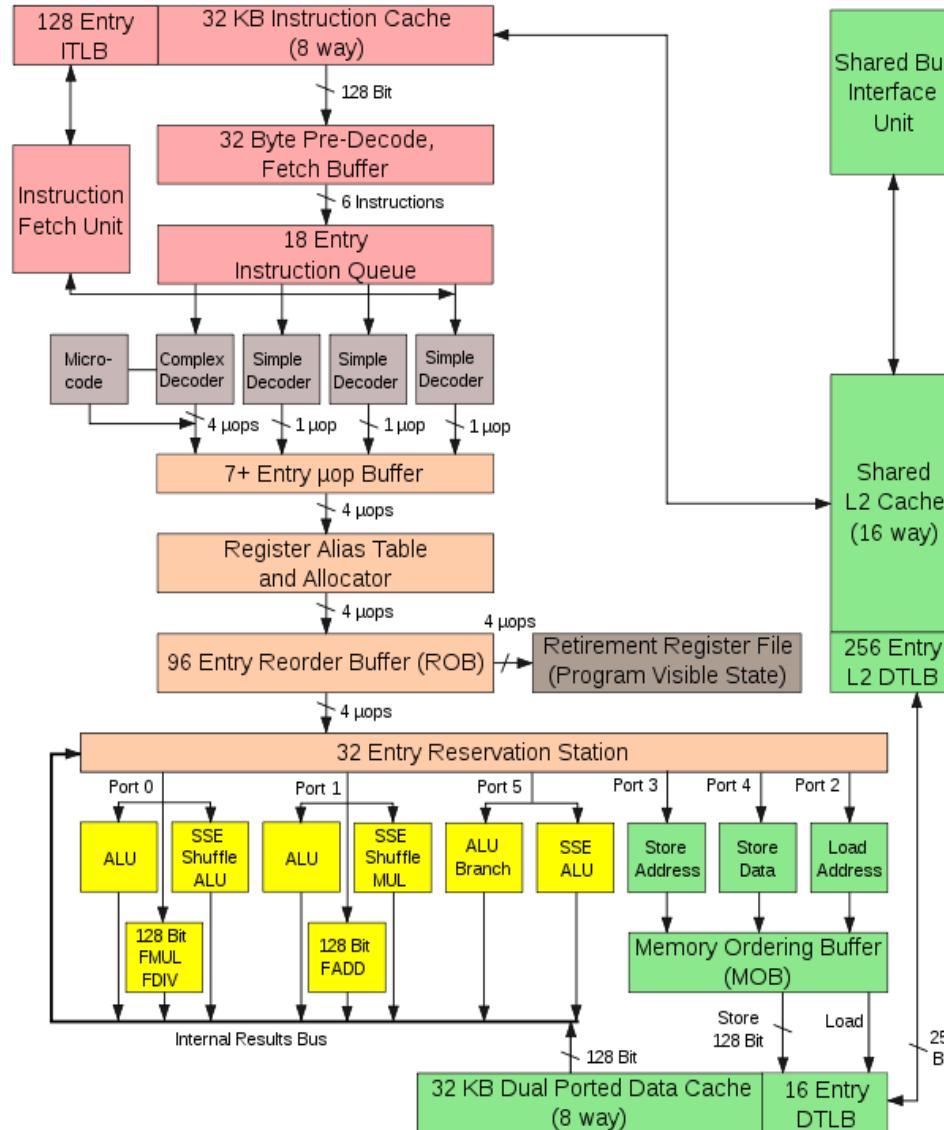
Микроархитектуры процессоров Intel



<http://www.ixbt.com/cpu/sandy-bridge.shtml>

https://en.wikipedia.org/wiki/List_of_Intel_CPU_microarchitectures https://ru.wikipedia.org/wiki/Sandy_Bridge

Микроархитектура Core (P6+)

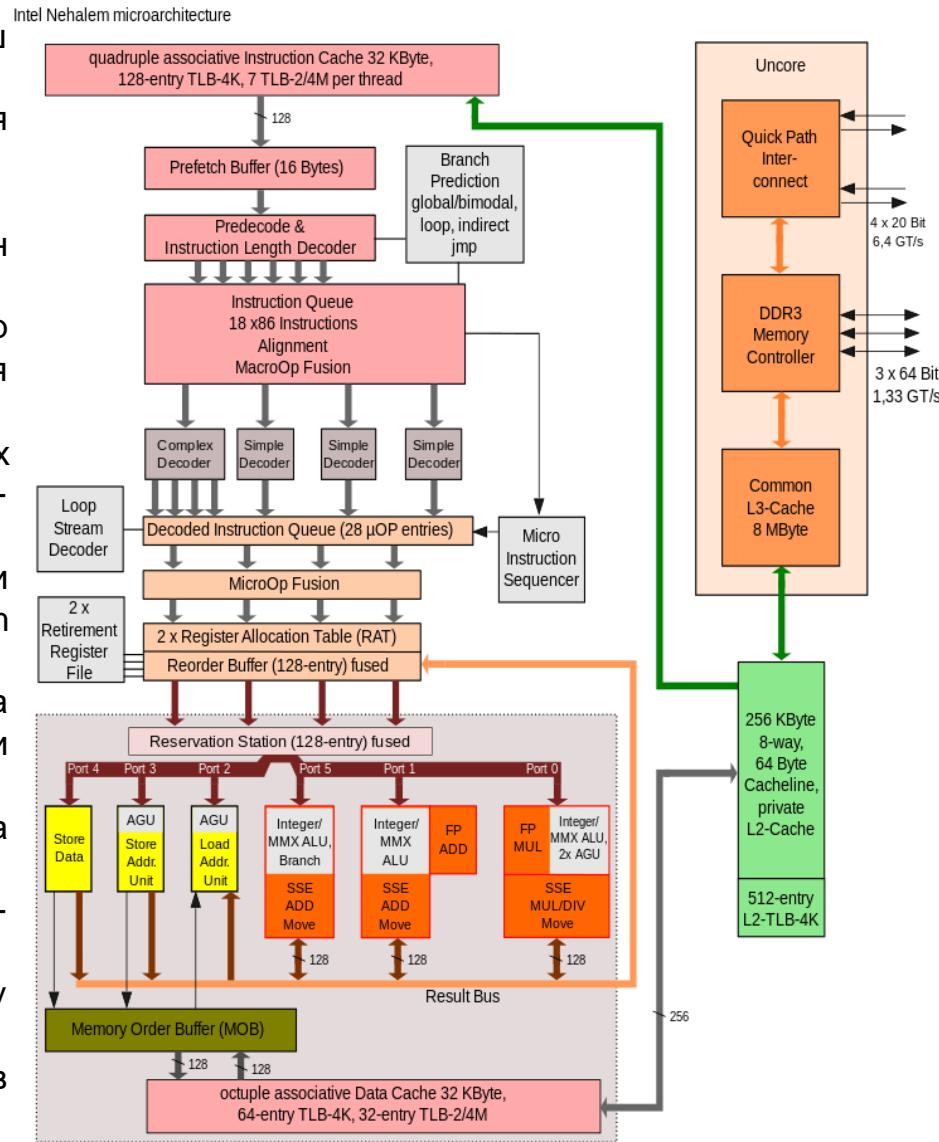


Intel Core 2 Architecture

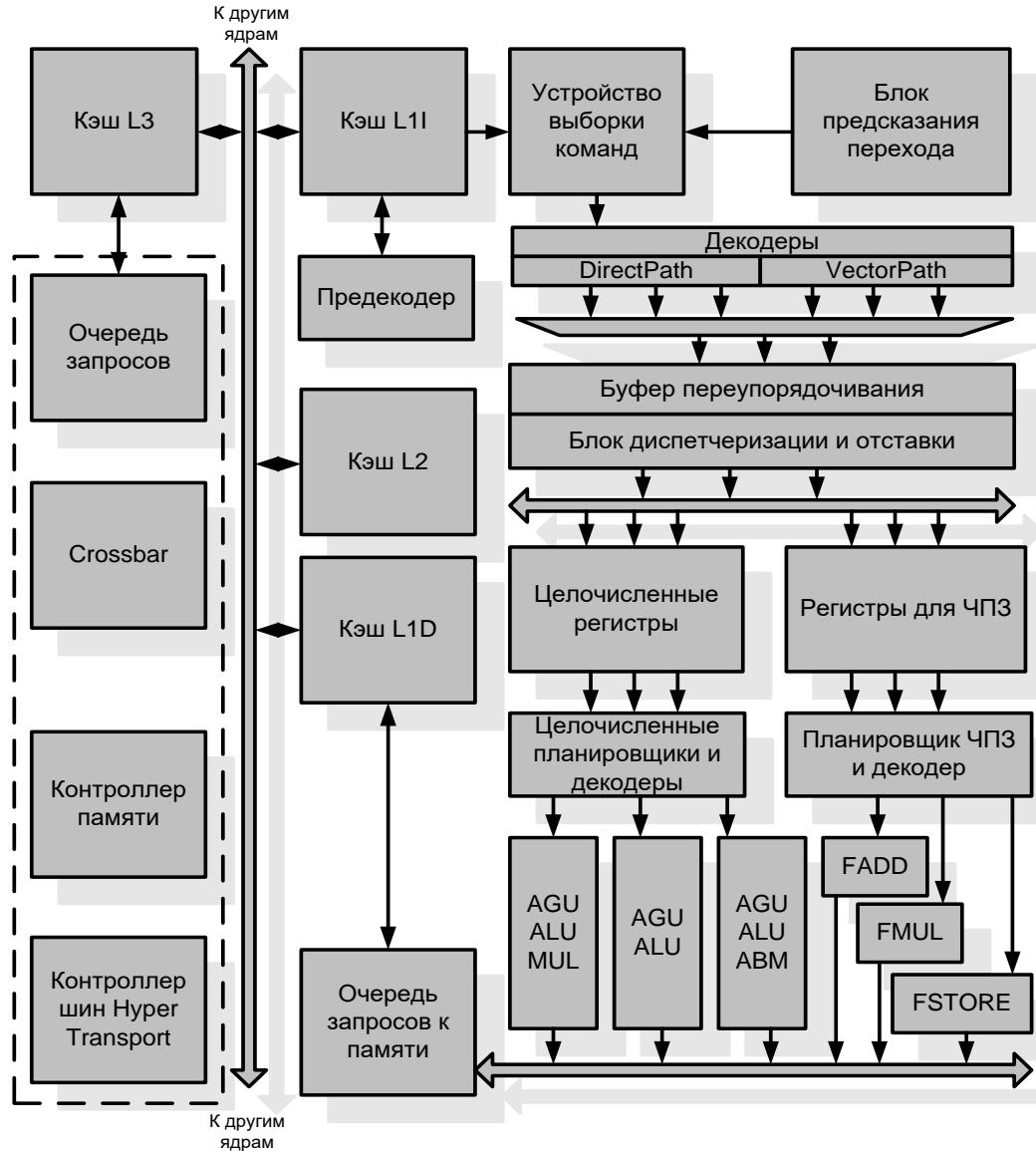
Микроархитектура Nehalem

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 (micro 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

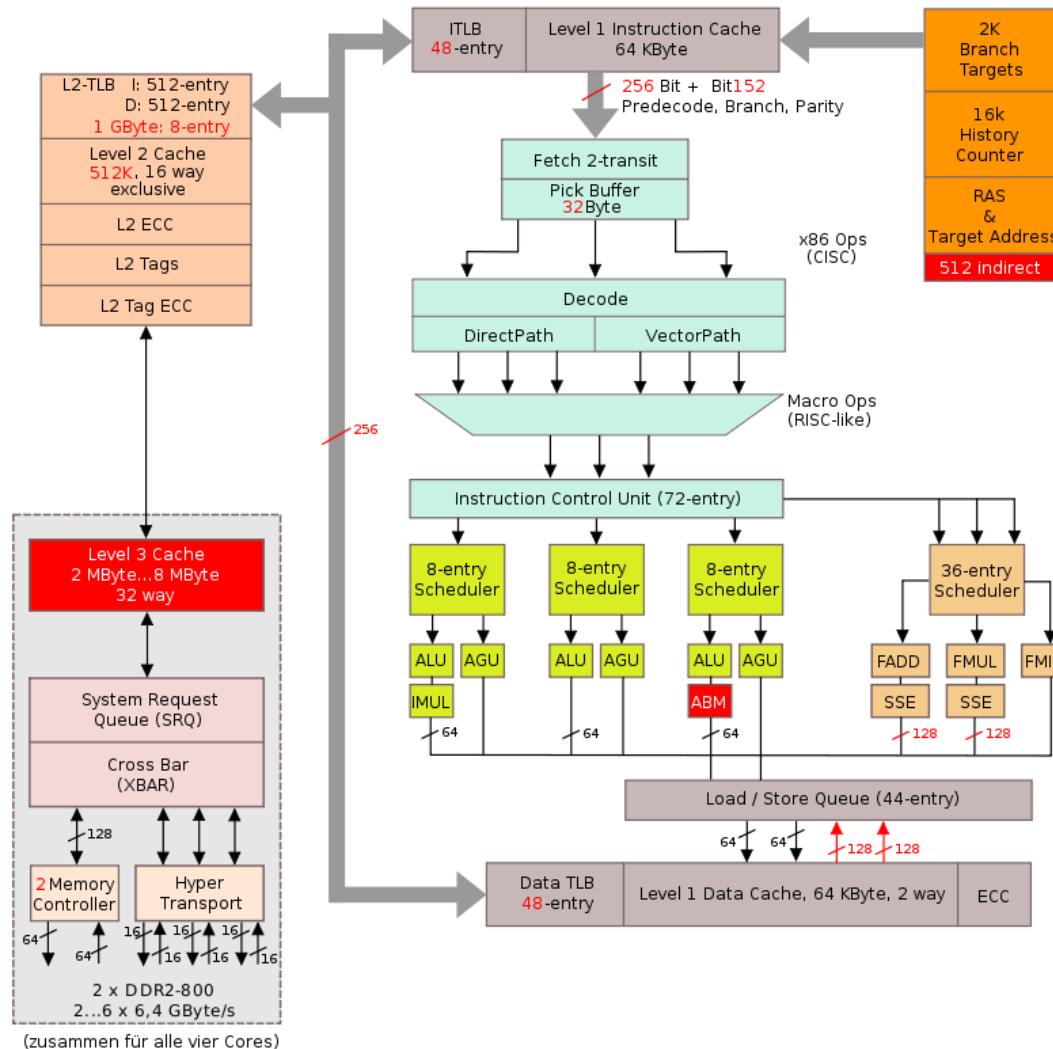


Микроархитектура AMD K10

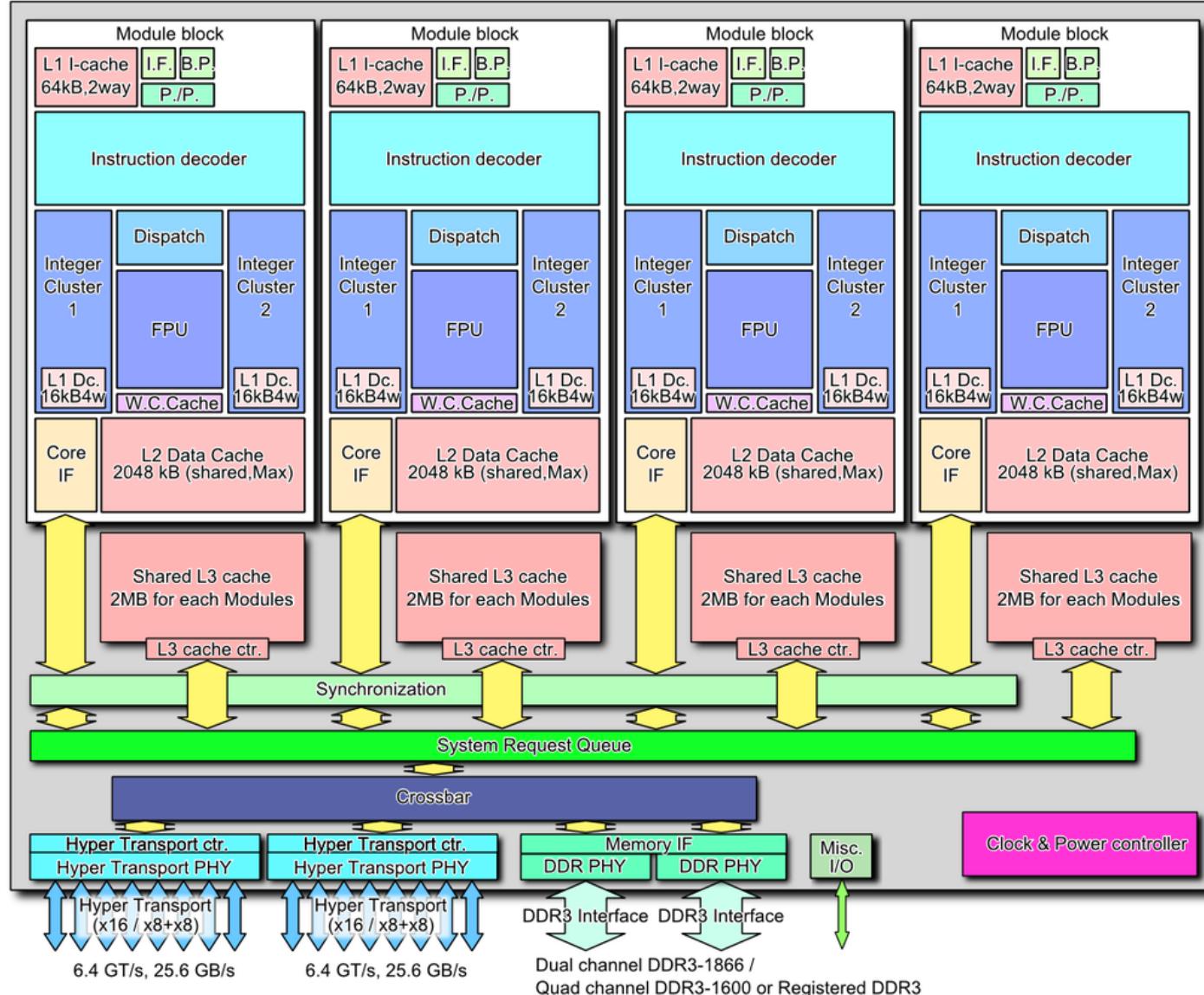
AMD K10 Architecture

Red: Difference between K8 and K10 Architecture

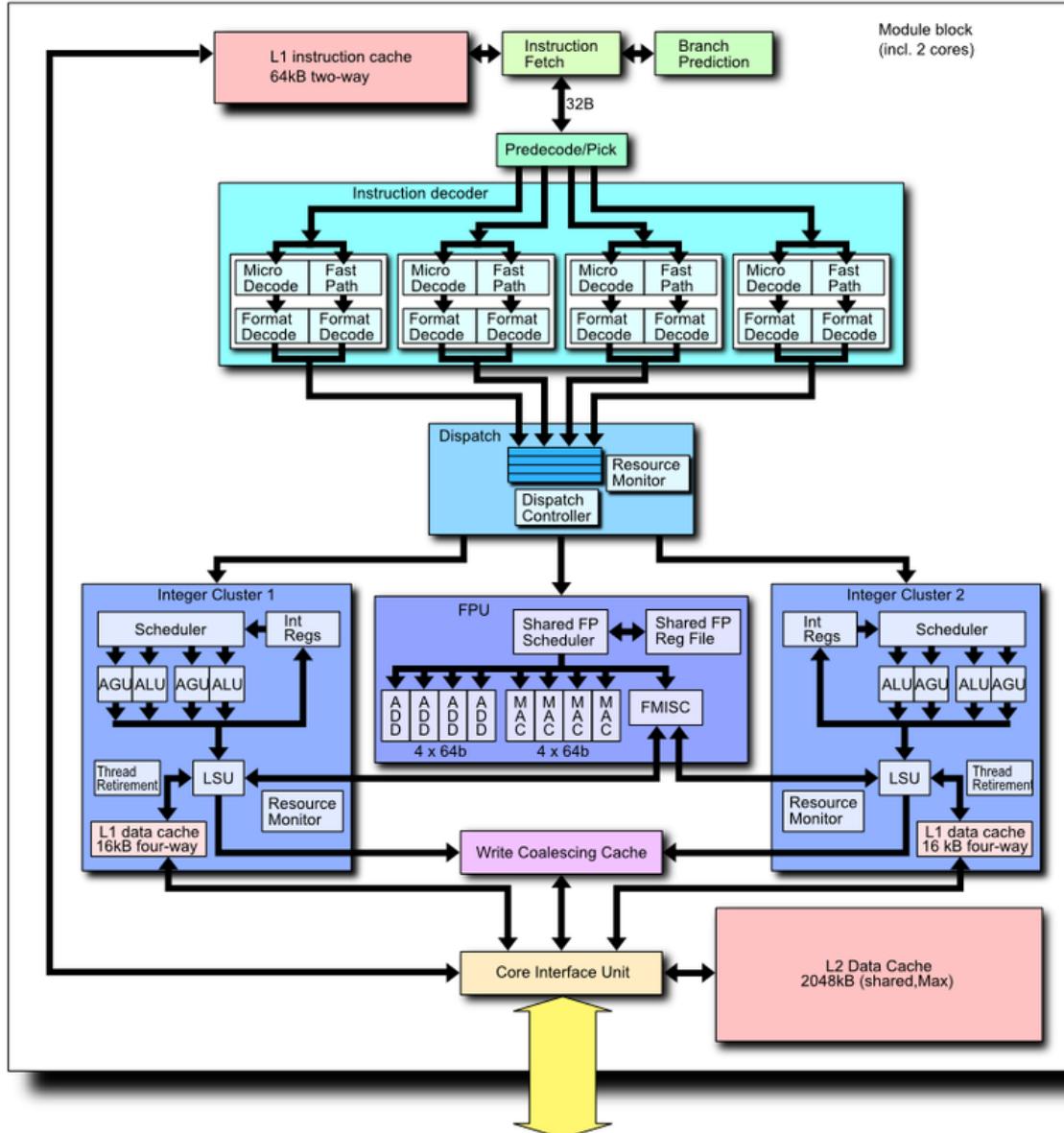
(Die Änderungen zwischen der K8- und K10-Architektur sind rot markiert)



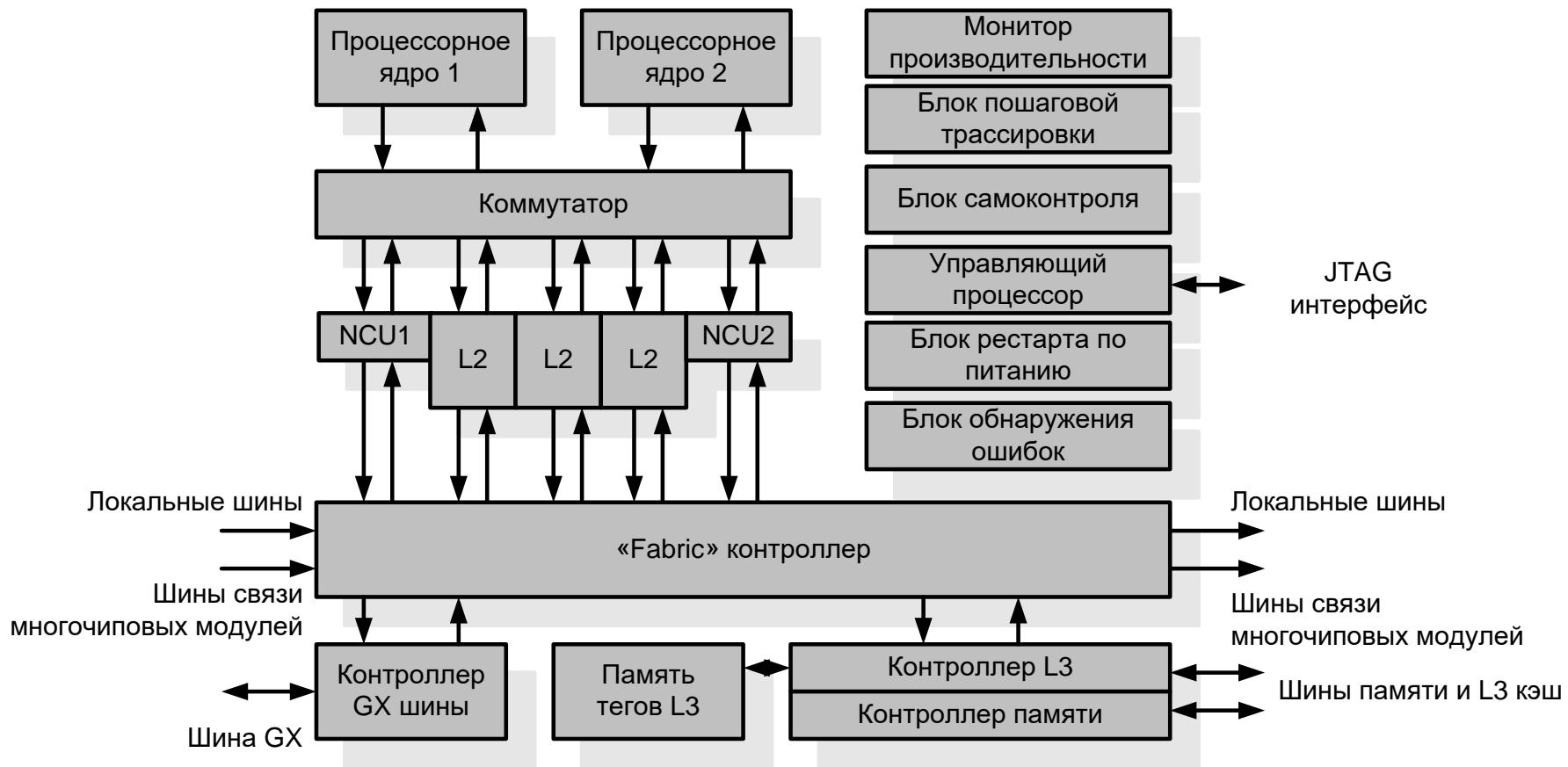
Многоядерный чип AMD Bulldozer



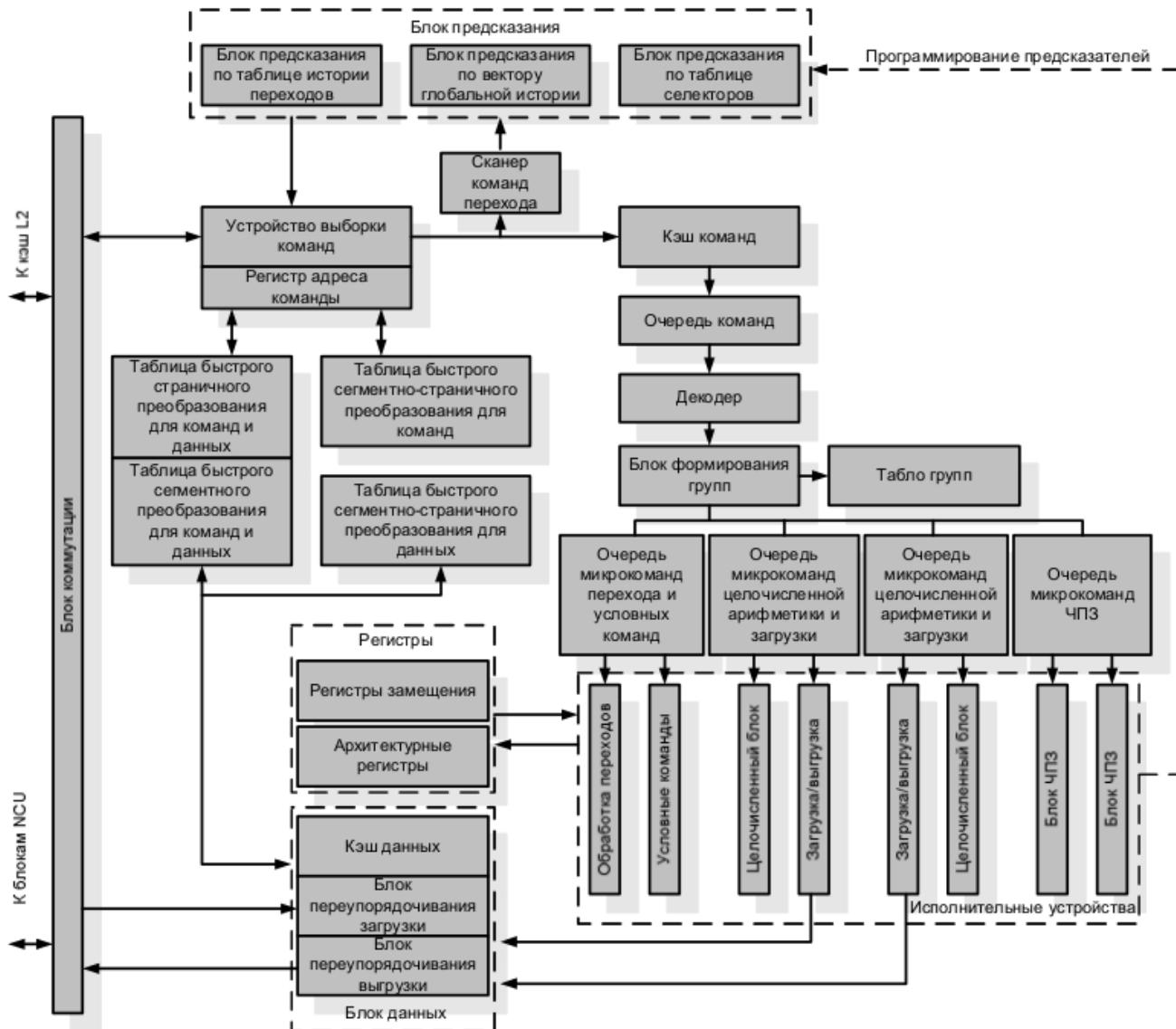
Микроархитектура ядра AMD Bulldozer



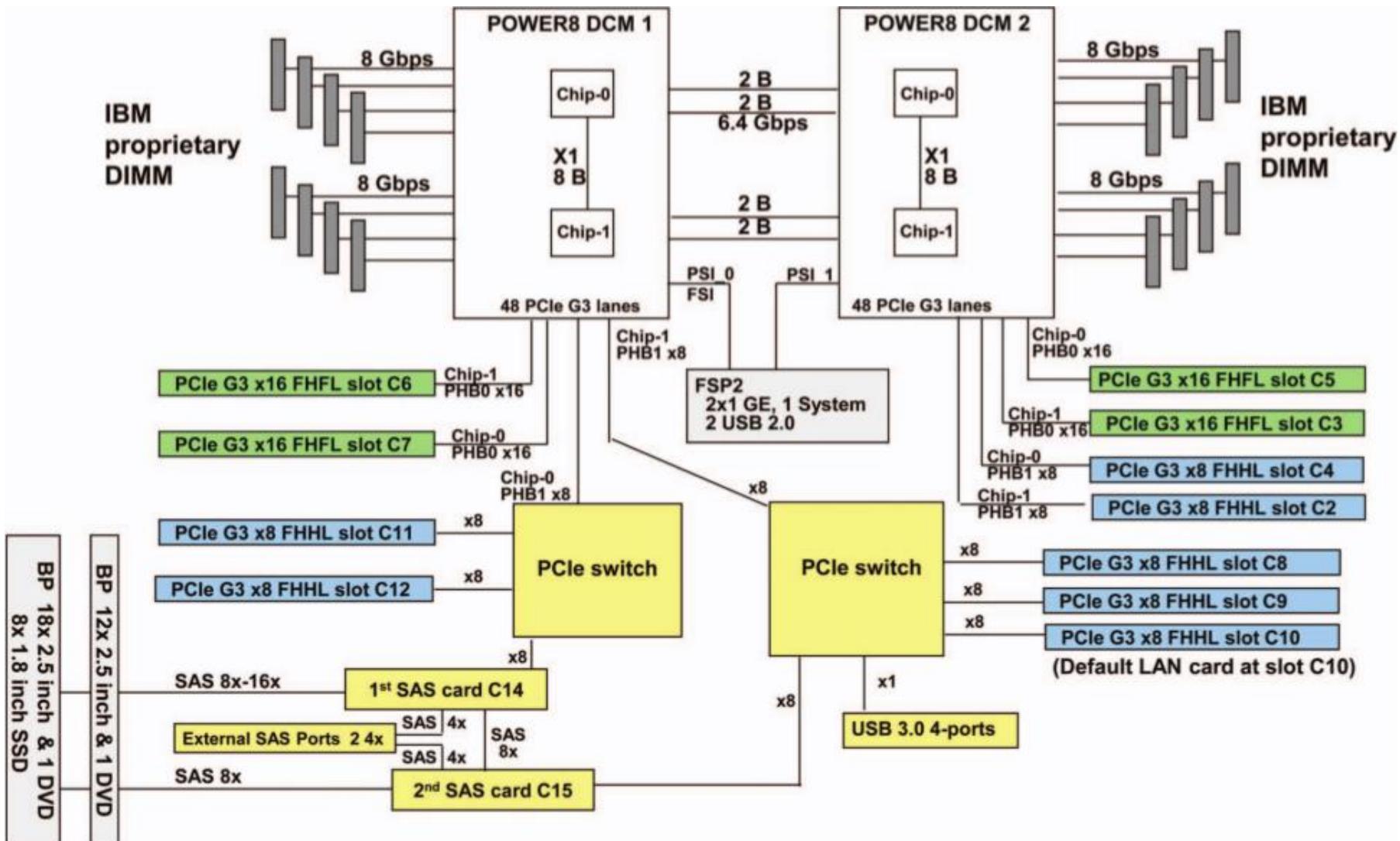
Микроархитектура суперскалярных процессоров IBM POWER4



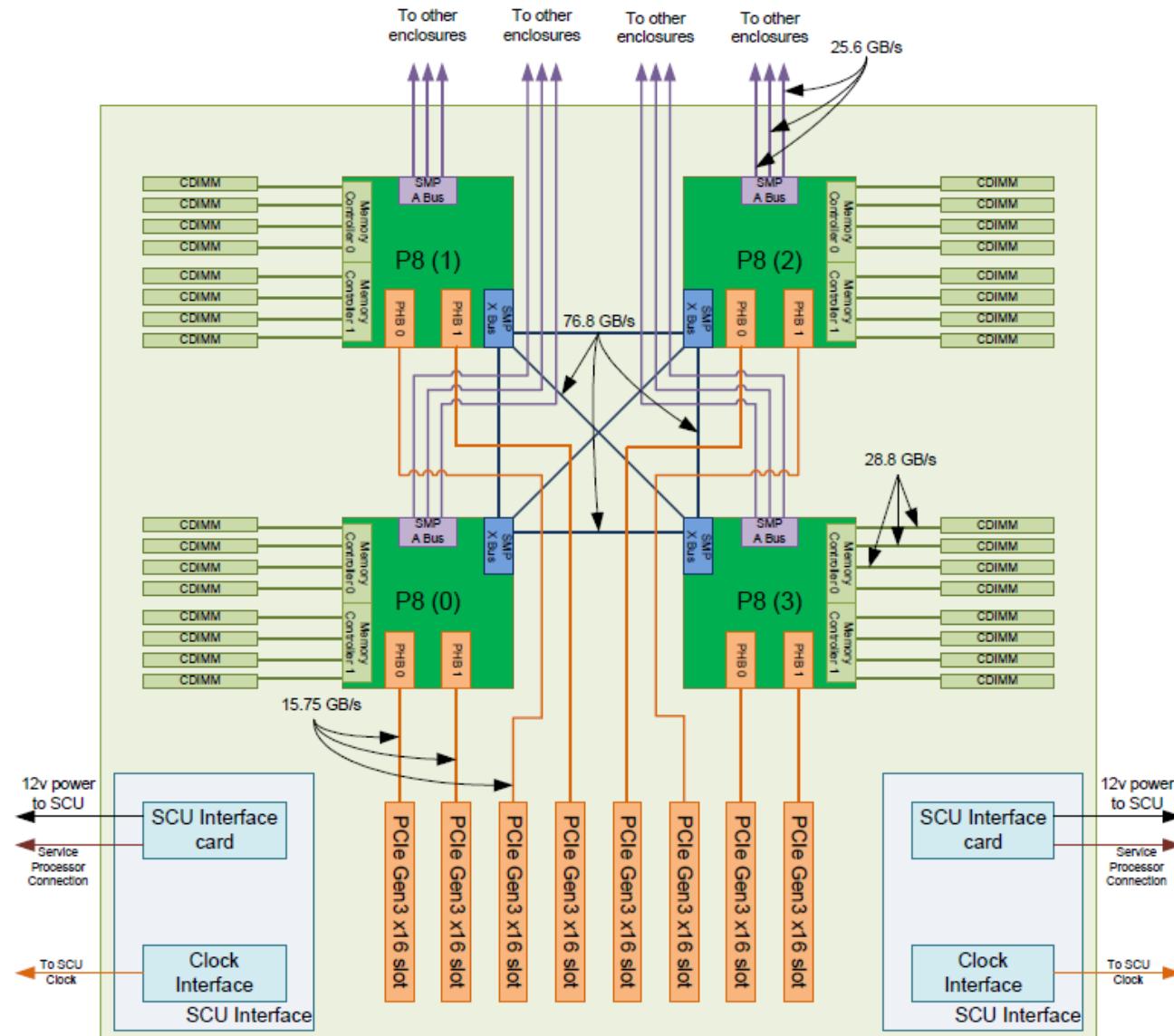
Структура процессорного ядра IBM POWER4



Серверная платформа IBM POWER8

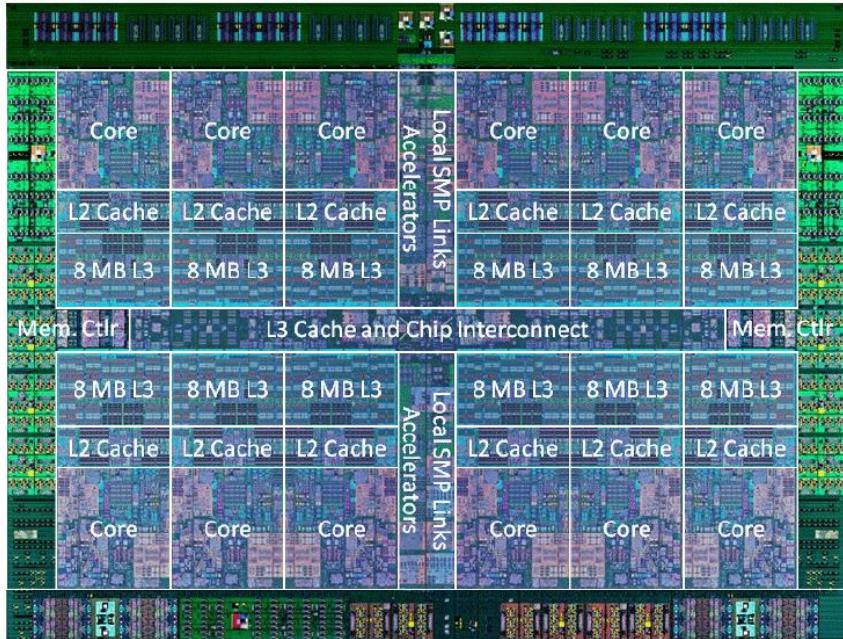


Вычислительный узел серверной платформы IBM POWER8

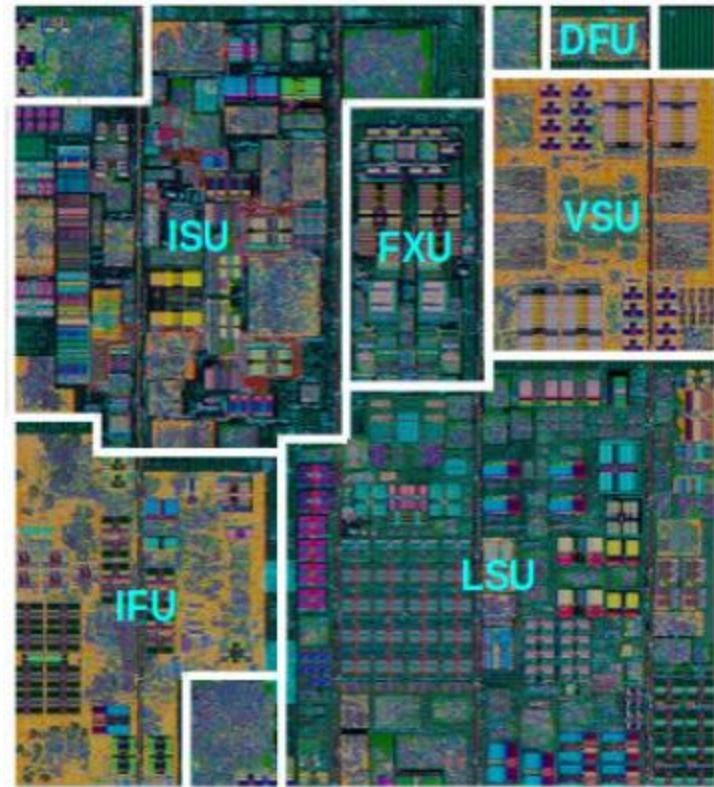


Структура процессорного ядра IBM POWER8

Микросхема POWER8

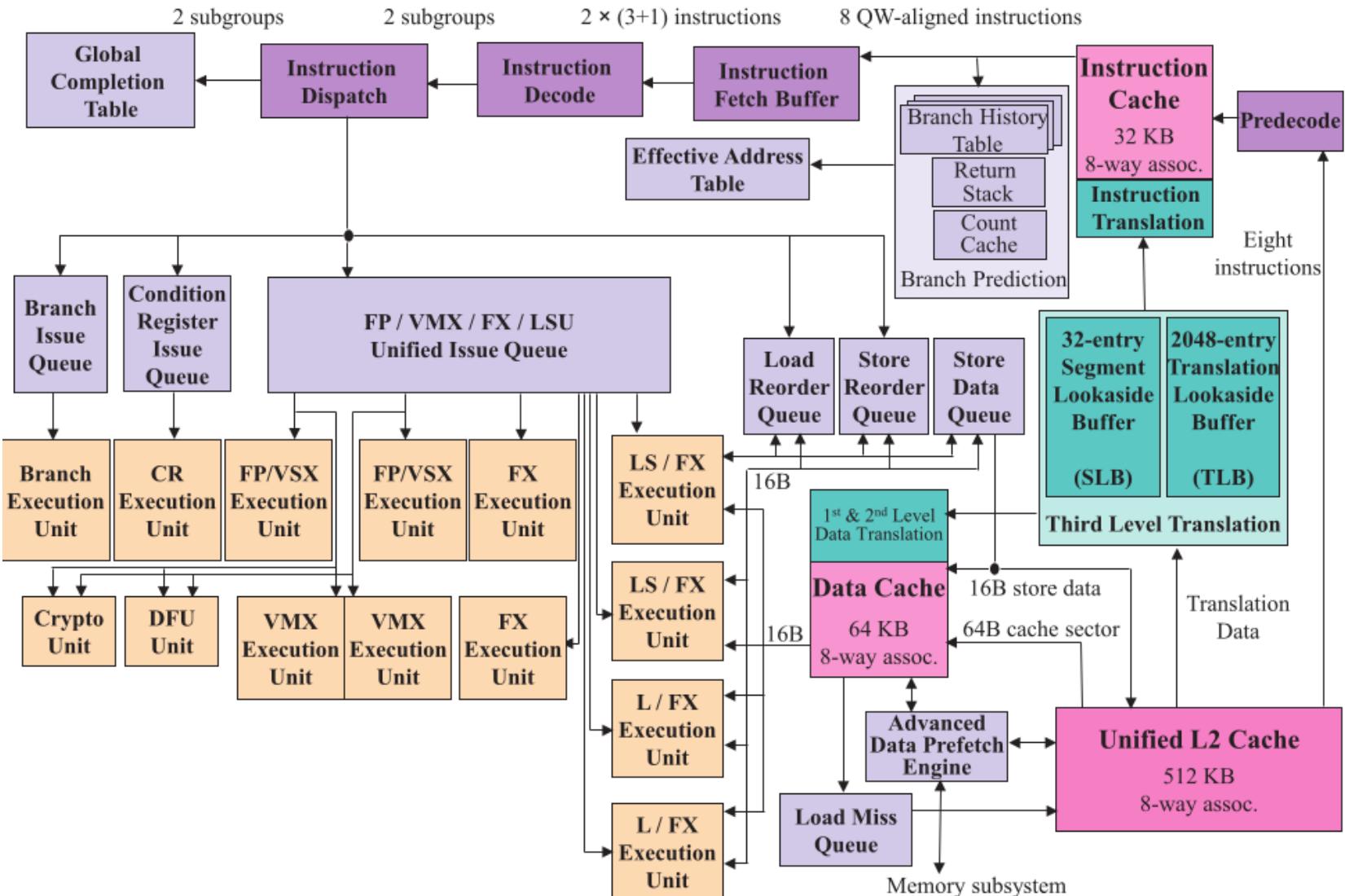


Микропроцессорное ядро

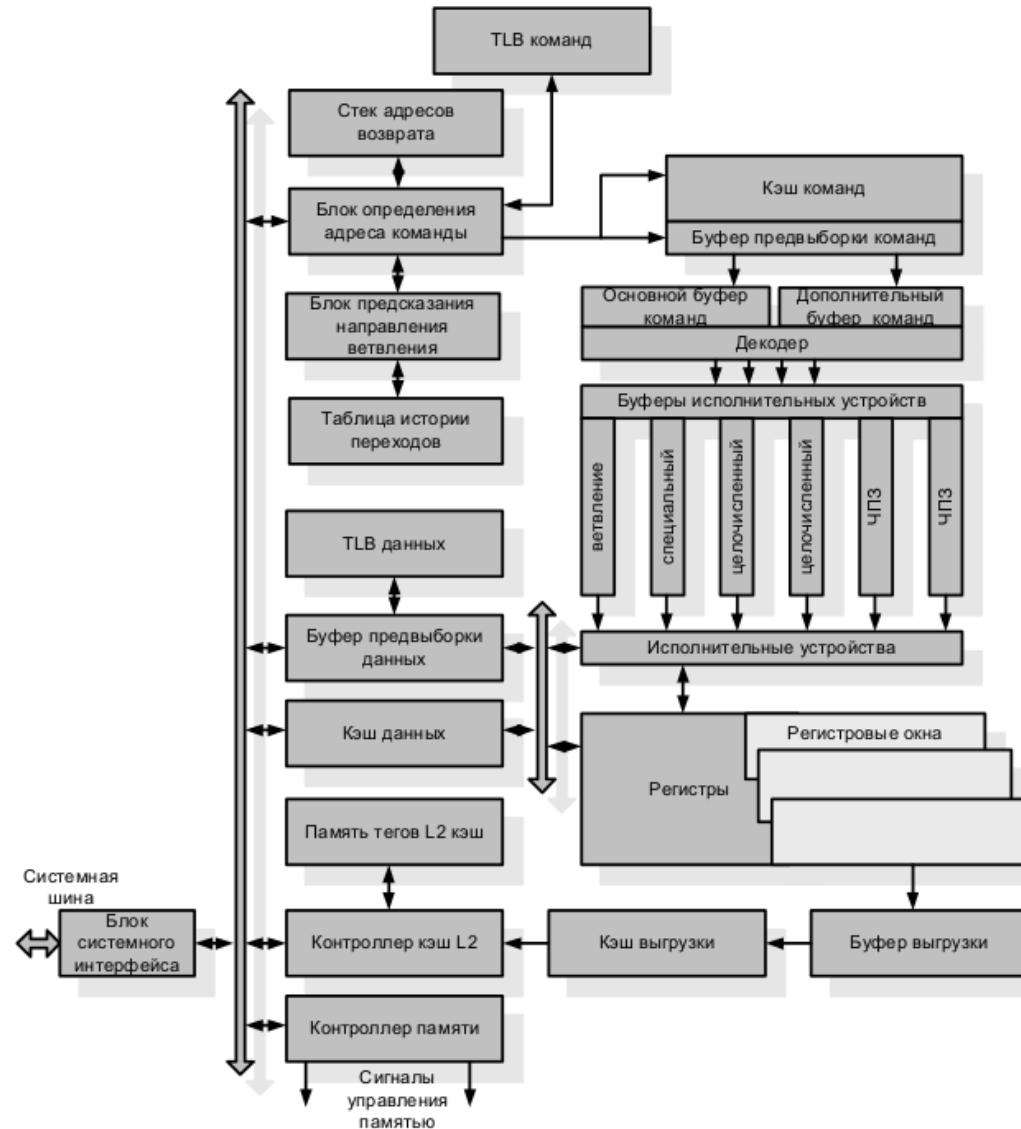


Структура процессорного ядра IBM POWER8

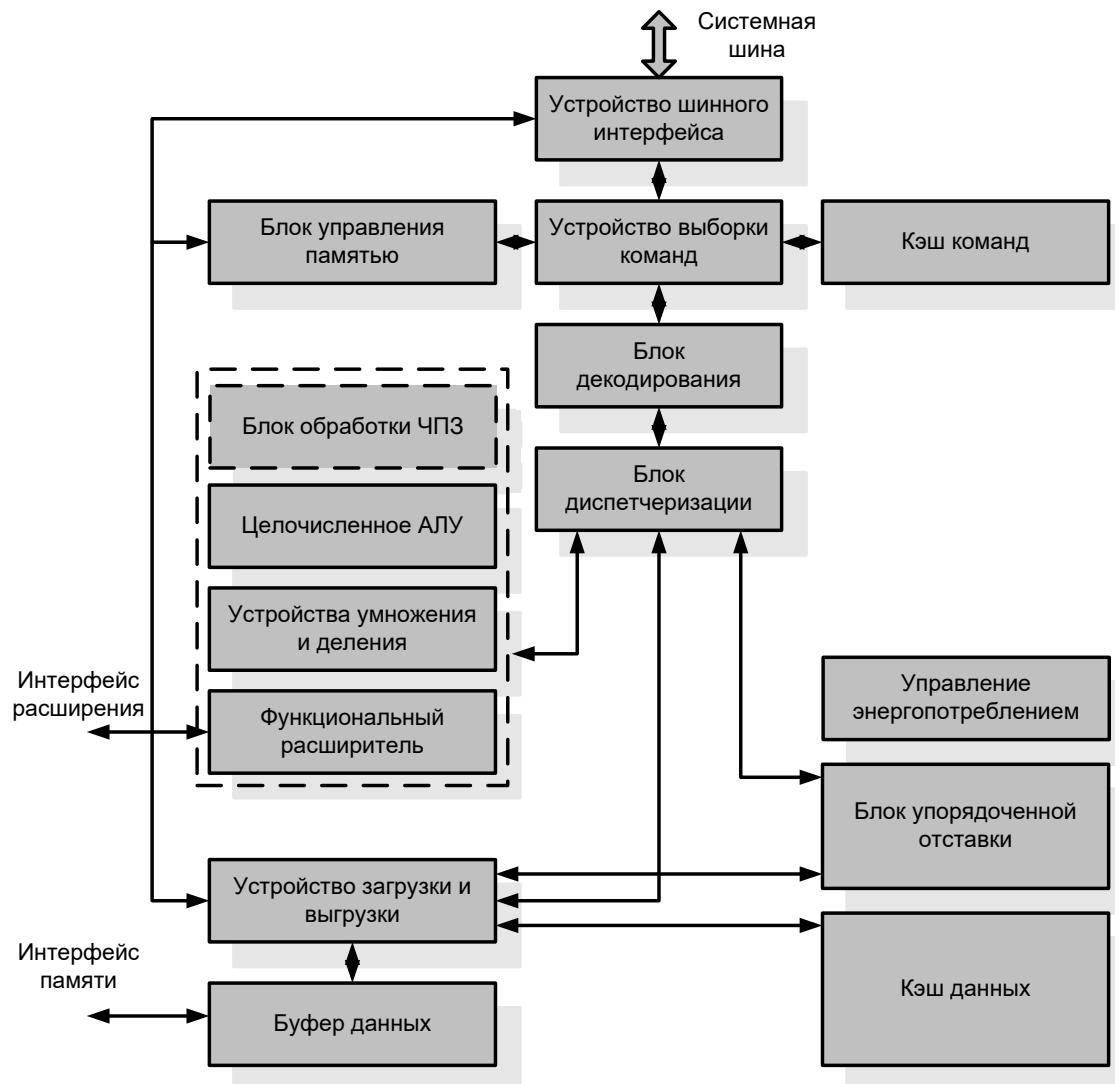
Конвейер



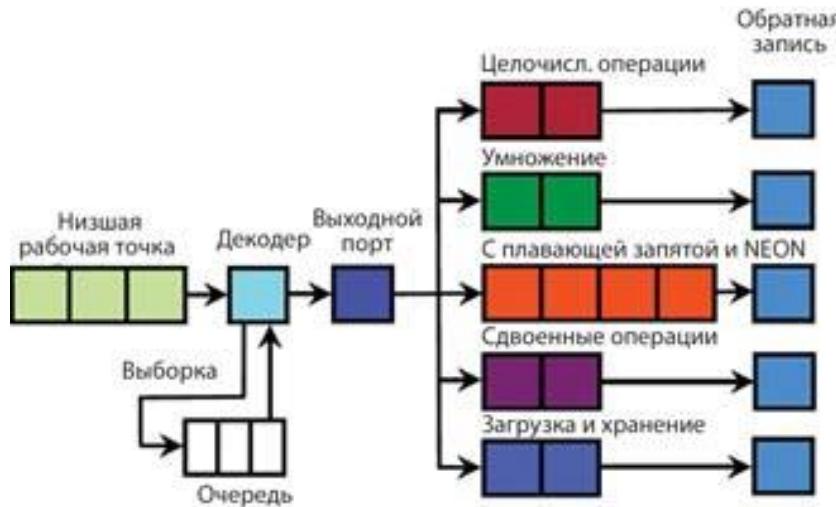
Микроархитектура суперскалярных процессоров Sun UltraSPARC III



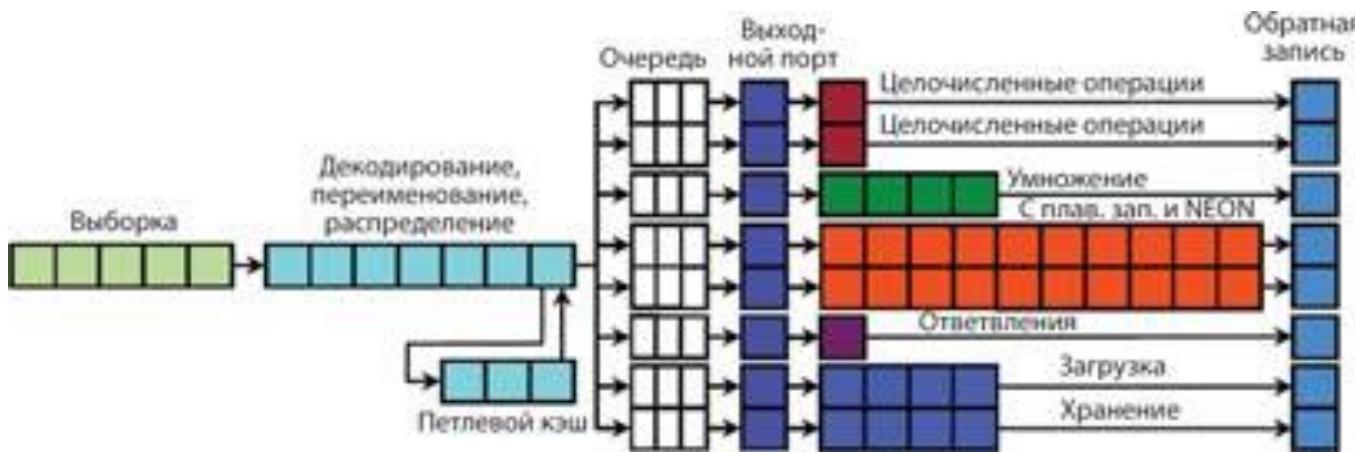
Архитектура синтезируемых суперскалярных процессорных ядер MIPS32 74K



Микроархитектура Cortex-A7

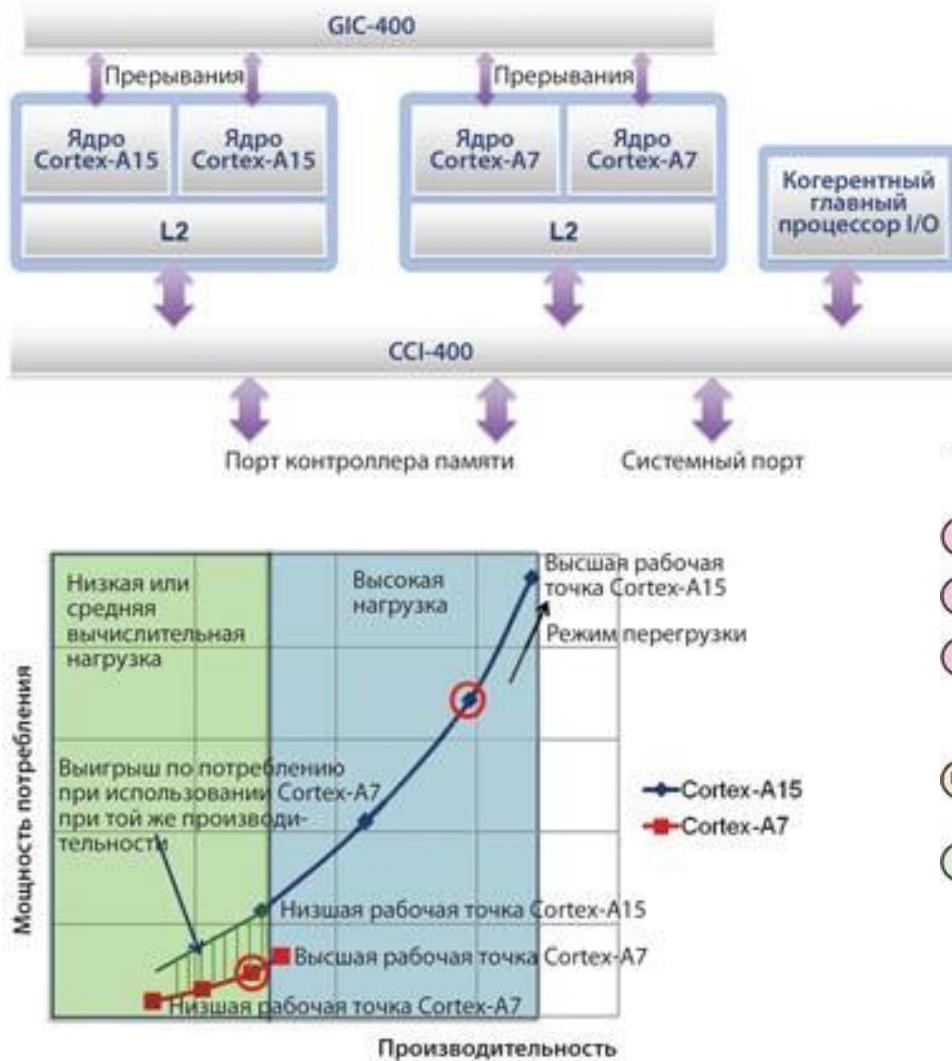


Микроархитектура Cortex-A15

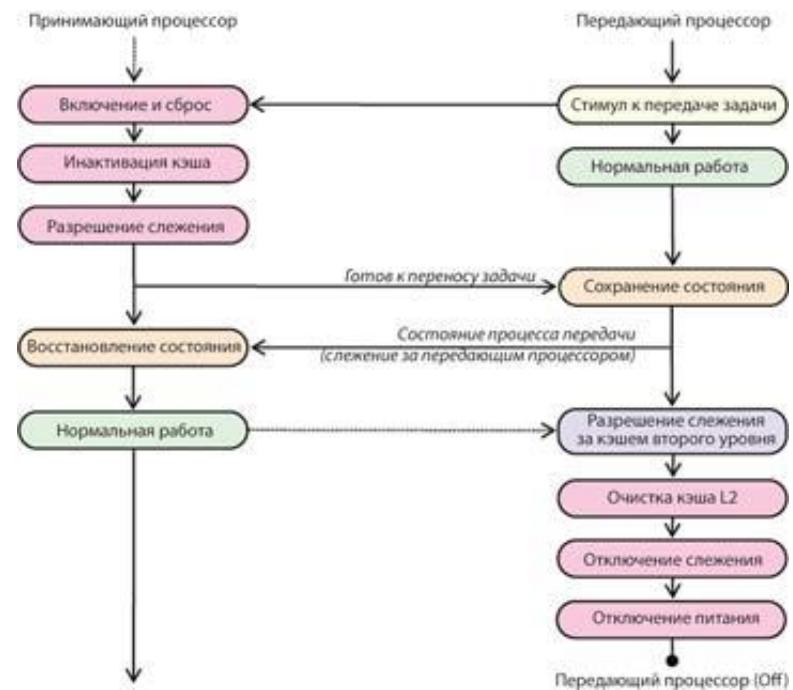


Greenhalgh P. Big.LITTLE processing with ARM Cortex-A15 & Cortex-A7//www.eetimes.com.

big.LITTLE



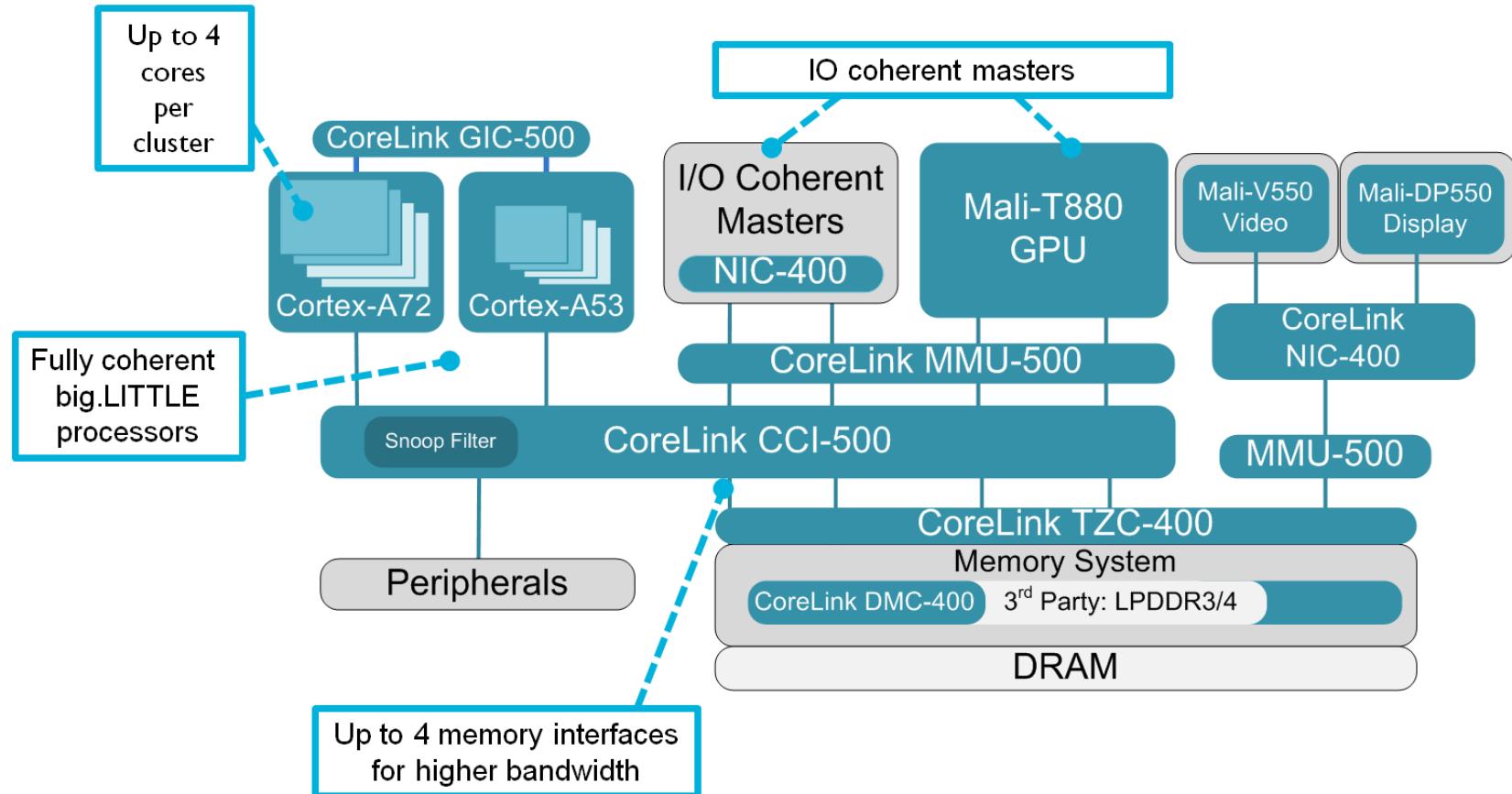
Одним из ключевых элементов является система межсоединений CCI-400, которая обеспечивает полную согласованность между Cortex-A15 и Cortex-A7, а также устройствами ввода-вывода. Контроллер прерываний GIC-400 способен распределить до 480 прерываний, причем прерывания могут направляться в любое ядро в кластере 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



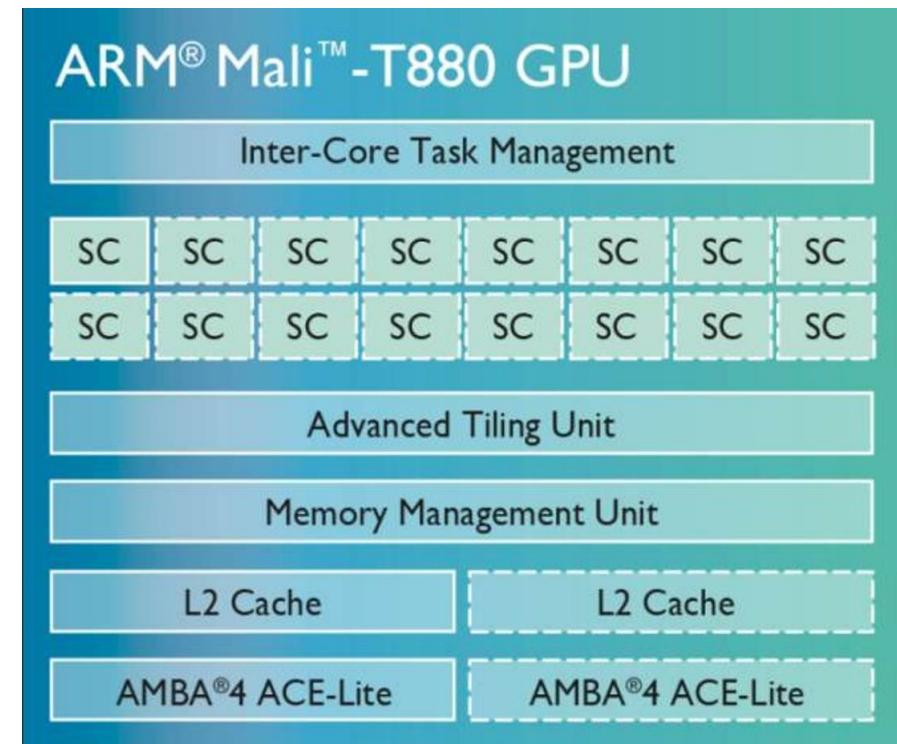
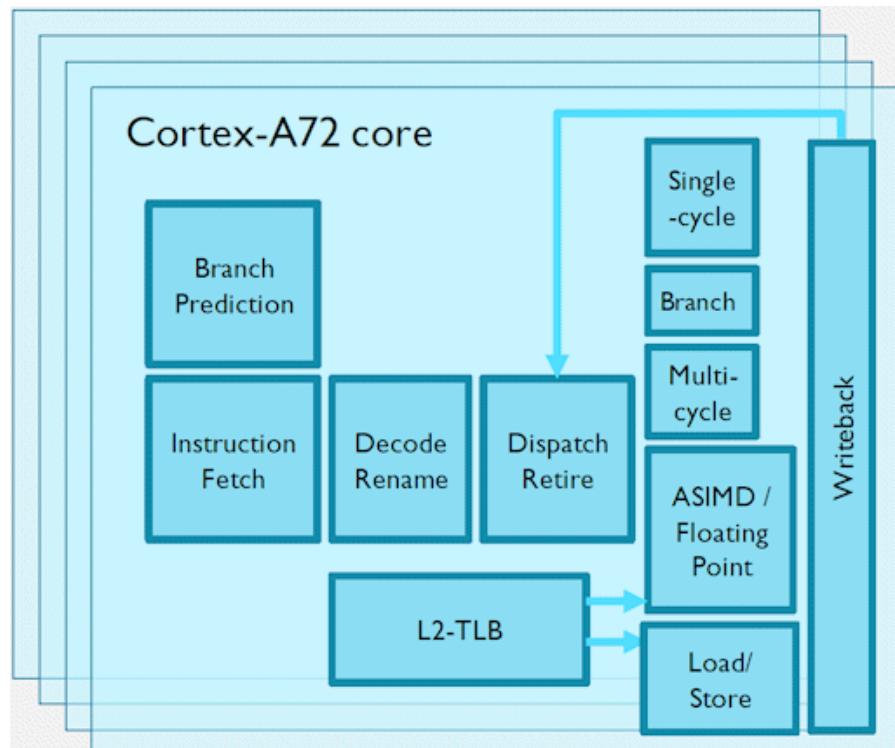
<http://www.russianelectronics.ru/leader-r/review/2192/doc/58808/>

Архитектура системы на кристалле Cortex-A72

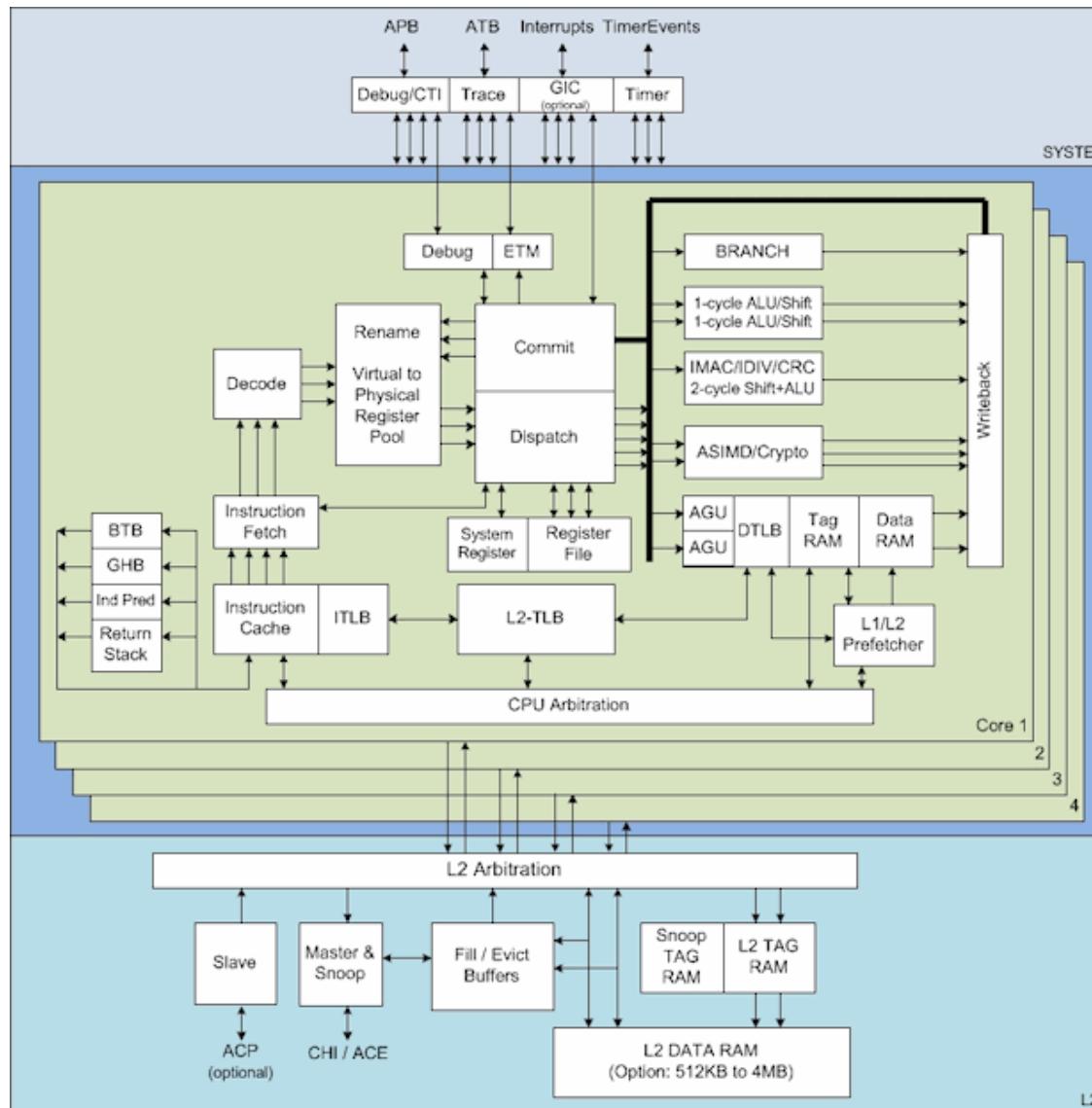
CoreLink™ CCI-500



Микропроцессор Cortex-A72

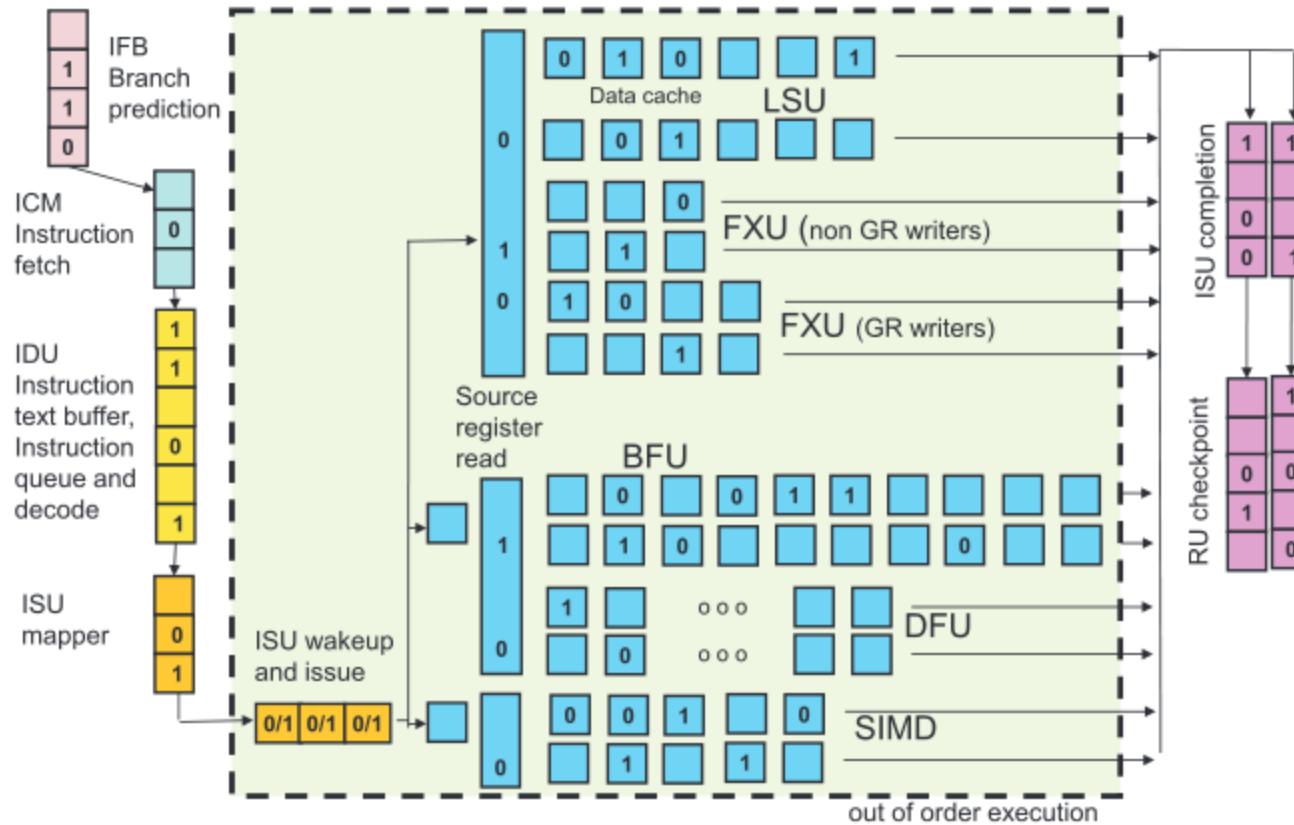


Микроархитектура Cortex-A72



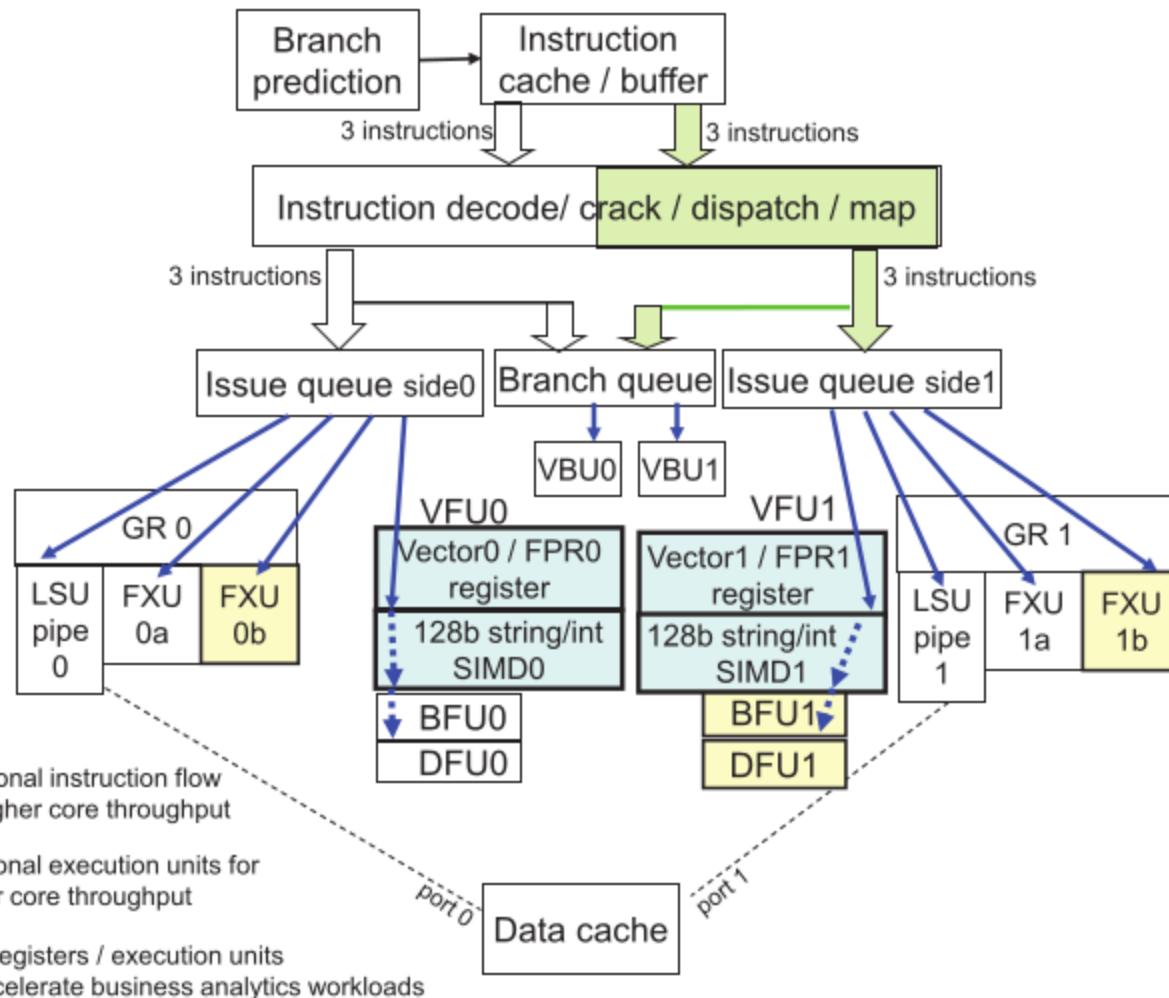
Микроархитектура IBM System z13

Конвейер



Микроархитектура IBM System z13

Стадии обработки команд



Устройства управления: классификация

По типу автомата:

- Автомат Мили.
- Автомат Мура.

По способу реализации:

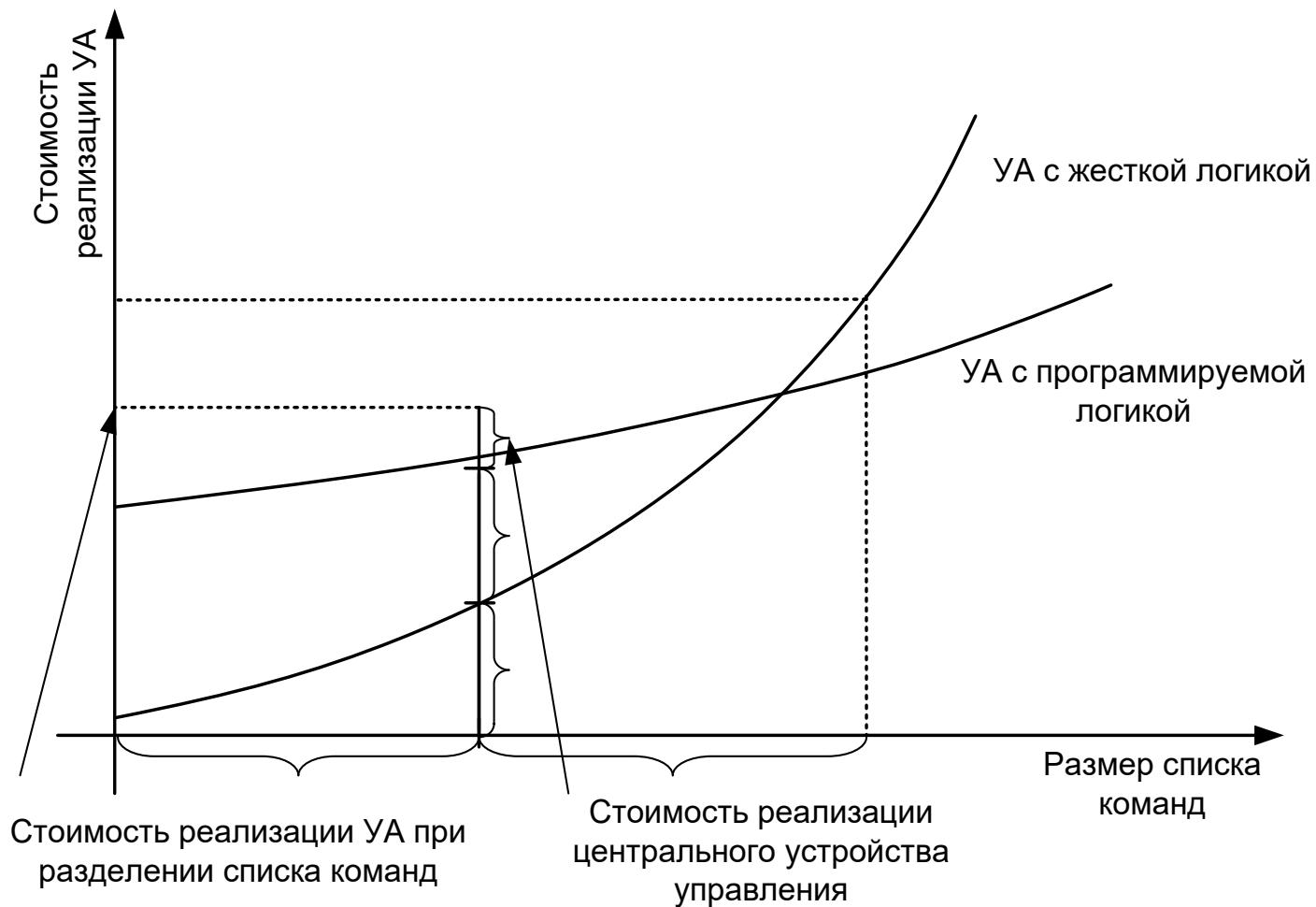
- Устройство управления с жесткой логикой.

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

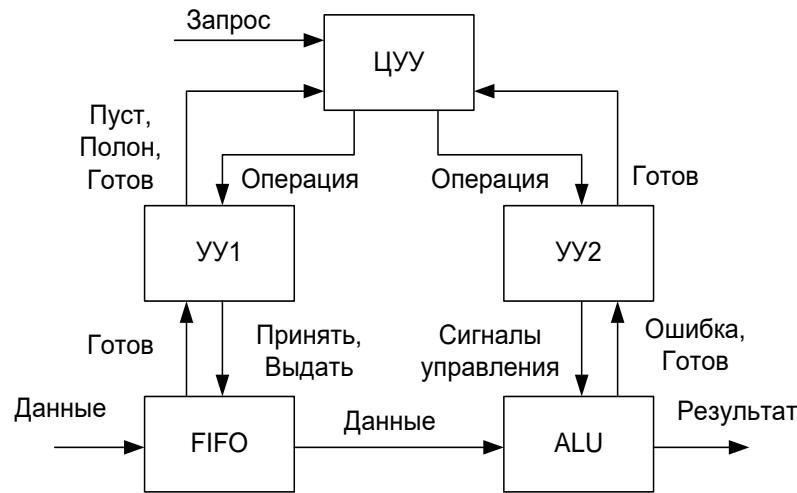
Каждой выполняемой операции ставится в соответствие совокупность хранимых в памяти слов (микрокоманд), каждая из которых содержит информацию о микрооперациях, подлежащих исполнению в текущем такте.

(+) Простота модификации и наращивания.
(-) Невысокое быстродействие для простых устройств.

Сравнение способов реализации УУ



Пример декомпозиции УУ



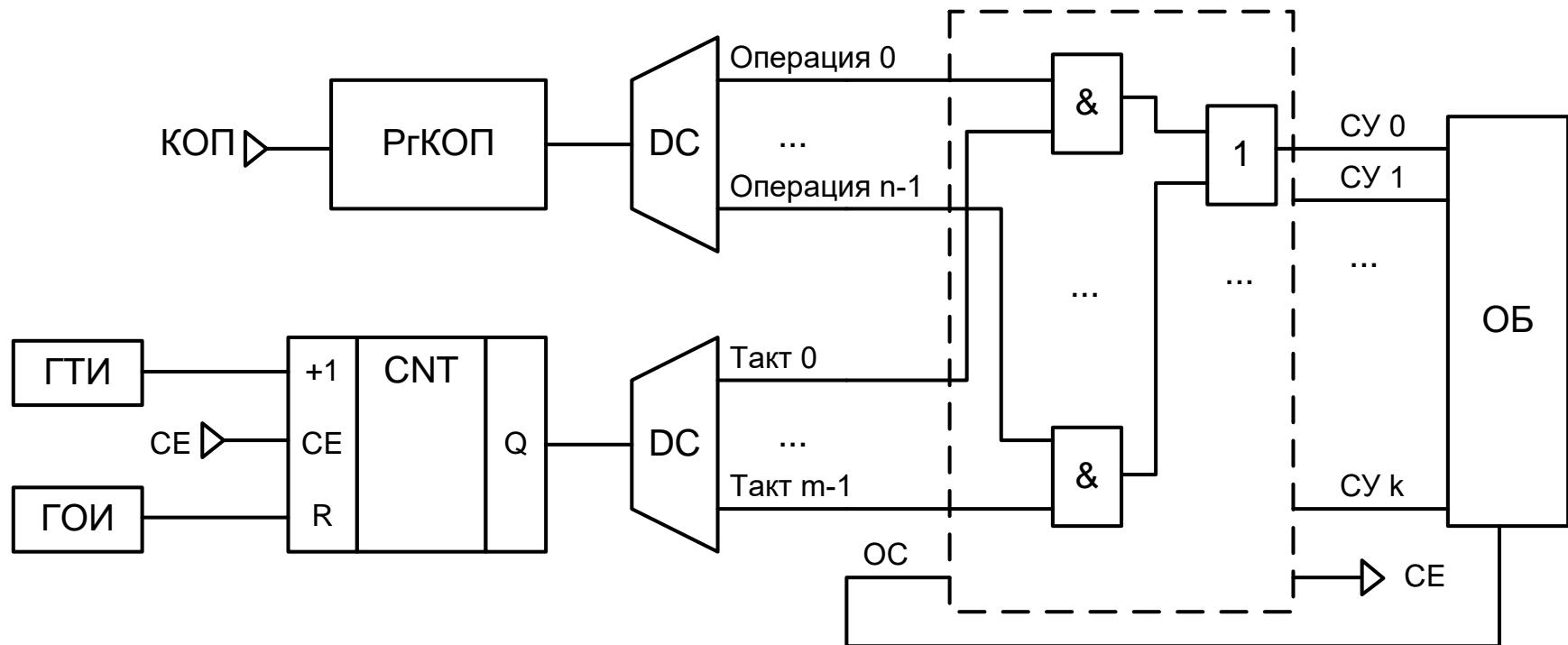
По способу кодирования микрокоманд:

- 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.

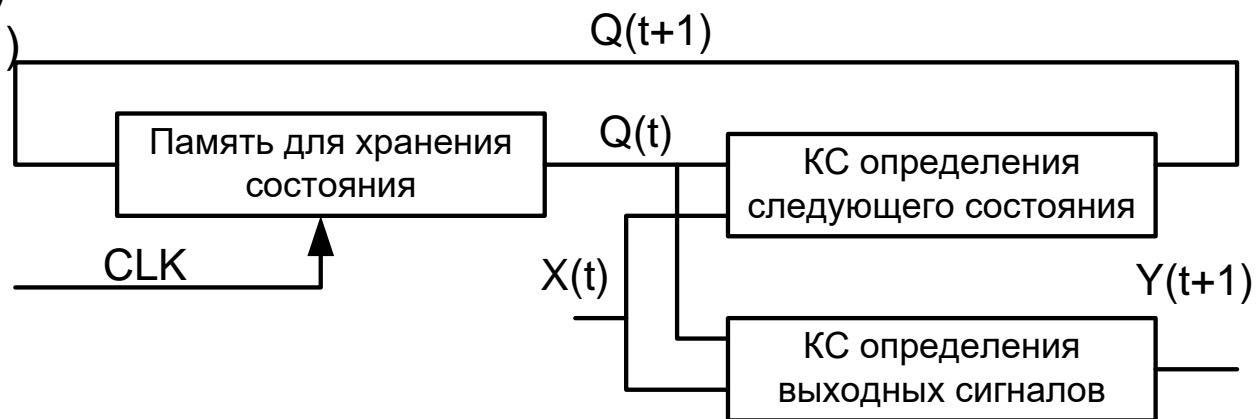
Процедура синтеза:

- 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

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

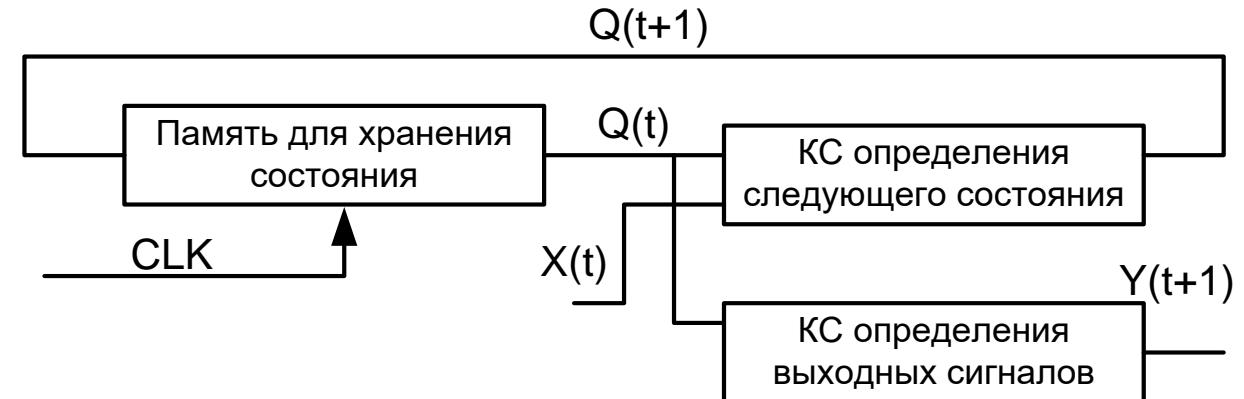
Схема автомата Мили



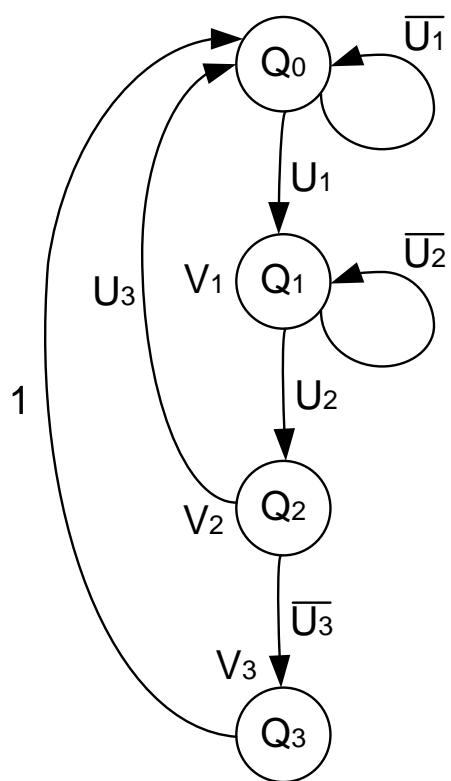
Автомат Мура

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

Схема автомата Мура



Пример синтеза УУ с ЖЛ на основе автоматов Мили и Мура



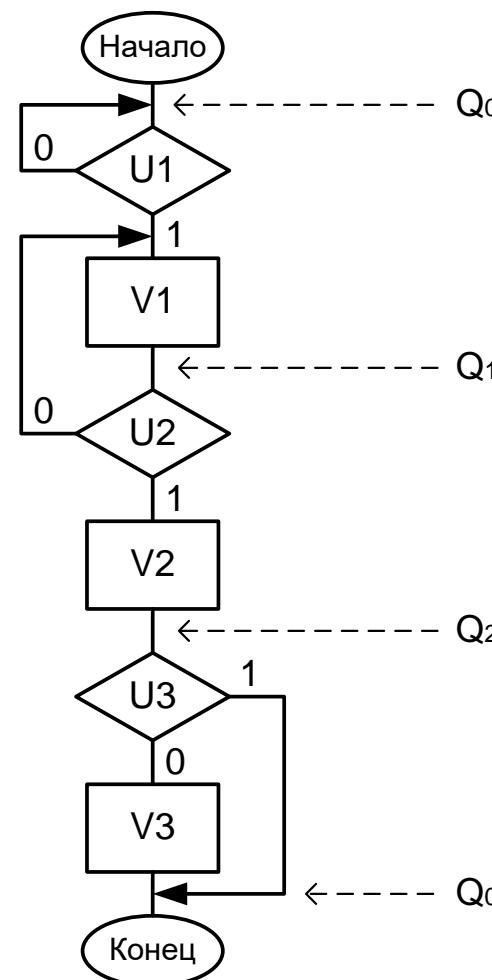
Состояния
автомата Мура

Q_0

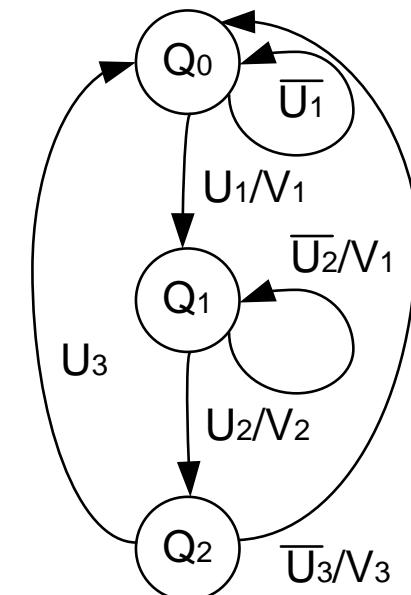
Q_1

Q_2

Q_3



Состояния
автомата Мили



Автомат Мура

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

$$\left\{ \begin{array}{l} V_1 = Q_1; V_2 = Q_2; V_3 = Q_3; \\ Q_0 = Q_0!U_1 \cup Q_2U_3 \cup Q_3; \\ Q_1 = Q_1!U_2 \cup Q_0U_1 \\ Q_2 = Q_1U_2 \\ Q_3 = Q_2!U_3 \end{array} \right.$$

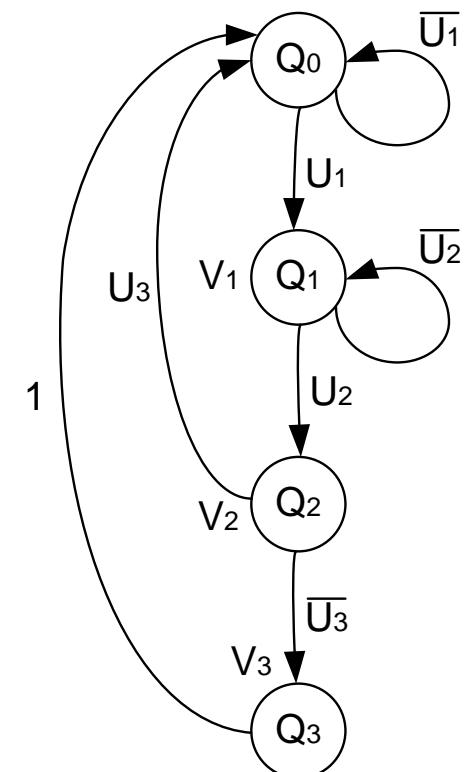
Таблица кодирования
состояний

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

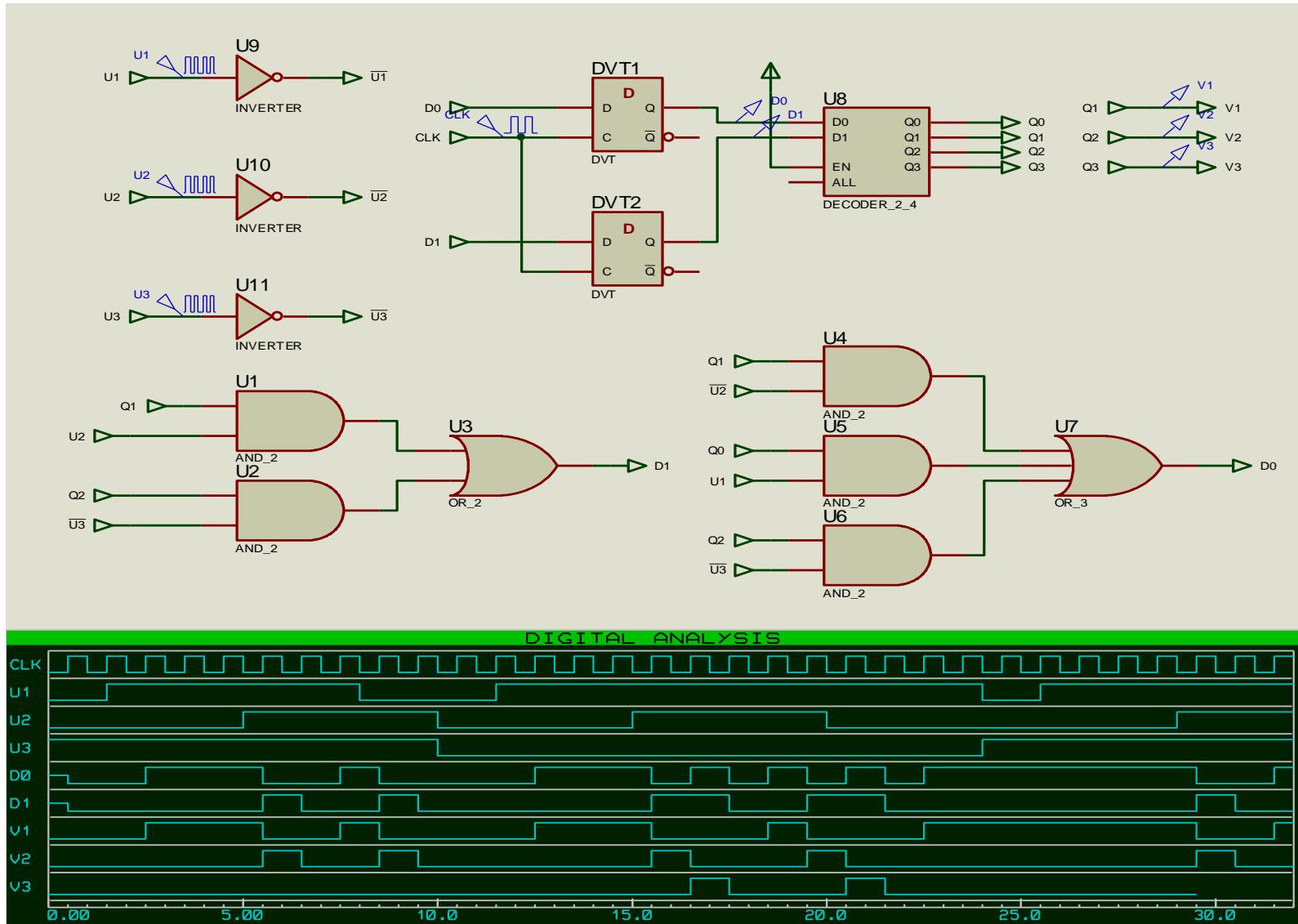
Наиболее используемые способы кодирования состояний: двоичное, One-Hot, Код Грэя

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

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



Автомат Мура



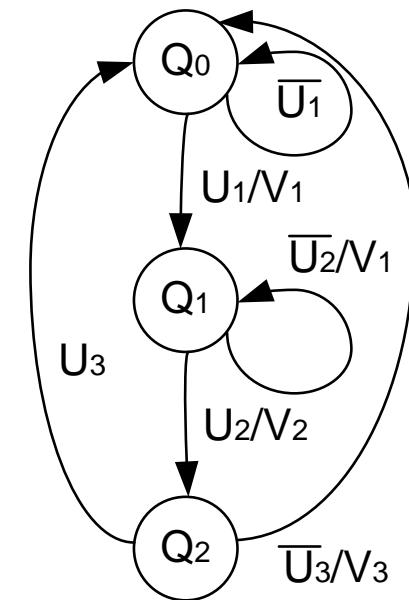
Автомат Мили

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

$$\left\{ \begin{array}{l} V_1 = Q_0 U_1 \cup Q_1! U_2; V_2 = Q_1 U_2; V_3 = Q_2! U_3; \\ Q_0 = Q_0! U_1 \cup Q_2 U_3 \cup Q_2! U_3; \\ Q_1 = Q_1! U_2 \cup Q_0 U_1 \\ Q_2 = Q_1 U_2 \end{array} \right.$$

Таблица кодирования состояний

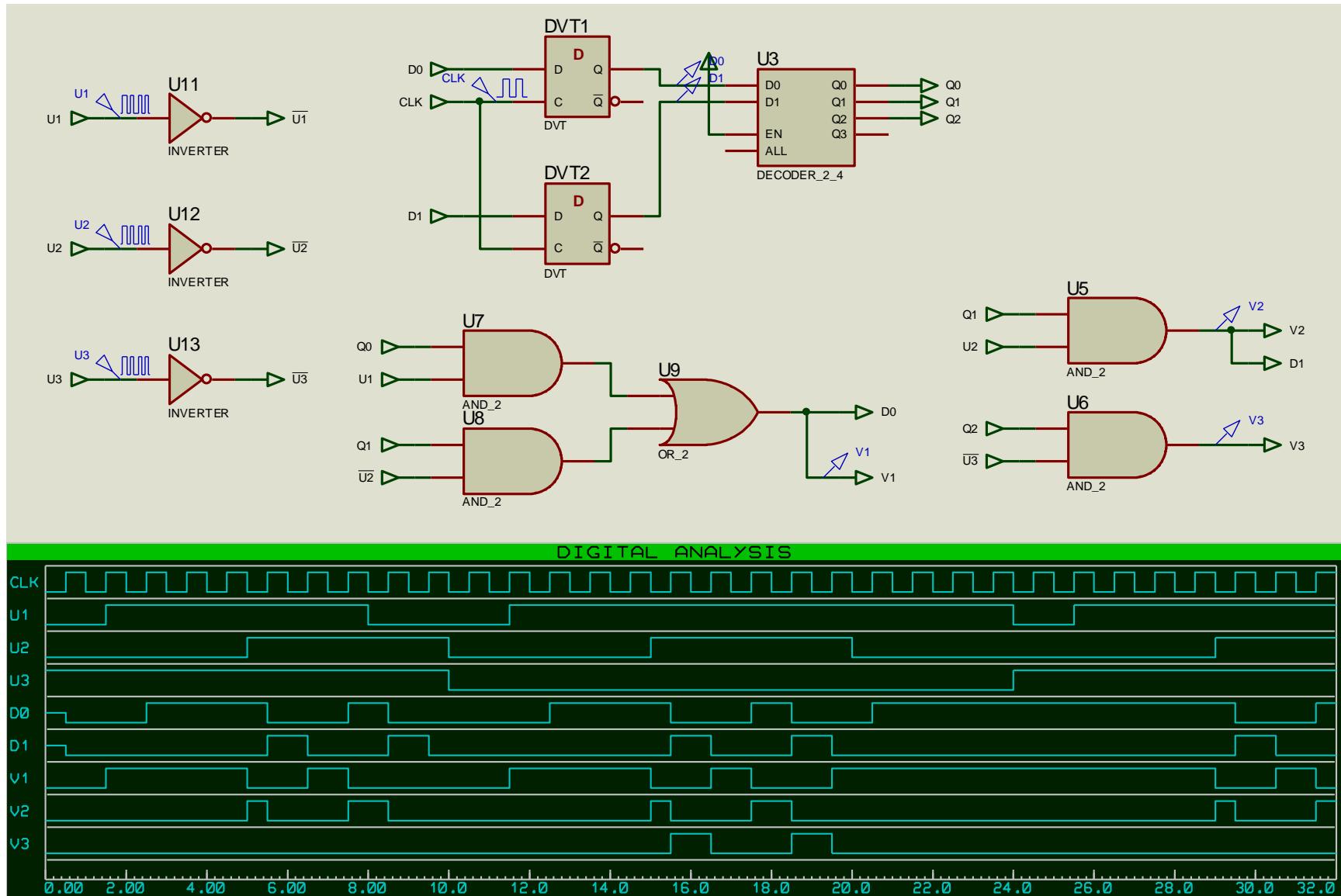
	D1	D0
Q0	0	0
Q1	0	1
Q2	1	0



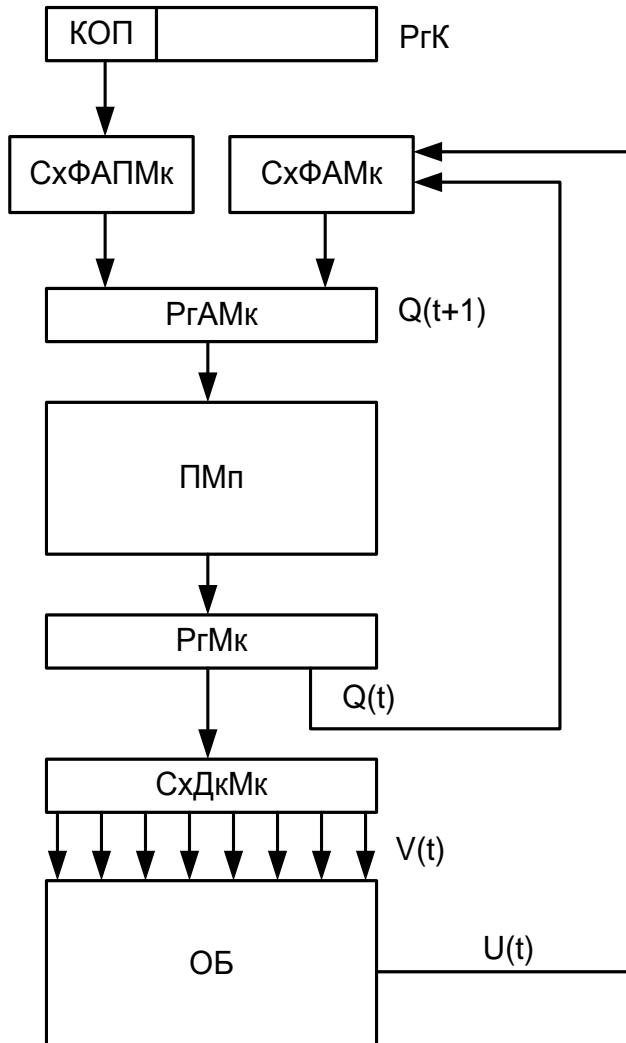
$$D_0(t+1) = Q_1(t) = Q_1! U_2 \cup Q_0 U_1;$$

$$D_1(t+1) = Q_2(t) = Q_1 U_2;$$

Автомат Мили



Управляющие устройства с программируемой логикой



PrK – 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.

В зависимости от типа ПМп:

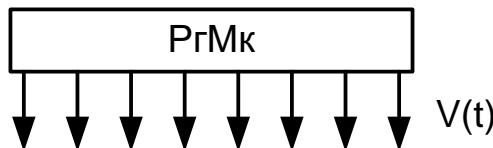
- Статическое микропрограммирование (ROM);
- Динамическое микропрограммирование (RAM).

По способу адресации микрокоманд:

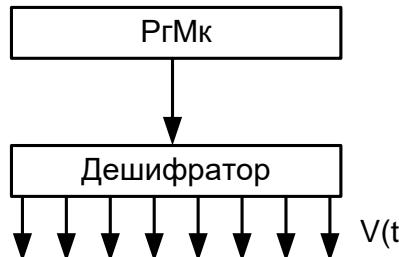
- Принудительная адресация;
- Естественная адресация.

Способы кодирования микрокоманд

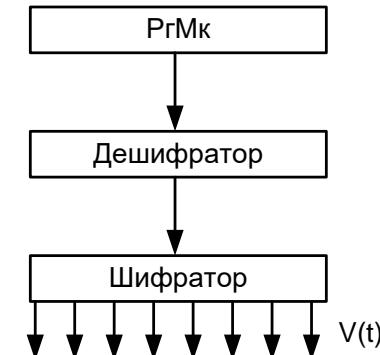
Минимальное
кодирование
(горизонтальное).



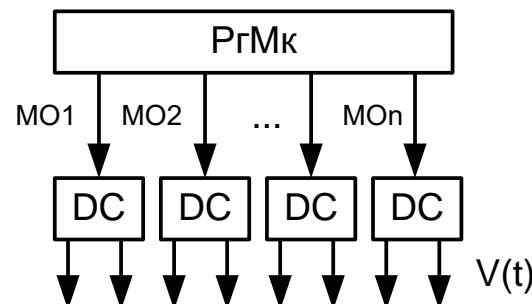
Максимальное
кодирование
(вертикальное).



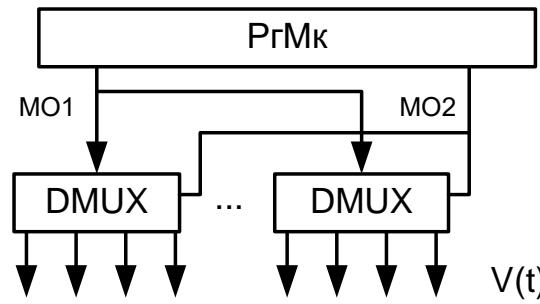
Максимальное
кодирование с
шифратором.



Вертикально-
горизонтальное
кодирование



Горизонтально-
вертикальное
кодирование.



Цикл микрокоманды

Цикл исполнения микрокоманды;

- Формирование адреса микрокоманды.
- Выборка микрокоманды.
- Исполнение микрокоманды.

При последовательном исполнении цикла:

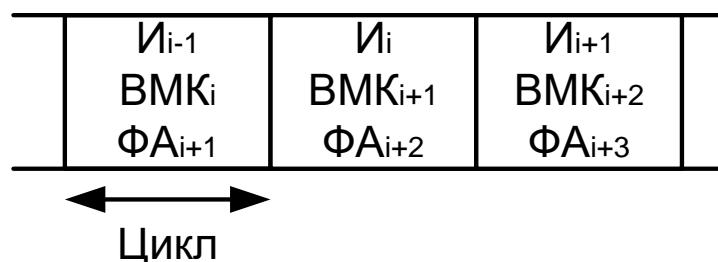
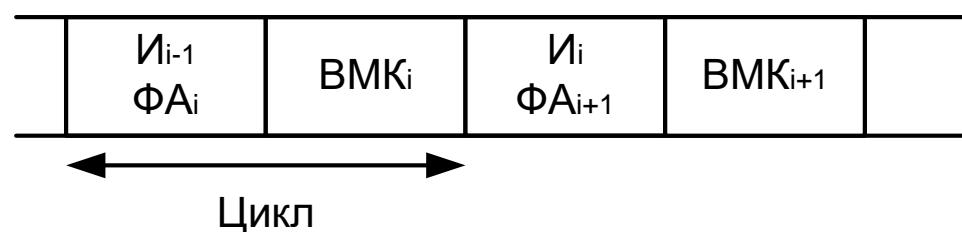
$$T_{MK} = T_{FA} + T_{VMK} + T_i.$$

При совмещении исполнения и формирования адреса следующей микрокоманды:

$$T_{MK} = \text{MAX}(T_{FA} + T_{VMK}, T_i).$$

При совмещении всех действий:

$$T_{MK} = \text{MAX}(T_{FA}, T_{VMK}, T_i).$$



Архитектура графических процессоров и GPGPU

Алгоритмы машинной графики (потоковые, вычислительные, параллельные):

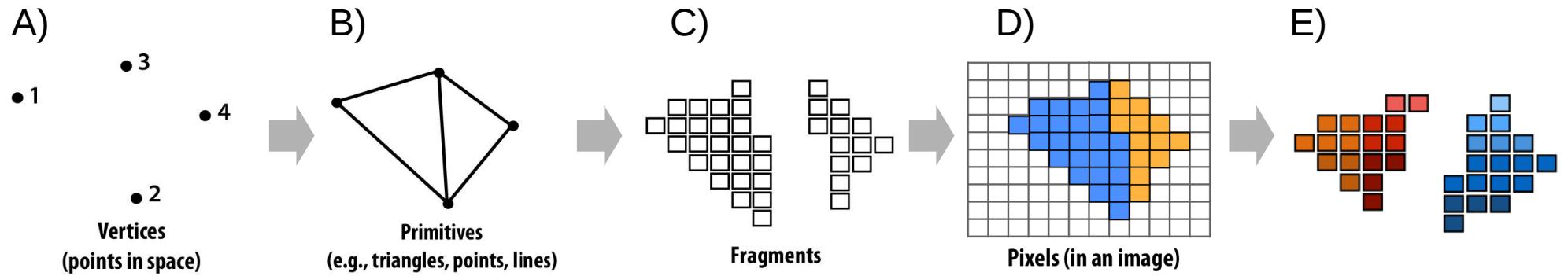
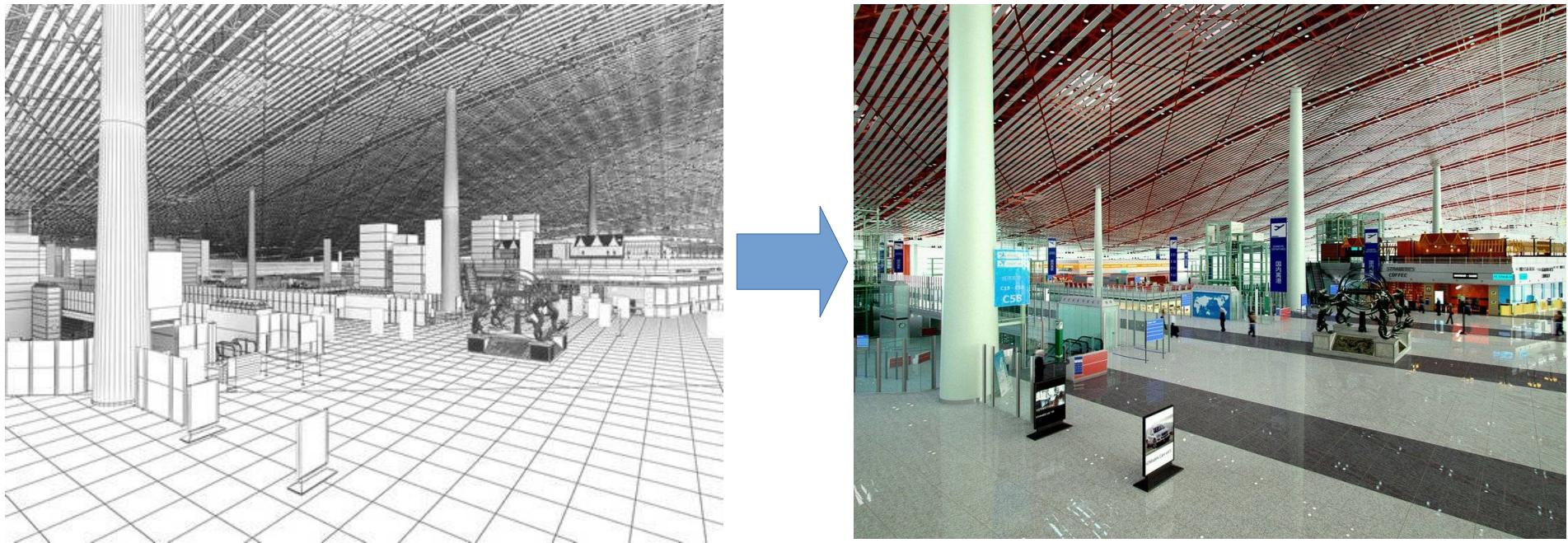
- преобразование систем координат;
- удаление невидимых поверхностей;
- отсечение невидимых областей;
- отрисовка базовых графических примитивов (точек, прямых, ломаных и т.п.);
- заливка / штриховка (растровая развертка сплошных областей);

Графический конвейер



- *Transform* – задание положения каждой вершины в сцене;
- *Clip* – отсечение скрытых областей, в т.ч. за пределами области видимости;
- *Rasterize* – переход от векторного представления к пикельному;
- *Shade* – вычисление цвета каждого пикселя;
- *Visibility/Blend* - расчет наложений и цвета для прозрачных объектов.
- *Display* – окончательное формирование изображения в памяти.

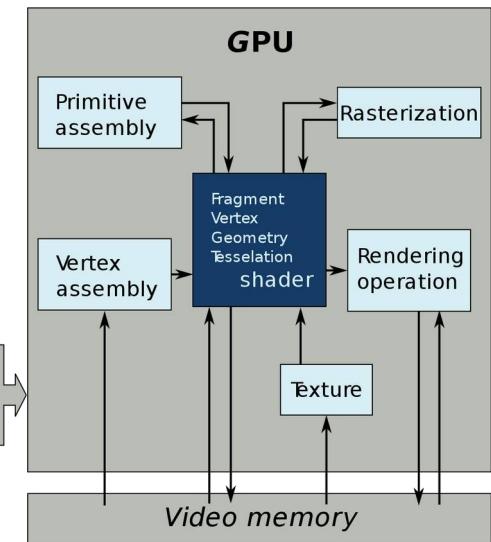
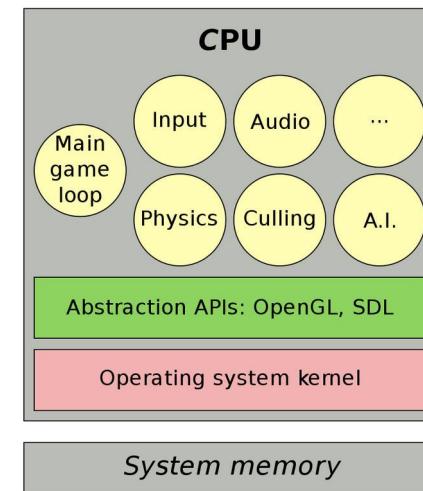
Графический конвейер (пример)



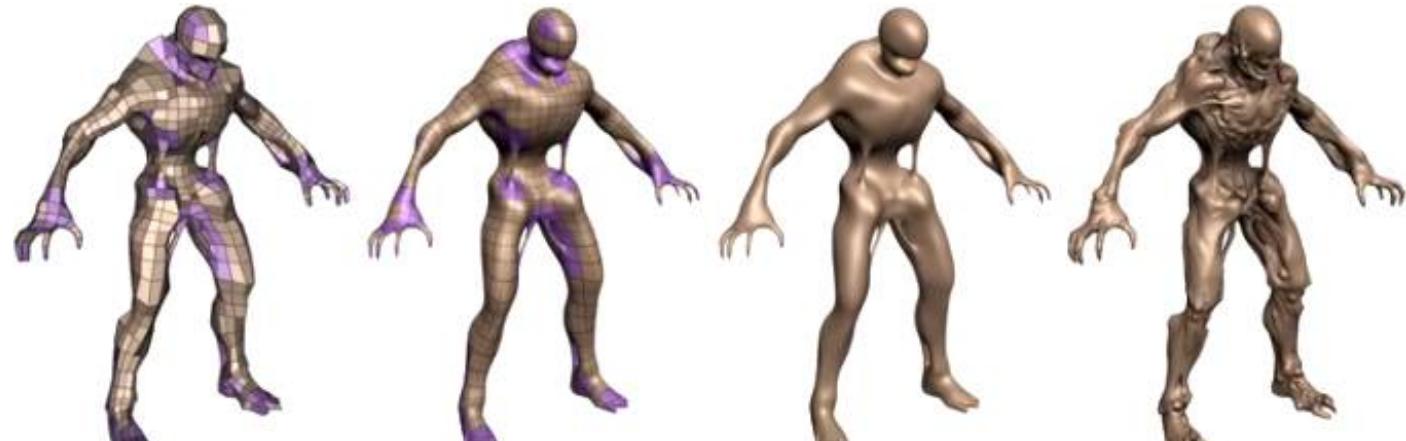
Шейдеры

Шейдер – это программа, которая загружается в ускоритель, и конфигурирует его узлы для обработки соответствующих элементов. Шейдер позволяет снять ограничение на способ обработки эффектов.

- вершинные;
- геометрические;
- пиксельные или фрагментные.



Shader Forge это
визуальный редактор
шейдеров для Unity.



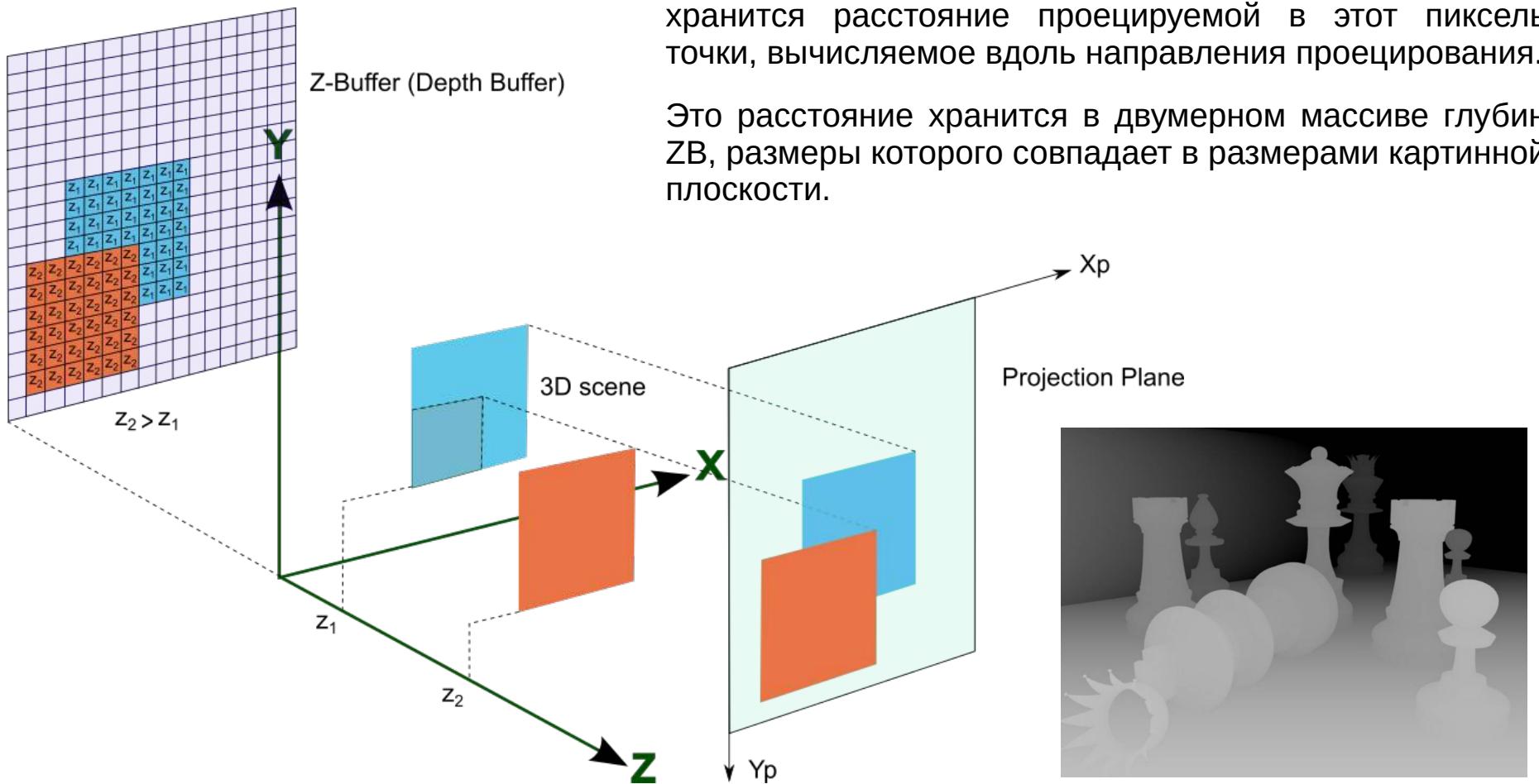
Тесселяция используется для увеличения геометрической сложности моделей, когда из низкополигональной получается более сложная.

Z-буферизация

Z-буферизация — в компьютерной трёхмерной графике способ учёта удалённости элемента изображения.

При решении задачи загораживания методом Z-буфера при выводе текущего полигона формируется его растровое представление на картинной плоскости и для каждого пикселя картинной плоскости, кроме цвета, хранится расстояние проецируемой в этот пиксель точки, вычисляемое вдоль направления проецирования.

Это расстояние хранится в двумерном массиве глубин ZB, размеры которого совпадают в размерами картинной плоскости.



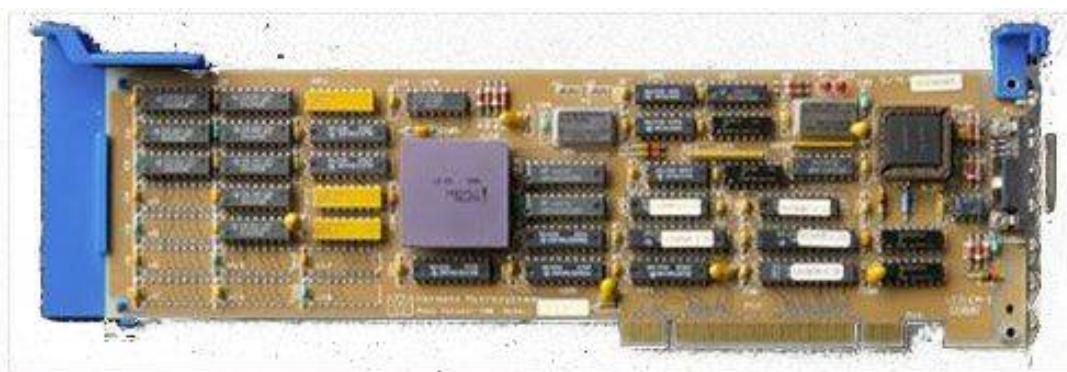
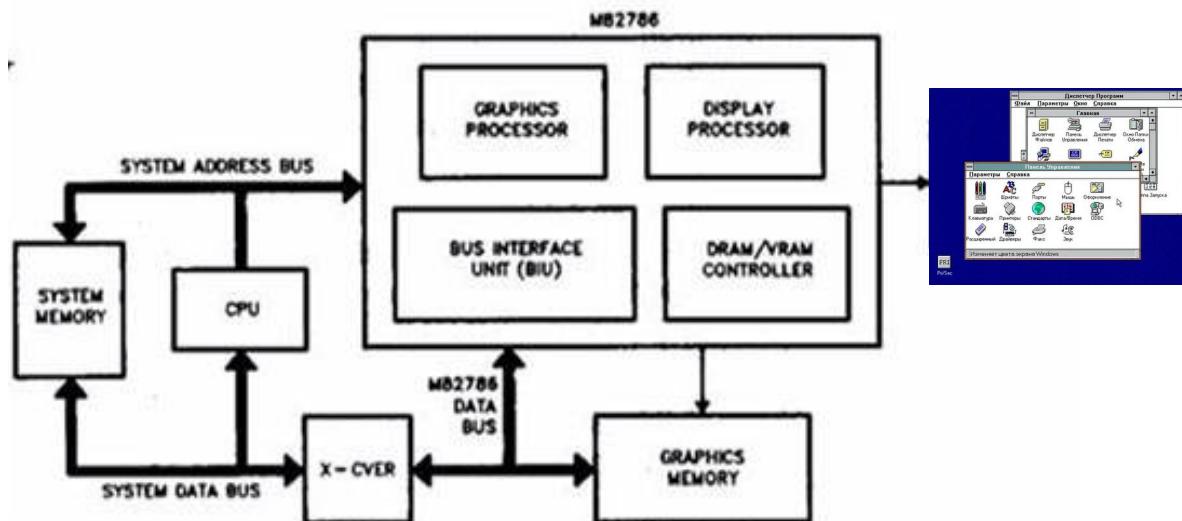
Различия в архитектурах GPU и CPU

CPU	GPU
Ядра CPU проектируются для выполнения одного потока последовательных инструкций с максимальной производительностью	GPU предназначен для выполнения большого количества параллельных потоков команд.
В CPU доступ к памяти зависит от поступивших команд и часто происходит по случайным адресам. Требуется использовать кэш-память для ускорения доступа к памяти.	В GPU доступ к памяти преимущественно последовательный (пиксели и текстуры читаются и пишутся последовательно). Большой кэш не требуется.
Доступ к памяти плохо распараллеливается, данные сосредоточены в сегментах и выборка осуществляется в небольшое количество модулей памяти (по крайней мере, для одной задачи.)	Доступ к памяти легко распараллеливается и пропускная способность каналов памяти велика.
В универсальных процессорах большую часть площади кристалла занимают различные блоки конвейера: декодеры, буферы, ROB, кэш-память и пр.	Аппаратная часть GPU оптимизирована под выполнение небольших и программ (шейдеров).
CPU хорошо справляется с зависимыми данными.	GPU предназначен для вычислений независимых данных (пикселей, текстур). При наличии зависимостей скорость вычислений существенно падает.

Первый дискретный графический сопроцессор Intel 82786

Графический процессор (GP) и дисплейный процессор (DP) были независимыми процессорами в 82786. Шинный интерфейсный блок (BIU) с контроллером DRAM / VRAM обрабатывал запросы шины между графическим процессором, дисплеем и внешним ЦП или шиной.

Графический процессор выполнял команды, размещенные в системной памяти и формирует изображения в битовых картах видеопамяти для дисплейного процессора во взаимодействии с контроллером видеопамяти и интерфейсным устройством шины.



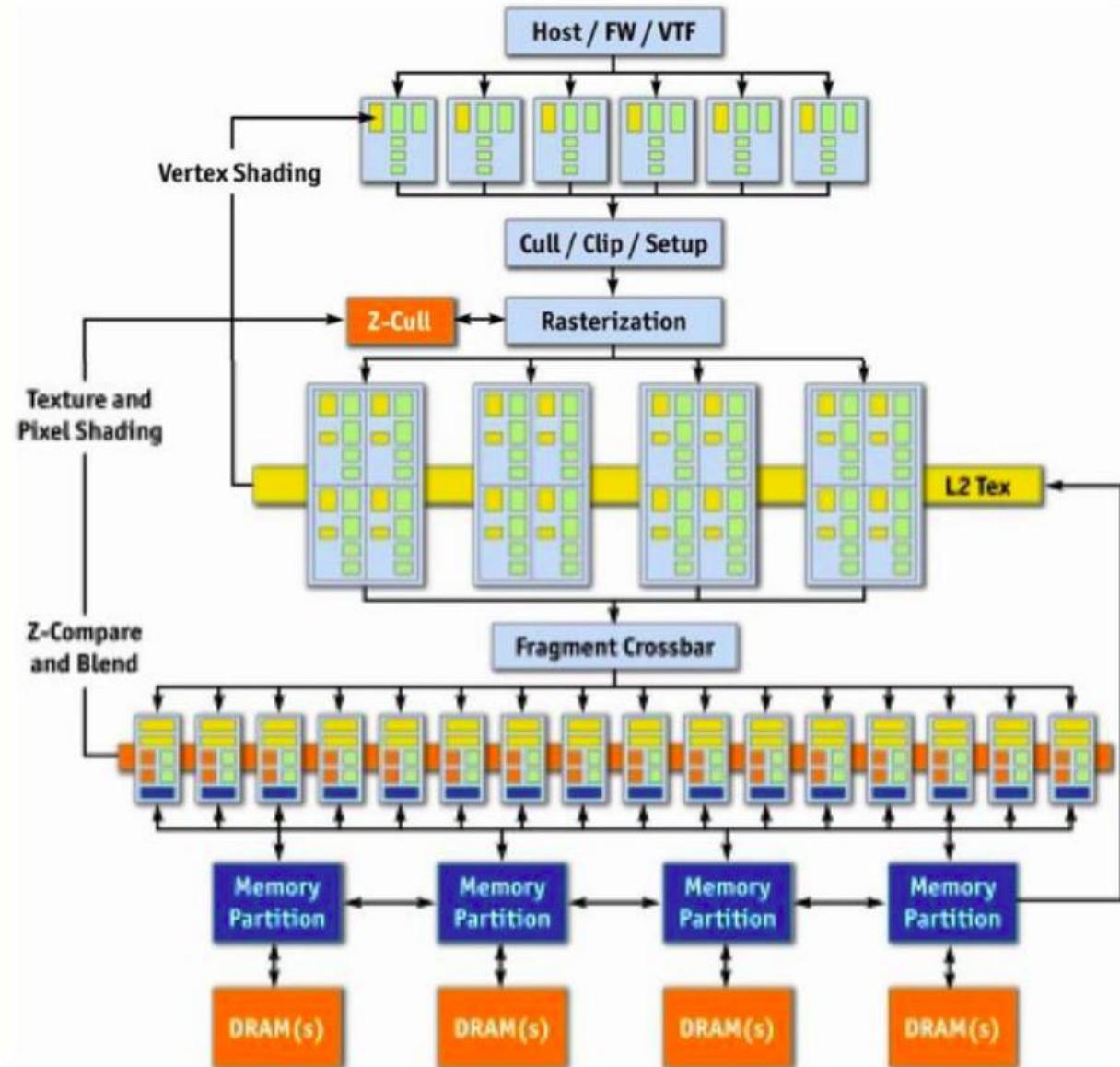
Дисплейный процессор преобразует битовые карты, создаваемые графическим процессором в растровые последовательности для видеоконтрольного устройства, которое отображает их в виде отдельных окон на экране графического монитора.

Современные графические процессоры GPU

Nvidia GeForce6 (NV40), 2004

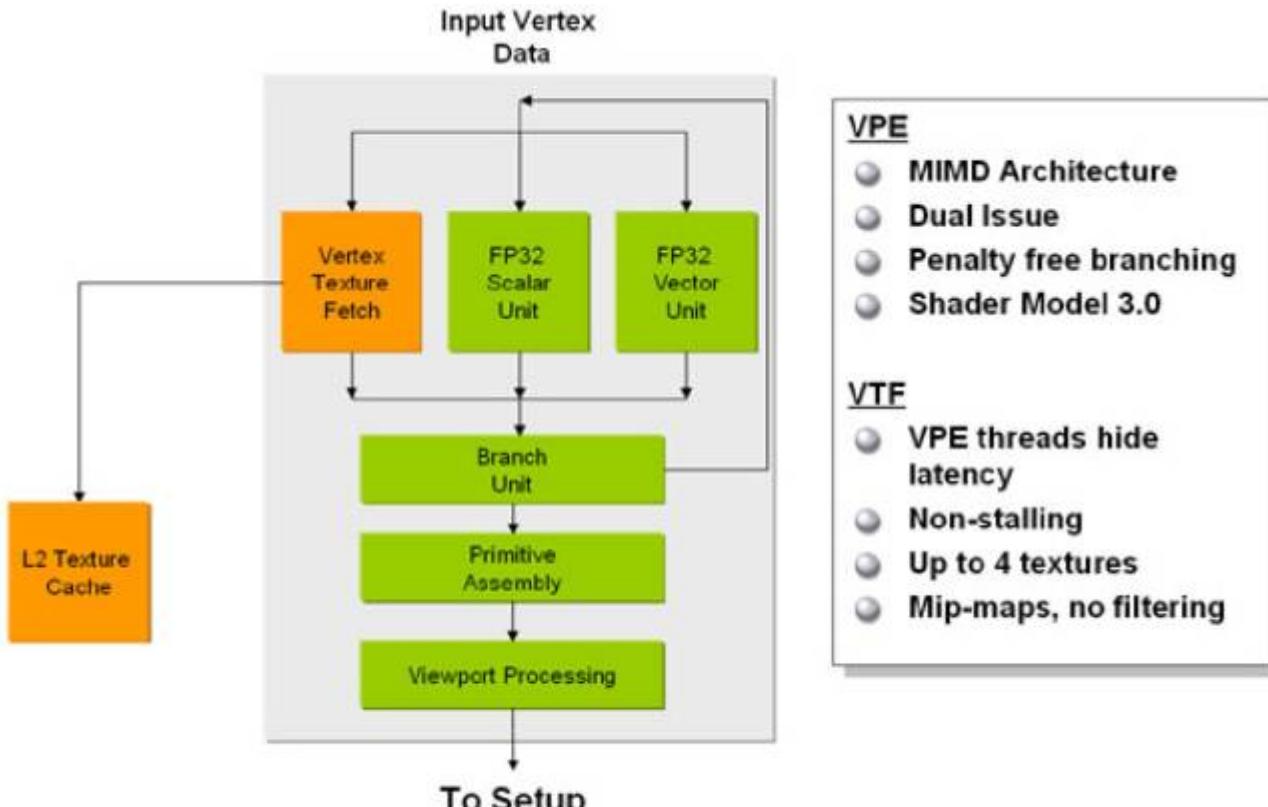
Этапы обработки:

- Загрузка в чип ускорителя вершин из памяти акселератора
- Обработка в вершинных процессорах.
- Установка треугольников, отсечение невидимых поверхностей.
- Треугольники растеризуются и производится отсечение невидимых частей с использованием Z-буфера (Hidden Surface Removal, HSR).
- Формируются квады 2x2 пикселя, подлежащие закраске. На квады накладываются текстурные фрагменты и производится интерполяция.
- Закраска фрагментов.
- Производится блендинг (смещение)
- Значения квадов записываются в буфер кадра
- Вывод изображения на экран.



Современные графические процессоры GPU

Nvidia GeForce6 Вершинные процессоры

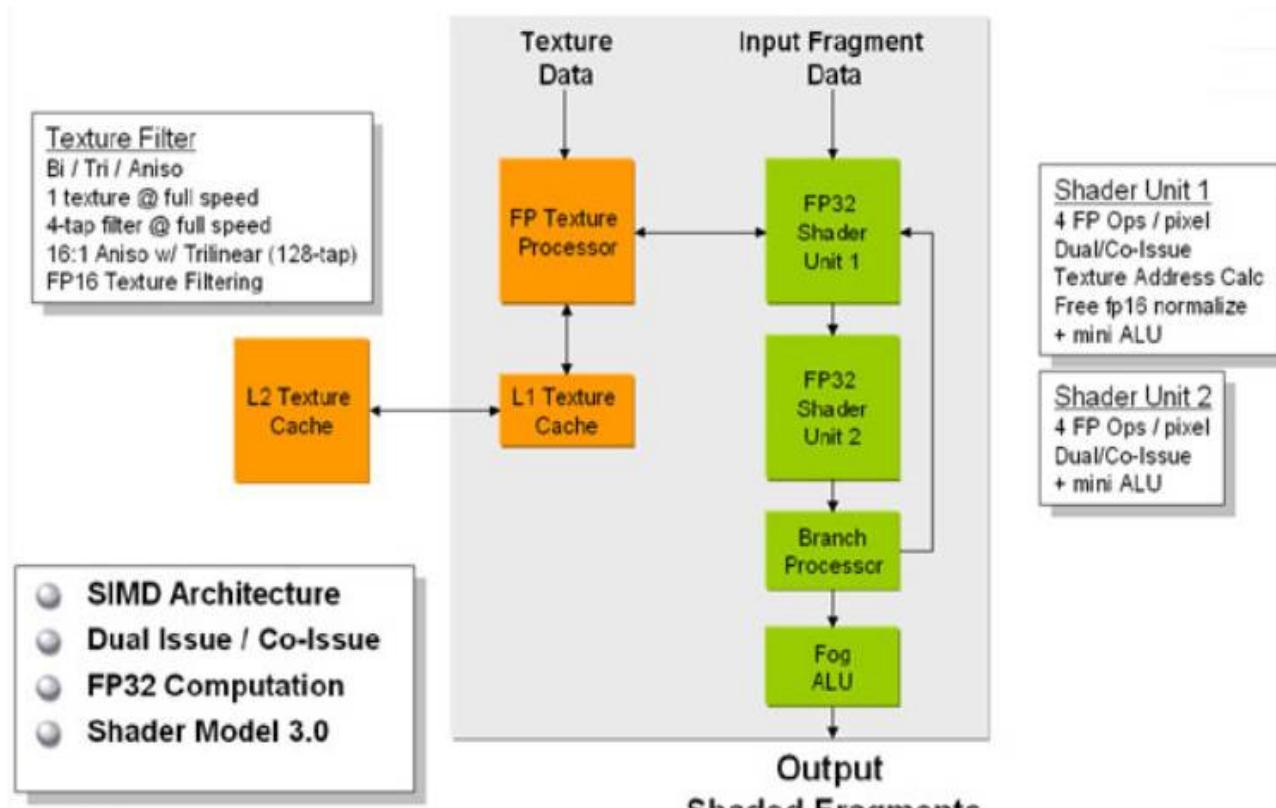


Благодаря расширению динамического выполнения (больше вариантов по циклам/ветвлений и новые функции подпрограмм), можно создавать более эффективный код и реализовывать новые возможности для эффектов.

- полная поддержка вершинных программ 3.0;
- 216 (65,535) длинных вершинных программ;
- вершинная обработка с учётом текстуры - карты смещения (displacement mapping);
- динамический контроль выполнения (flow control) - циклы и ветвления, вызов подпрограмм и возврат;
- Geometry Instancing (vertex stream divider).

Современные графические процессоры GPU

Nvidia GeForce Пиксельные конвейеры

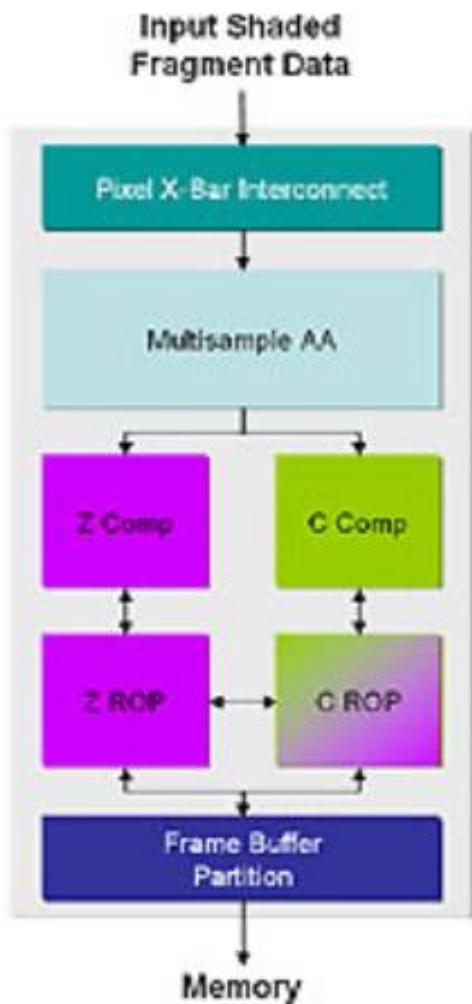


- Каждый из 16 пиксельных конвейеров имеет два блока пиксельных программ (суперскалярный дизайн) и один текстурный процессор с плавающей запятой.
- NV40 также оснащён четырьмя кэшами текстур L1, каждый из которых обслуживает четыре конвейера.
- Разгрузить интерфейс памяти помогает и массивный кэш L2.
- Архитектура блоков пиксельных программ-шейдеров имеет настоящий дизайн SIMD (одна инструкция - много данных).
- Если первый блок пиксельных программ на каждом конвейере может выполнять как арифметические операции, так и чтение текстур и нормализацию, то второй блок ограничен только арифметикой.
- Если блок не занимается текстурированием, то он может выполнять (в данный проход) пиксельное затенение. Блок 2 всегда доступен для пиксельного затенения.

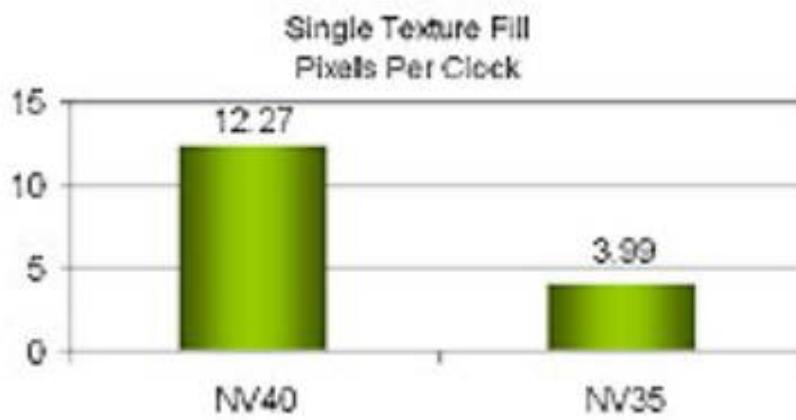
Современные графические процессоры GPU

Nvidia GeForce6

Пиксельные конвейеры ROP (растровые операции)



- OpenEXR Floating point blending
- Rotated Grid AA
- Double-speed Z
- Lossless Color & Z Compression
- Multiple Render Targets



Подсистема ROP карты GeForce 6800 имеет следующие характеристики:

- 16 пикселей за такт, цвет и Z;
- 32 пикселей за такт, только Z;
- 64-bit FP Frame Buffer Blending;
- цветовое и Z-сжатие без потерь;
- качественное сглаживание - поворот сетки (Rotated Grid);
- полная поддержка MRT;
- ускоренный рендеринг теней.

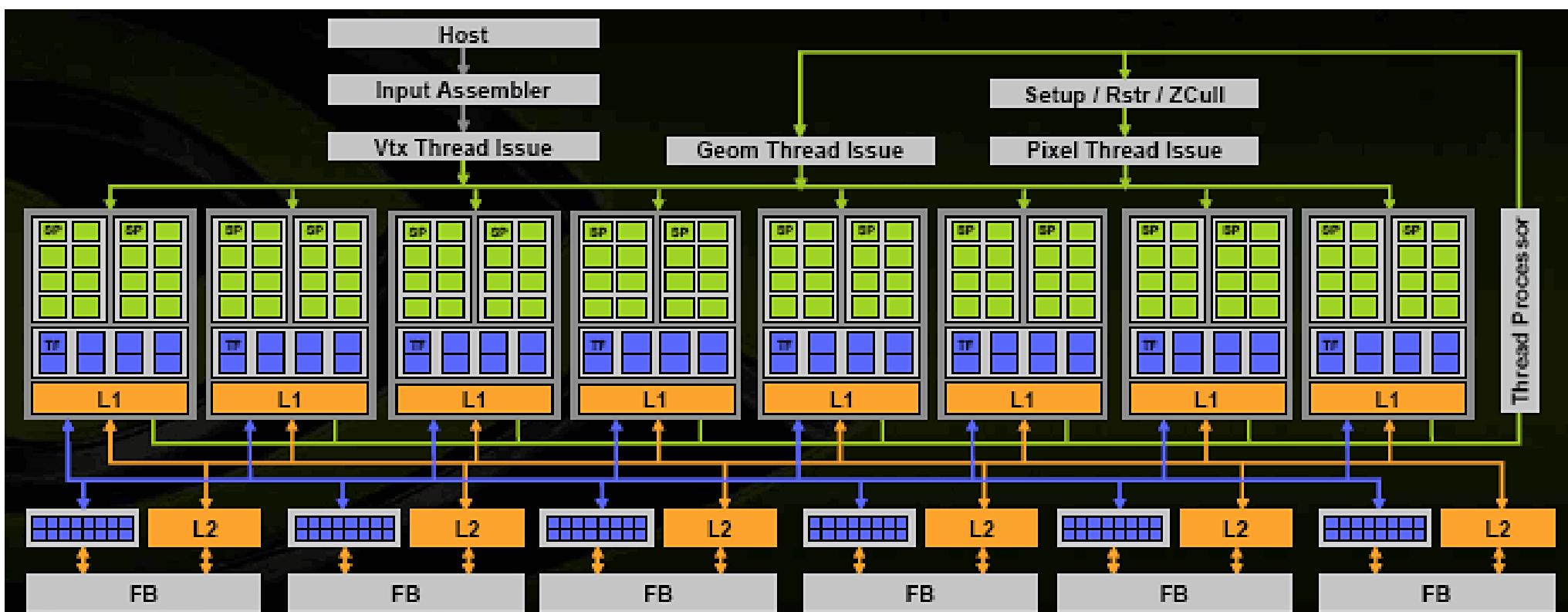
Современные графические процессоры GPGPU

NVidia G80

Чип состоит из 8 универсальных вычислительных блоков (шейдерных процессоров) по 4 TMU и 16 ALU в каждом. Всего, таким образом, имеется 128 ALU (называются потоковыми процессорами (SM, Stream Processors) и 32 TMU. Все ветвления, переходы, условия и т.д. применяются целиком к одному блоку и таким образом логичнее всего, его и называть шейдерным процессором, пускай и очень широким.

Каждый такой процессор снабжен собственным кэшем первого уровня, в котором теперь хранятся не только текстуры, но и другие данные, которые могут быть запрошены шейдерным процессором.

Кроме управляющего блока и 8 вычислительных шейдерных процессоров в наличии 6 блоков ROP, исполняющих определение видимости, запись в буфер кадра и MSAA (синие, рядом с блоками кэша L2) сгруппированные с контроллерами памяти, очередями записи и кэшем второго уровня.



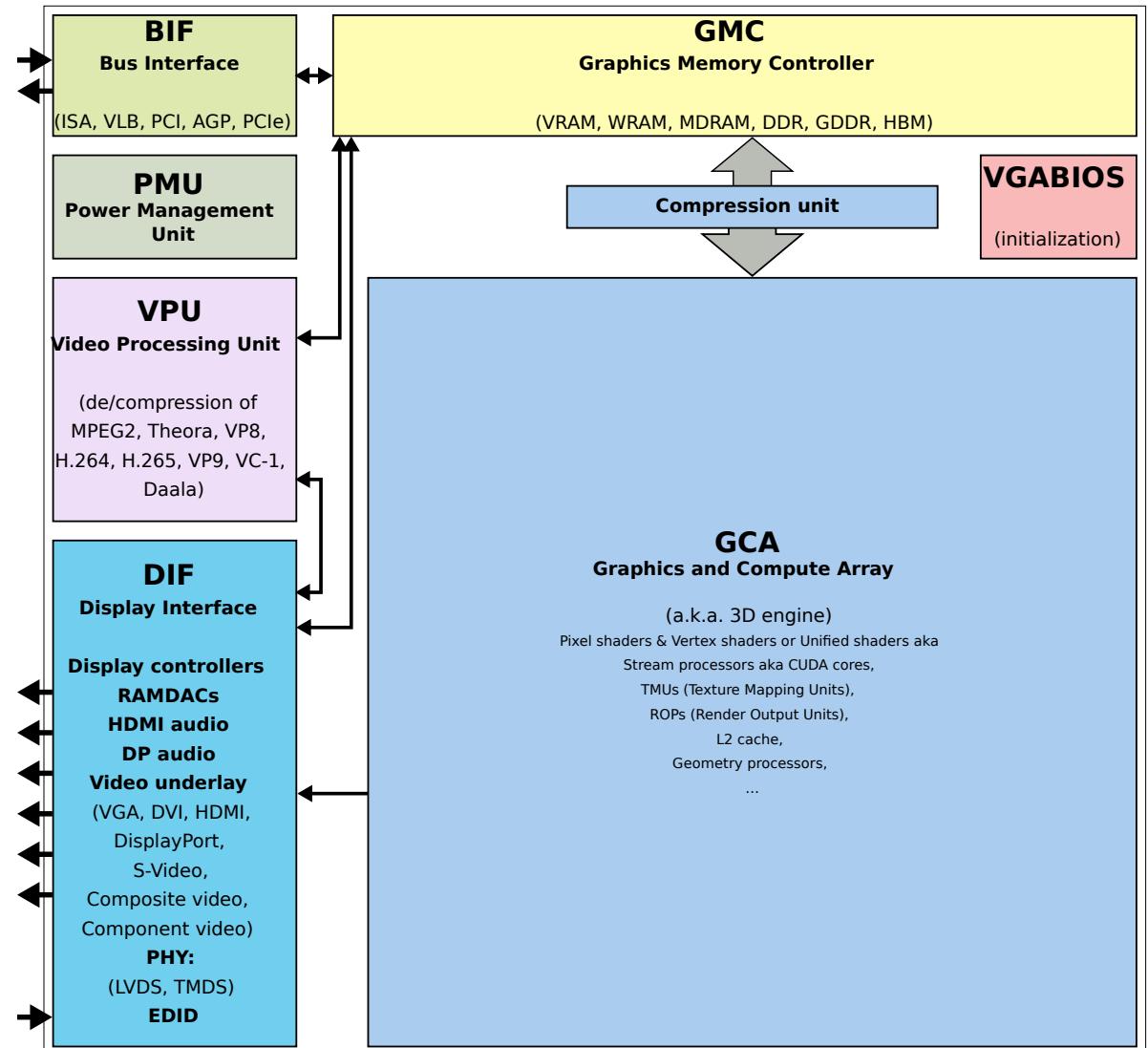
Обобщенная схема GPGPU

Отличительными особенностями по сравнению с ЦП являются:

- архитектура, максимально нацеленная на увеличение скорости расчёта текстур и сложных графических объектов;
- ограниченный набор команд.
-

Высокая вычислительная мощность GPU объясняется особенностями архитектуры. Современные CPU содержат несколько ядер, тогда как графический процессор изначально создавался как многопоточная структура с множеством ядер. Разница в архитектуре обуславливает и разницу в принципах работы. Если архитектура CPU предполагает последовательную обработку информации, то GPU исторически предназначался для обработки компьютерной графики, поэтому рассчитан на массивно параллельные вычисления.

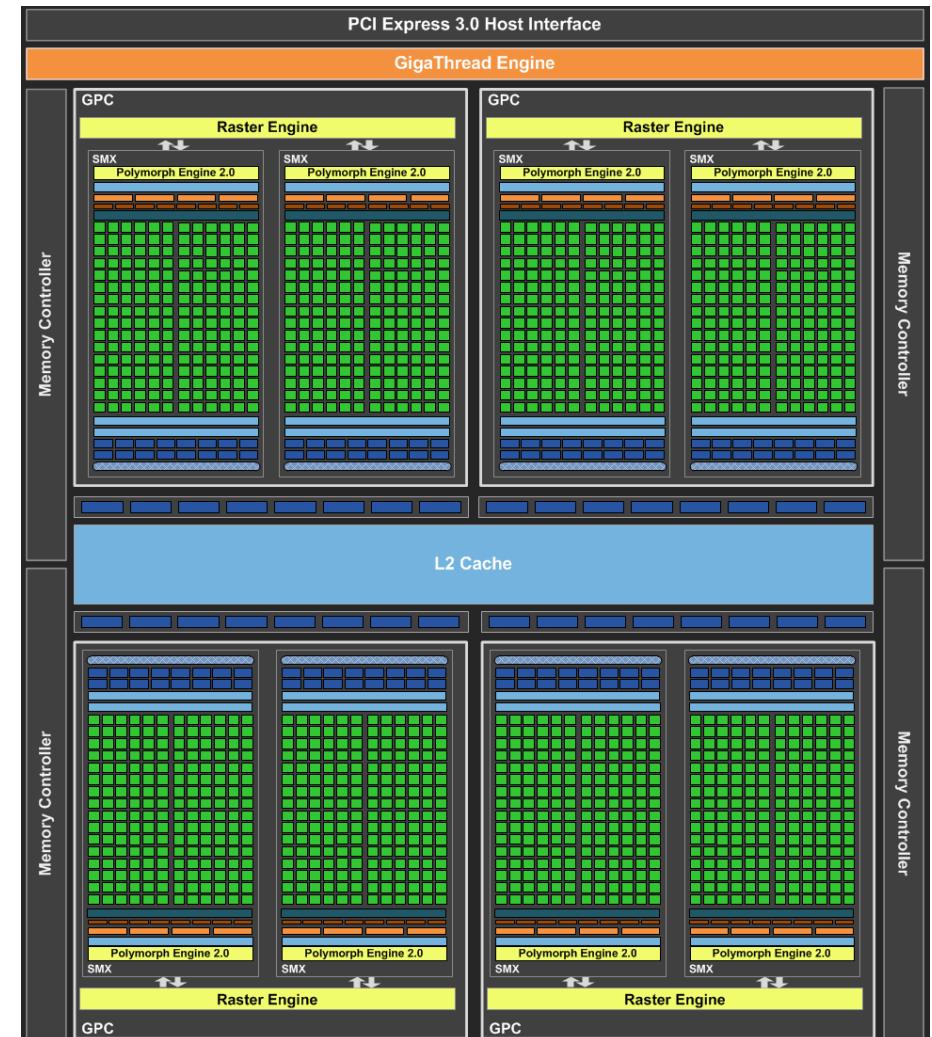
Каждая из этих двух архитектур имеет свои достоинства. CPU лучше работает с последовательными задачами. При большом объёме обрабатываемой информации очевидное преимущество имеет GPU. Условие только одно — в задаче должен наблюдаться параллелизм.



Современные графические процессоры GPU

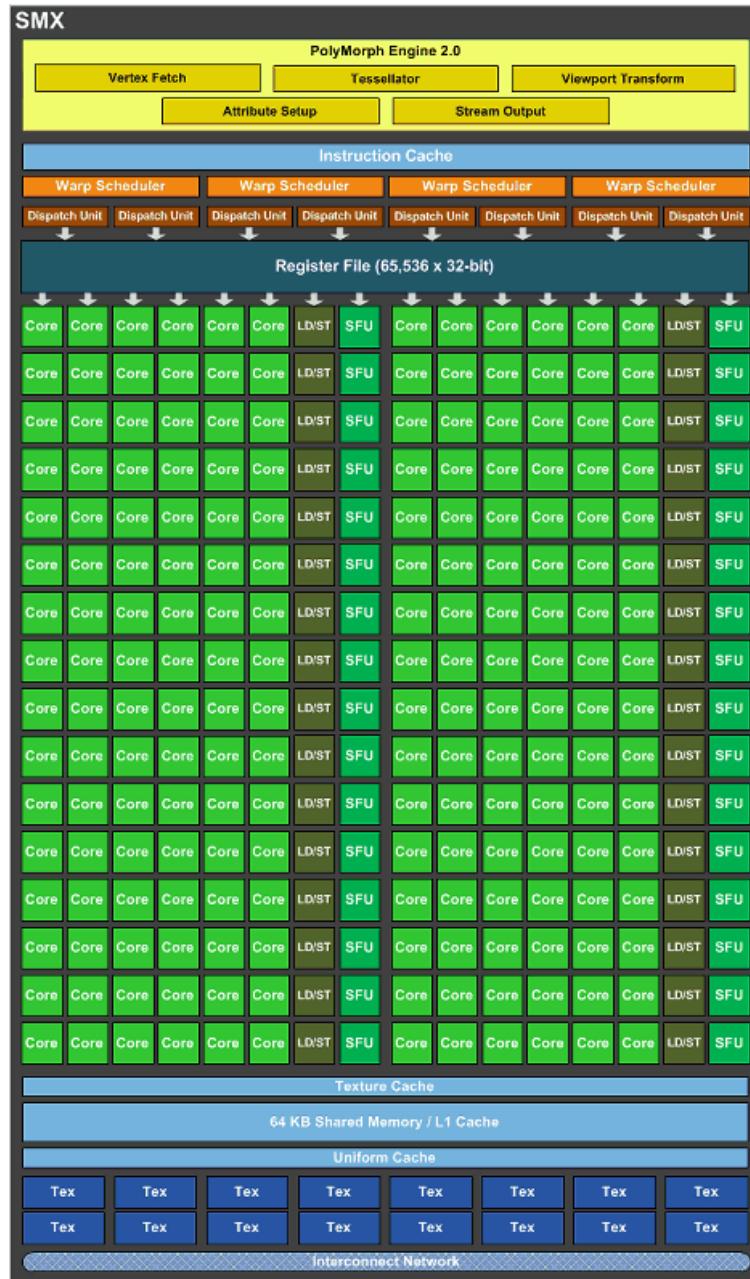
NVidia GeForce GTX 600

- Технология производства 28 нм;
- 3.54 миллиардов транзисторов;
- Площадь ядра 294 мм²;
- Унифицированная архитектура с массивом процессоров для потоковой обработки различных видов данных: вершин, пикселей и др.;
- Аппаратная поддержка DirectX 11 API, в том числе шейдерной модели Shader Model 5.0, геометрических и вычислительных шейдеров, а также тесселяции;
- 256-битная шина памяти, четыре независимых контроллера шириной по 64 бита каждый, с поддержкой GDDR5 памяти;
- Базовая частота ядра 1006 МГц;
- Средняя турбо-частота ядра 1058 МГц;
- 8 потоковых мультипроцессоров, включающих 1536 скалярных ALU для расчётов с плавающей запятой (поддержка вычислений в целочисленном формате, с плавающей запятой, с FP32 и FP64 точностью в рамках стандарта IEEE 754-2008);
- 128 блоков текстурной адресации и фильтрации с поддержкой FP16 и FP32 компонент в текстурах и поддержкой трилинейной и анизотропной фильтрации для всех текстурных форматов;
- 4 широких блока ROP (32 пикселя) с поддержкой режимов антиалиасинга до 32 выборок на пиксель, в том числе при FP16 или FP32 формате буфера кадра. Каждый блок состоит из массива конфигурируемых ALU и отвечает за генерацию и сравнение Z, MSAA, блендинг;
- Интегрированная поддержка RAMDAC, двух портов Dual Link DVI, а также HDMI и DisplayPort.
- Интегрированная поддержка четырёх мониторов, включая два порта Dual Link DVI, а также HDMI 1.4a и DisplayPort 1.2
- Поддержка шины PCI Express 3.0



Современные графические процессоры GPU

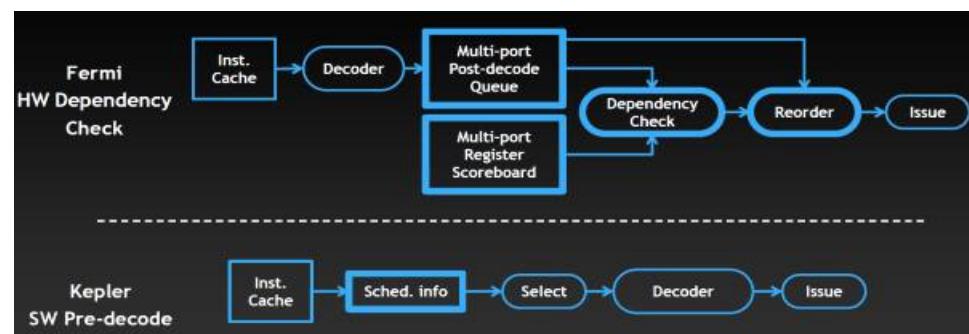
NVidia GeForce GTX 600



Мультипроцессоры — это основная составная часть GPU компании NVIDIA. По сравнению с предыдущими SM, новые SMX обеспечивают более высокую производительность, что видно по количеству функциональных устройств в составе SMX, но при этом потребляют значительно меньше энергии. А уменьшенное количество мультипроцессоров на GPU (8 в отличие от 16 в GF100/GF110) было продиктовано установленными рамками по площади ядра.

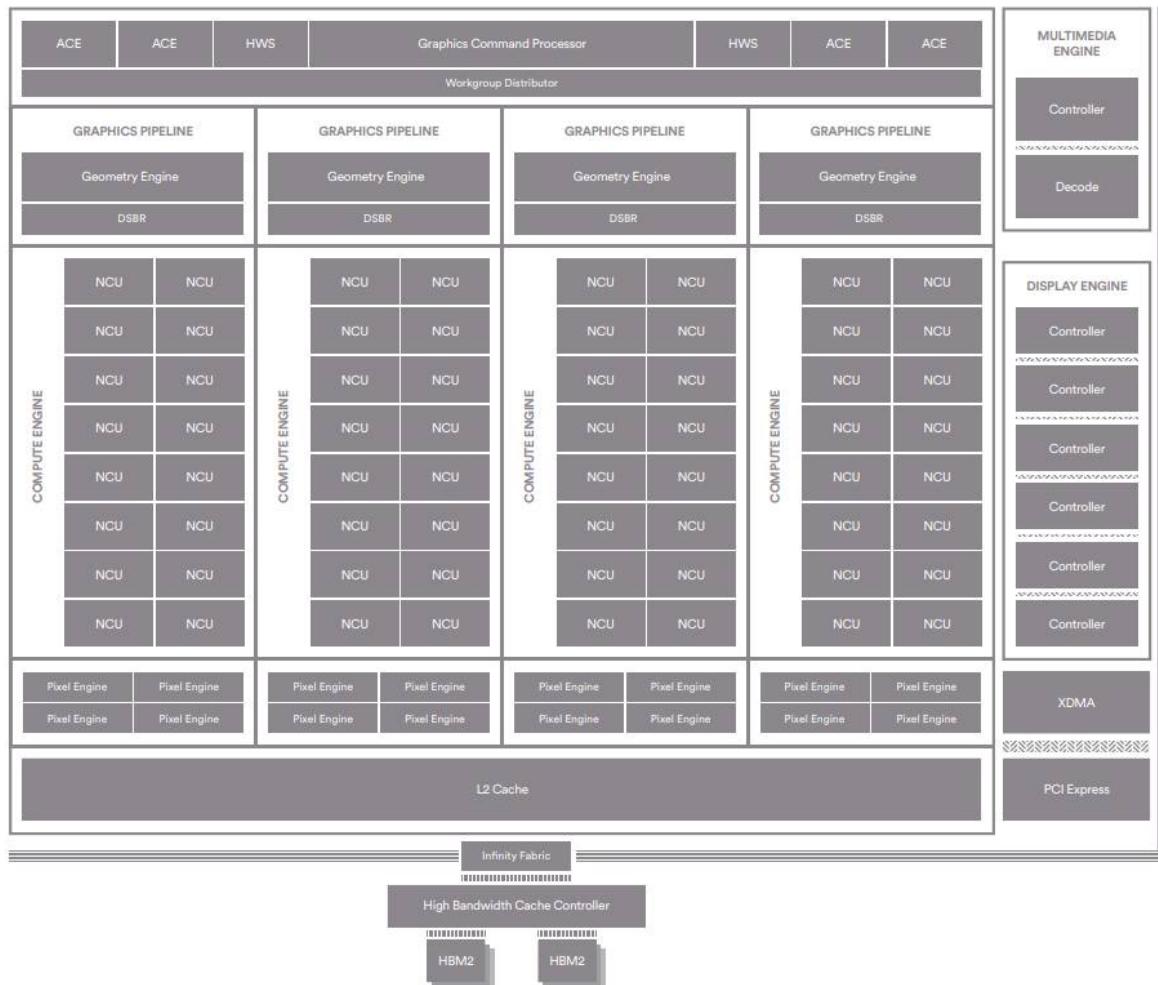
Большая часть ключевых блоков GPU включена в состав SMX: потоковые процессоры (CUDA Cores) выполняют все математические операции над пикселями, вершинами и занимаются неграфическими вычислениями, текстурные модули (TMU) фильтруют текстурные данные, загружают и записывают их из/в видеопамять, блоки специальных функций (Special Function Units, SFU) выполняют сложные операции (вычисление синуса, косинуса, квадратного корня и т.п.) и интерполяции графических атрибутов. Ну а движок PolyMorph обеспечивает выборку вершин, занимается тесселяцией, преобразованием в экранные координаты, установкой атрибутов и потоковым выводом (stream output).

Процессор содержит сложную аппаратную стадию, служащую для предотвращения конфликтов доступа к данным. Специальная таблица регистров (multi-port register scoreboard) отслеживает регистры, данные в которых ещё не готовы, а блок проверки зависимостей (dependency check) анализирует их использование, проверяя зависимости команд.

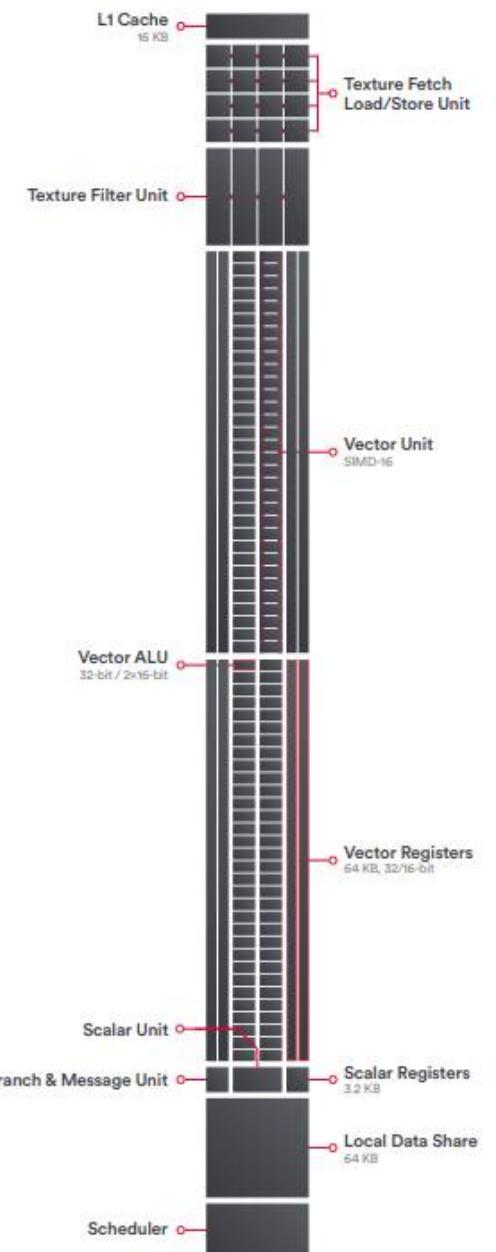


Современные графические процессоры GPU

AMD Radeon VII



Количество универсальных процессоров – 3840; Количество текстурных блоков – 240; Количество блоков блендинга – 64; Эффективная частота памяти - 2000 МГц; Тип памяти – HBM2; Шина памяти – 4096-бит; Объем памяти - 16 ГБ; Пропускная способность памяти - 1 ТБ/с; Количество транзисторов – 13.2 млрд.



Полезные ссылки

- 1) DX Current: Настоящее аппаратного ускорения графики: <https://www.ixbt.com/video2/dx-current.shtml>
- 2) Современная терминология 3D графики: <https://www.ixbt.com/video2/terms2k5.shtml#dm>

XI. Операционные устройства ЭВМ

- Целочисленные арифметико-логические устройства: устройства выполнения логических операций, устройства целочисленного сложения/вычитания, устройства целочисленного умножения, устройства целочисленного деления.

Устройства обработки чисел с плавающей запятой:
устройства сложения/вычитания, устройства умножения, устройства деления, устройства вычисления функций.

- Устройства SSE арифметики.

- Операционные устройства состоят из:

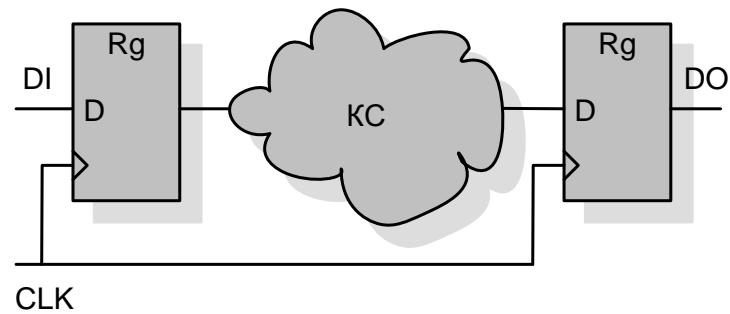
Регистров для хранения

данных;

Шин передачи данных;

Комбинационных схем

реализации функций;



Устройства целочисленного сложения/вычитания

-Накапливающие сумматоры (последовательные).

-Параллельные сумматоры: с последовательным переносом, с параллельным переносом, с условным переносом, с групповой структурой.

Схема последовательного сумматора

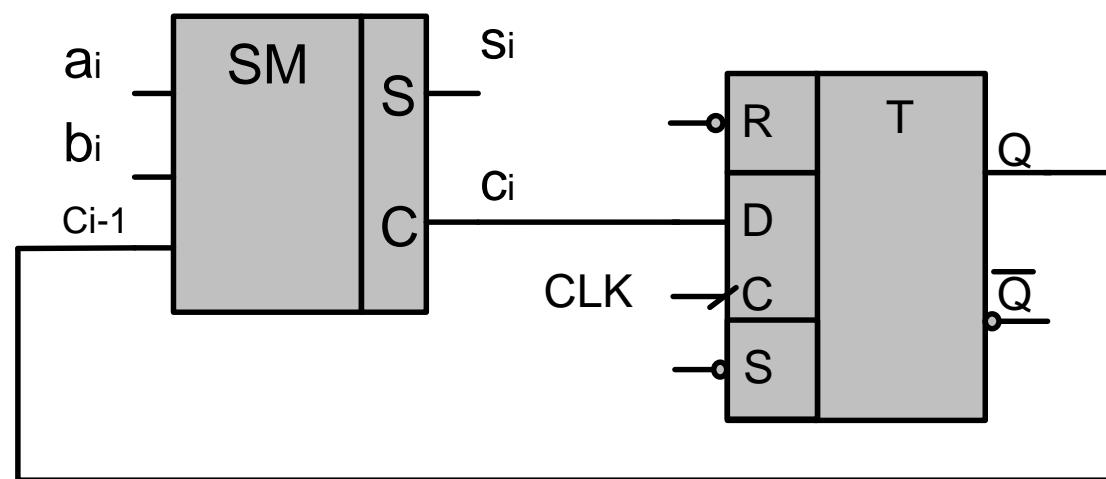


Схема параллельного сумматора с последовательным переносом

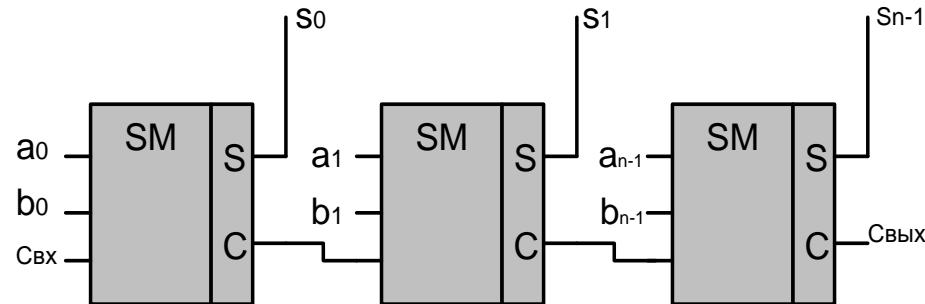
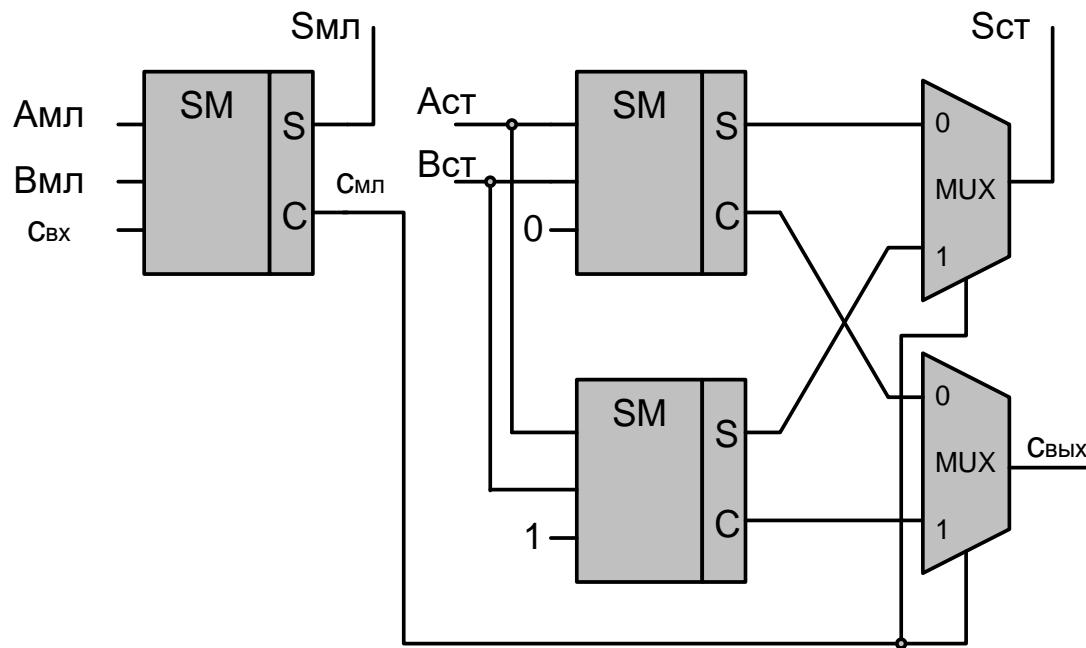


Схема сумматора с условным переносом



Структура блока типа SLICEL

	a	b	LUT	S	C
0	0	0	0	0	ab
0	0	1	1	1	C'
0	1	0	1	1	C'
0	1	1	0	0	ab
1	0	0	0	1	ab
1	0	1	1	0	C'
1	1	0	1	0	C'
1	1	1	0	1	ab

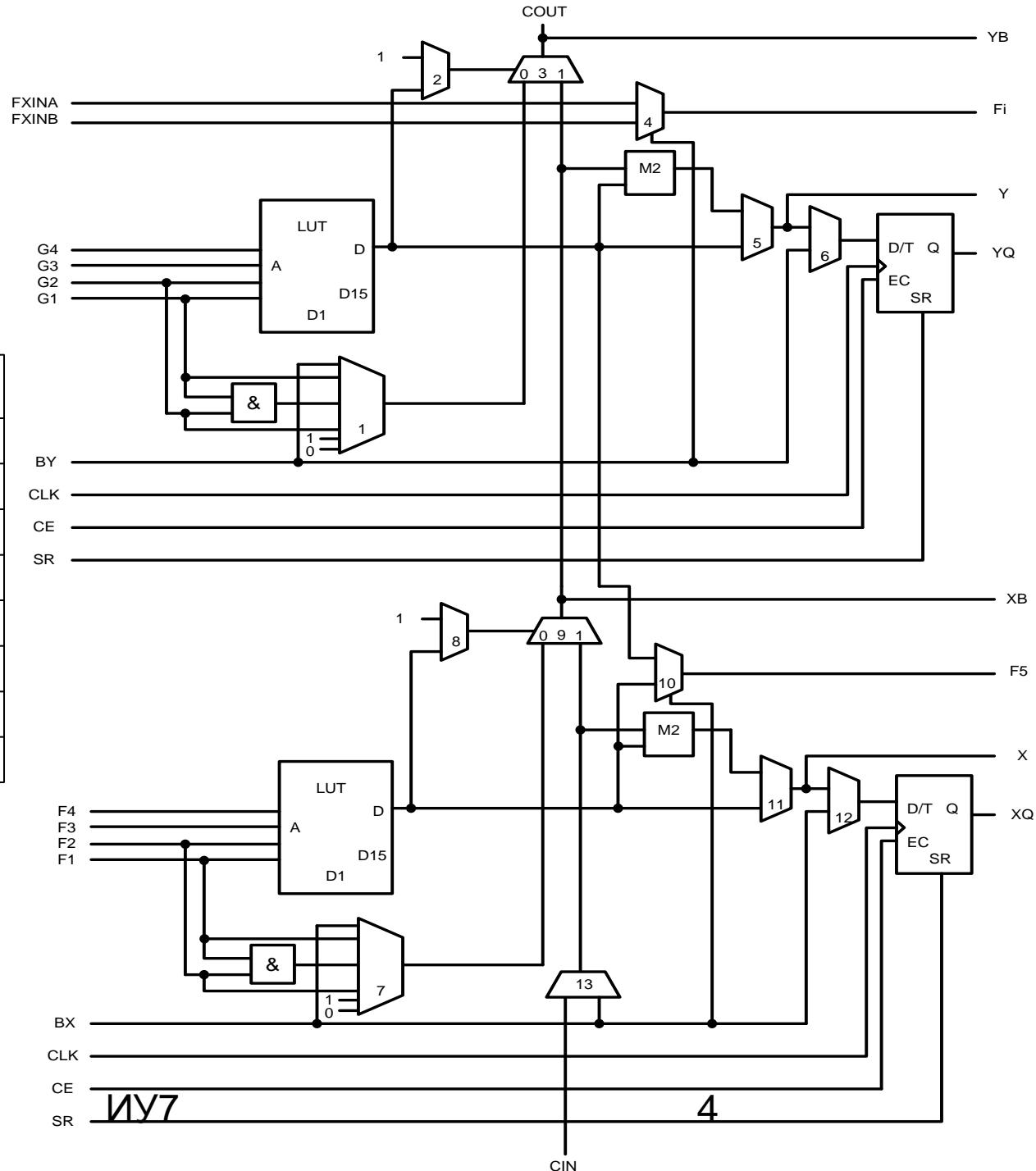
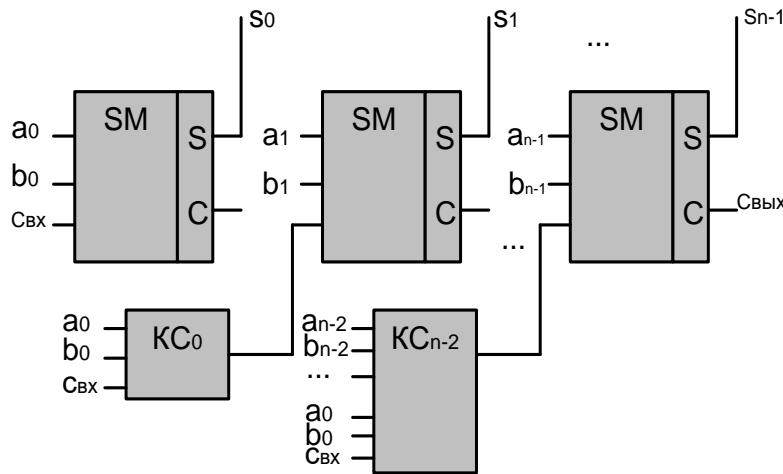
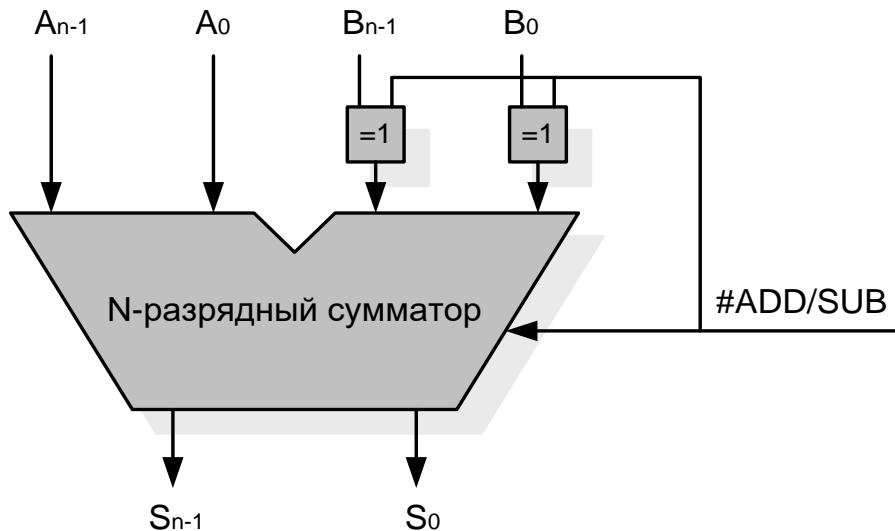


Схема параллельного сумматора с параллельным переносом



Устройство целочисленного сложения/вычитания



Устройства целочисленного умножения

Умножение сводится к последовательному формированию частных произведений и их сложению.

По способу формирования частных произведений:
умножение со старших разрядов множителя со сдвигом влево,
умножение с младших разрядов множителя со сдвигом вправо.

По способу накопления частных произведений: матричные
умножители, древовидные умножители.

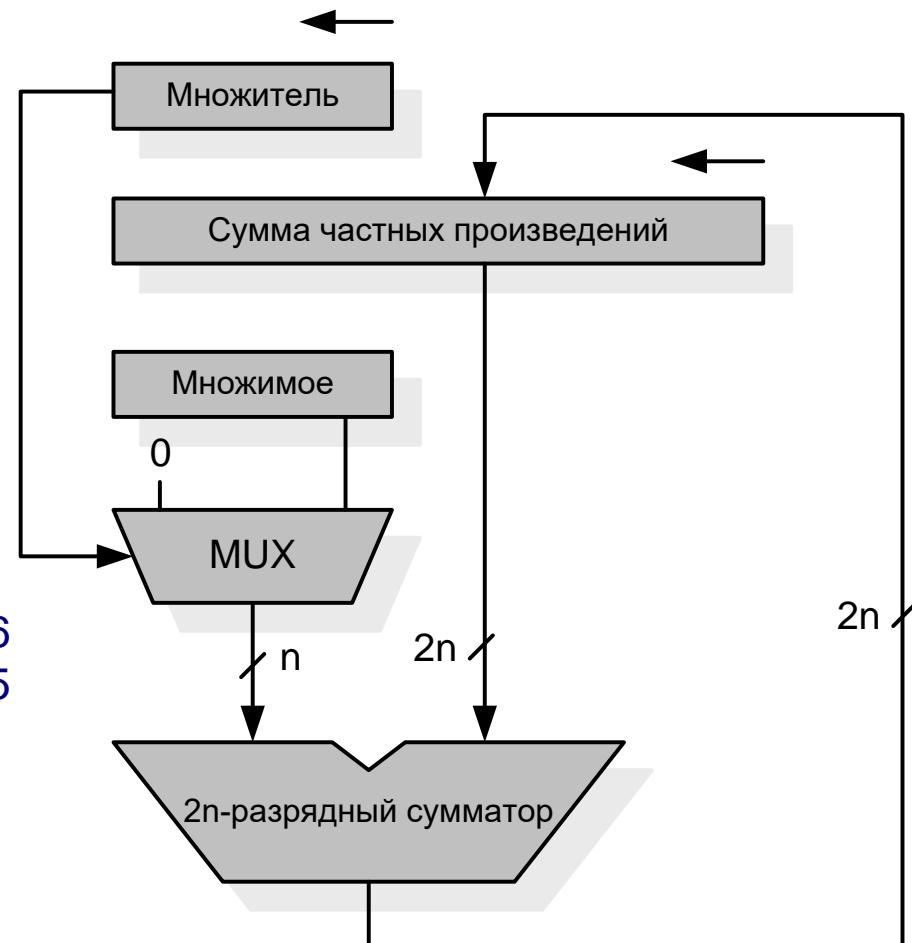
Способы ускорения работы устройств умножения:

- сокращение количества частных произведений;
- обработка нескольких разрядов множителя за такт;
- параллельное вычисление нескольких СЧП;
- конвейеризация умножителей.

Умножение со старших разрядов множителя со сдвигом влево

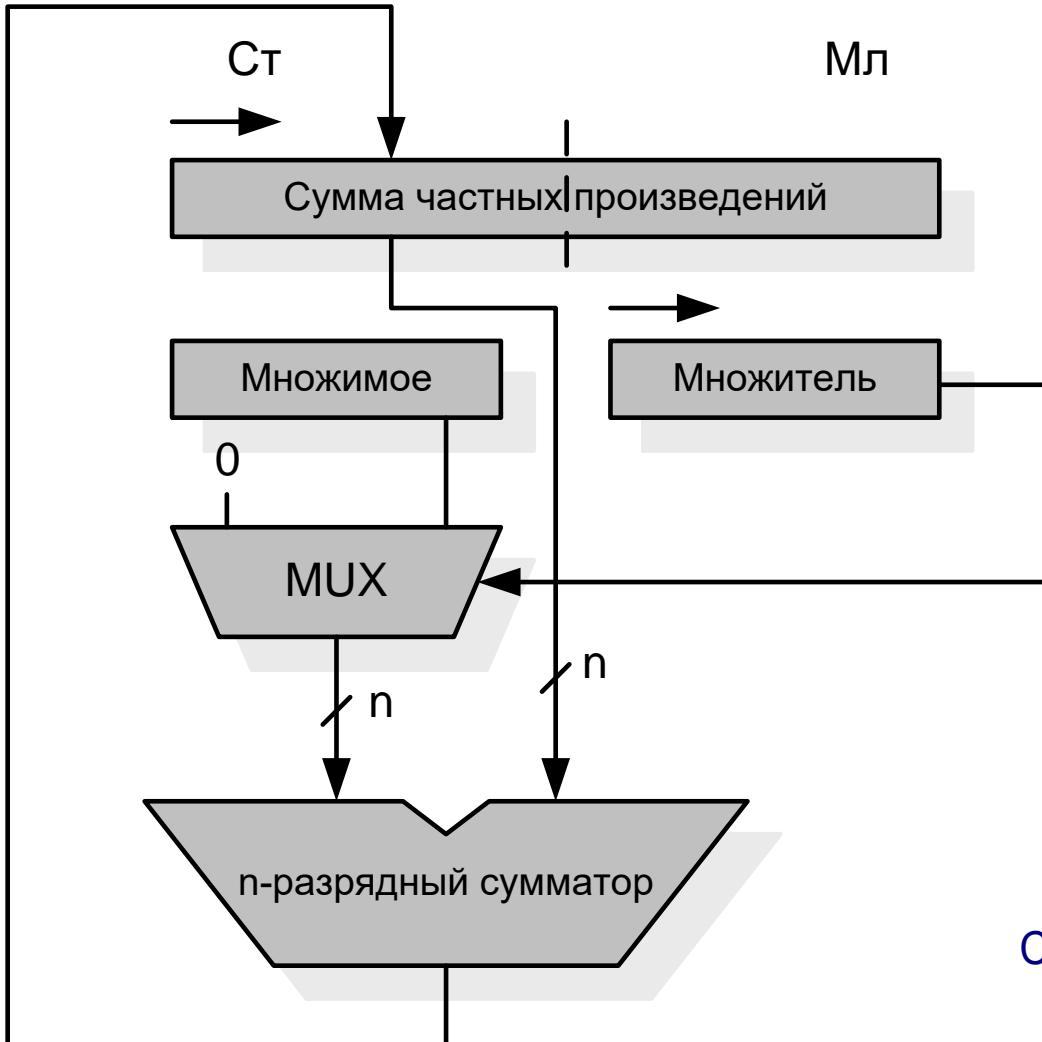
Старший разряд множителя определяет очередное частное произведение (ЧП), которое складывается с накопленной суммой частных произведений (СЧП). После этого СЧП и множитель сдвигаются на один разряд влево.

x	A	1	1	0
	B	1	0	1
ЧП0		1	1	0
СЧП0		1	1	0
<-СЧП0		1	1	0
ЧП1		0	0	0
СЧП1=СЧП0+ЧП1		1	1	0
<-СЧП1		1	1	0
ЧП2		1	1	0
СЧП2=СЧП1+ЧП2		1	1	1



(-) 2-п разрядный сумматор и шины данных.

Умножение с младших разрядов множителя со сдвигом вправо

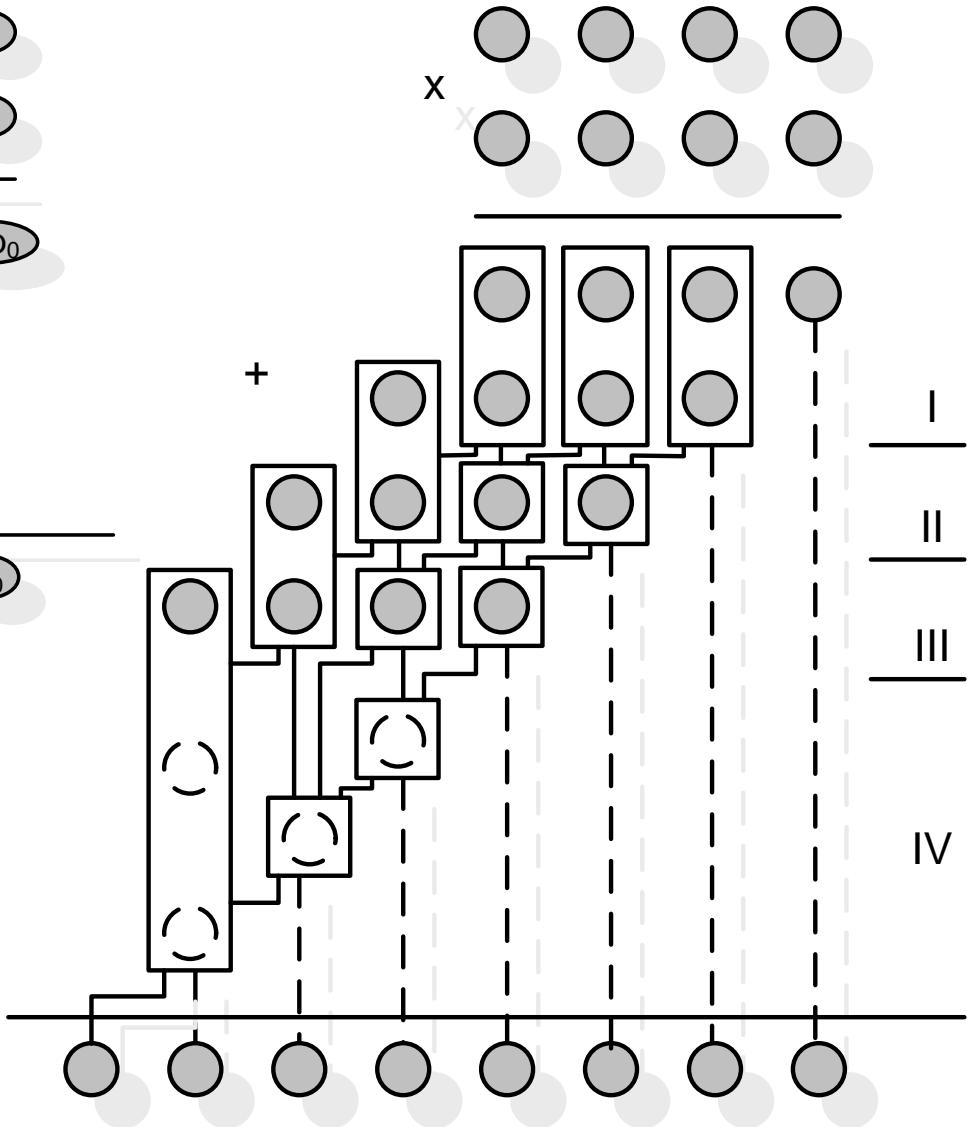
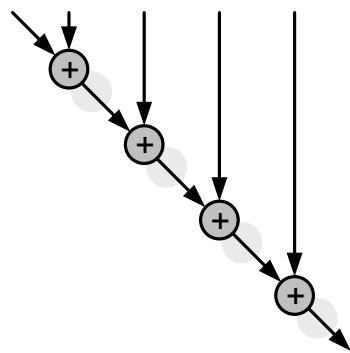
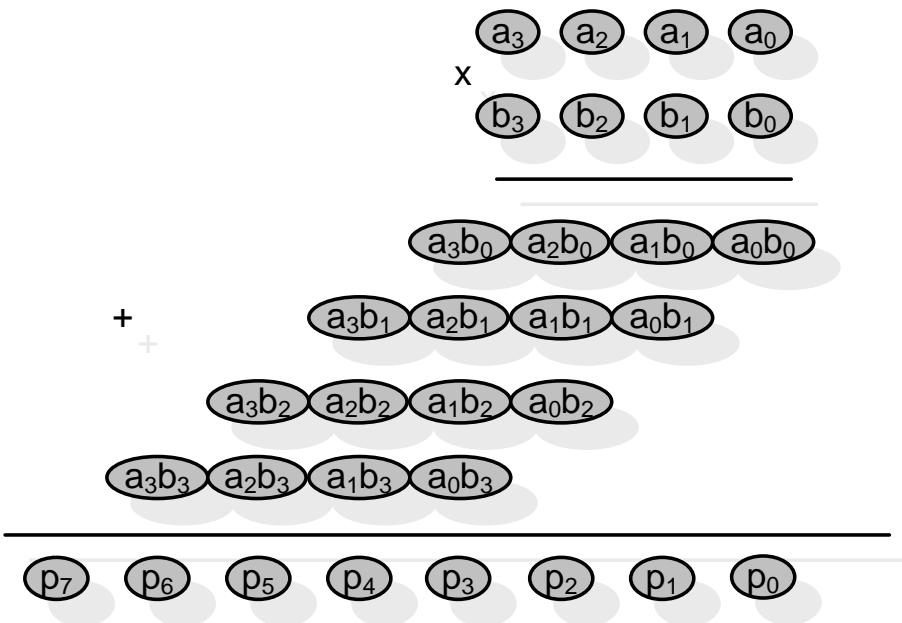


Младший разряд множителя определяет очередное частное произведение (ЧП), которое складывается с накопленной суммой частных произведений (СЧП). После этого СЧП и множитель сдвигаются на один разряд вправо.

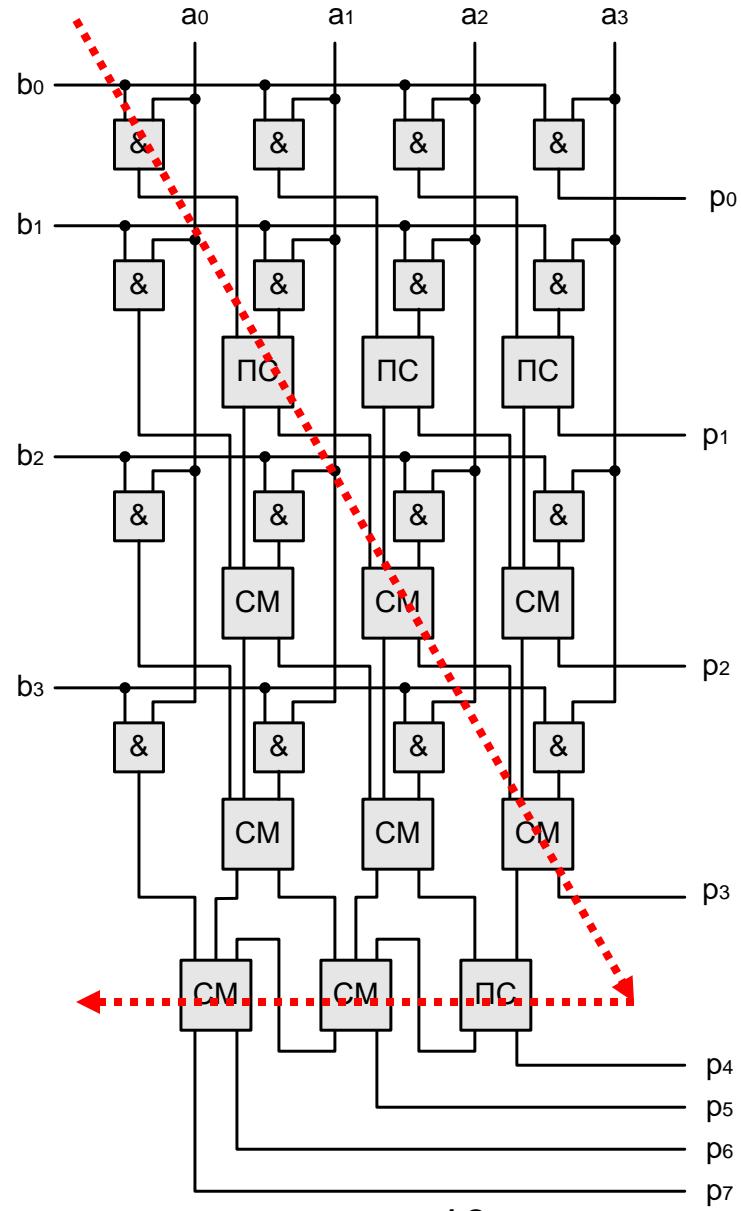
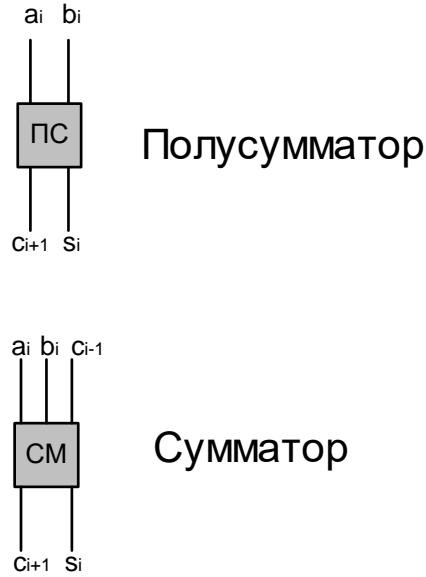
	A	1	1	0	6
x	B	1	0	1	5
ЧП0		1	1	0	
СЧП0		1	1	0	
СЧП0->		0	1	1	0
ЧП1		0	0	0	
СЧП1=СЧП0+ЧП1		0	1	1	0
СЧП1->		0	0	1	1
ЧП2		1	1	0	
СЧП2=СЧП1+ЧП2		1	1	1	0
					30

(+) n-разрядный сумматор и шины данных.

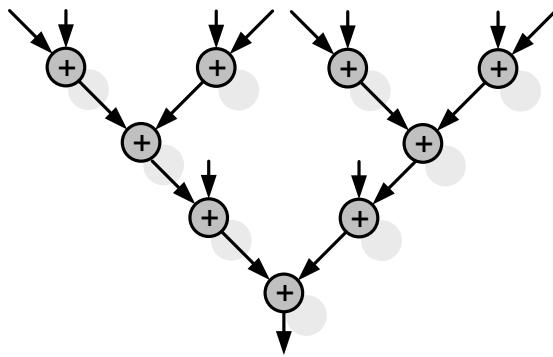
Матричные умножители



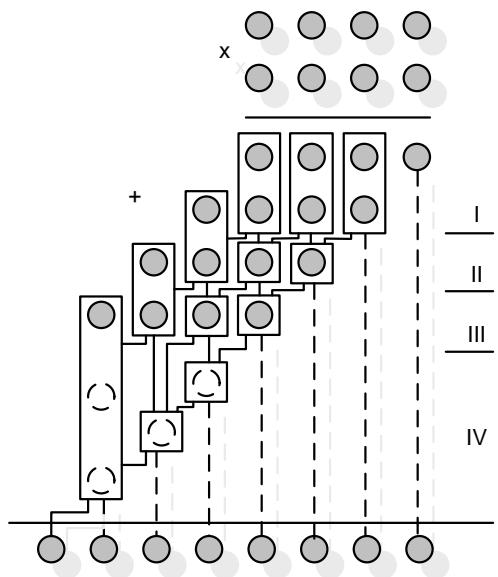
Матричные умножители



Древовидные умножители (схема Уоллеса)

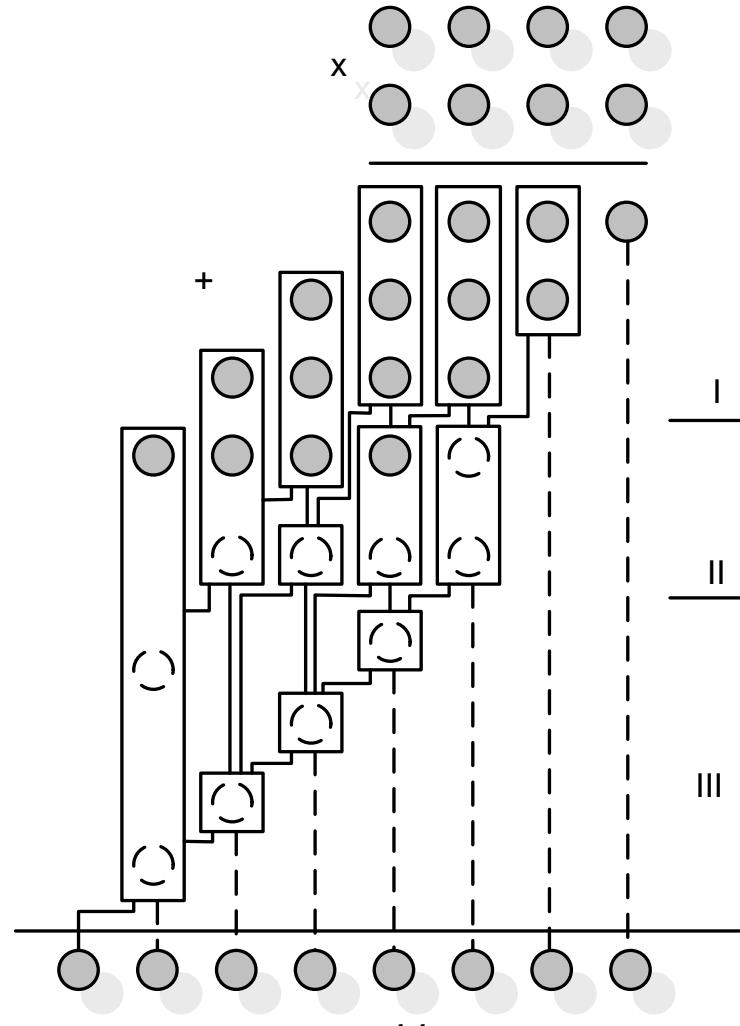


Матричные умножители

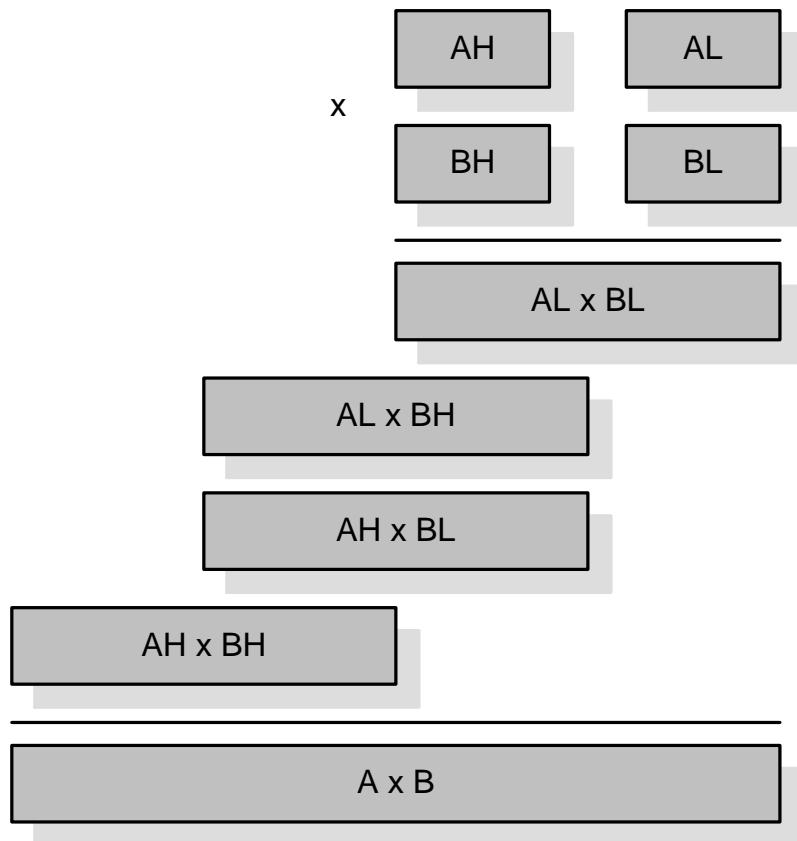


Архитектура ЭВМ

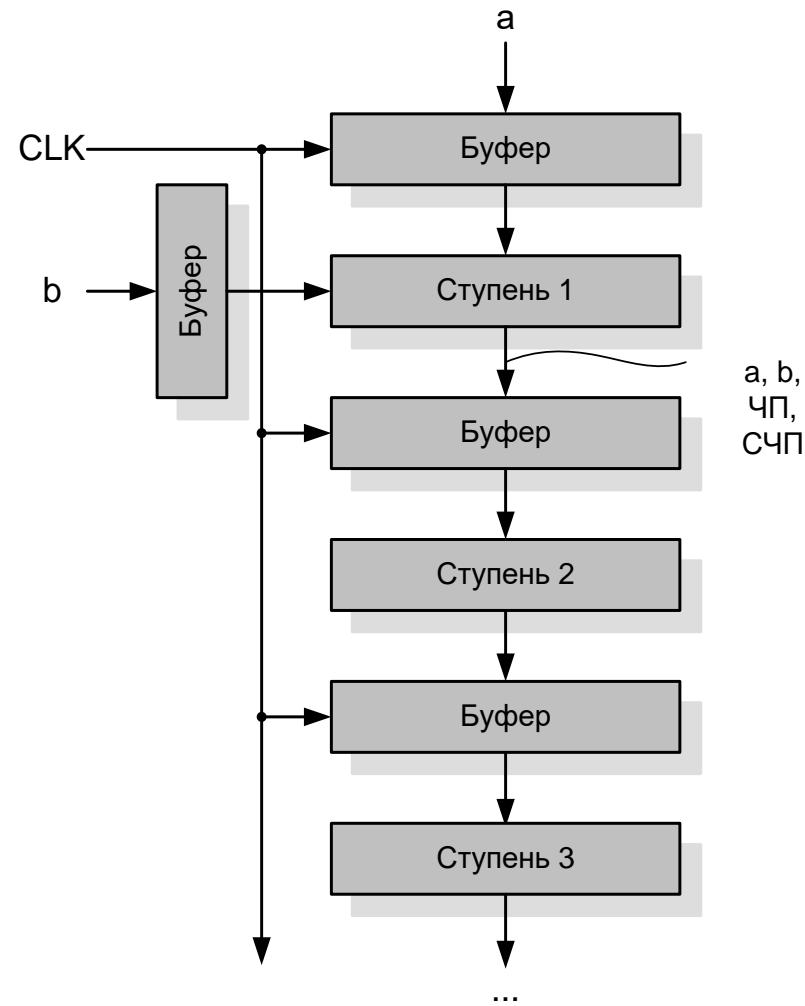
ИУ7



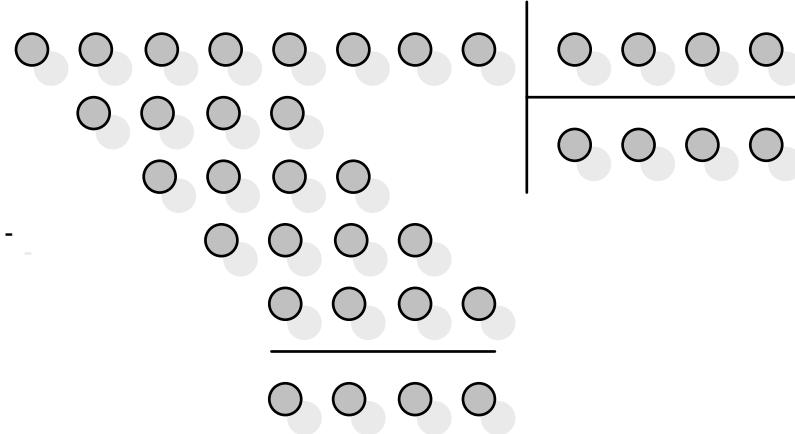
Наращивание размерности умножителей



Конвейеризация умножителей



Устройства целочисленного деления



Деление с восстановлением остатка

	17		
Делимое	0 1 0 0 0 1	0 1 1	3
ЧО	0 1 0 0 0 1	1 0 1	5
<-ЧО	0 1 0 0 0 1 0		
-Делитель	0 1 1		
ЧО>0	0 0 0 1 0 1 0		
<-ЧО	0 0 0 1 0 1 0 0		
-Делитель	0 1 1		
ЧО<0	1 1 1 1 1 1 0 0		
+Делитель	0 1 1		
Восст. ЧО	0 0 0 1 0 1 0 0		
<-ЧО	0 0 0 1 0 1 0 0 0		
-Делитель	0 1 1		
ЧО>0	0 0 0 0 1 0 0 0 0		

Алгоритм:

- 1) ЧО = Делимое;
- 2) ЧО = ЧО*2;
- 3) ЧО = ЧО – Делитель * 2ⁿ;
- 4) Если ЧО<0 то
Ч<-0,
ЧО = ЧО + Делитель * 2ⁿ
иначе Ч<-1;
- 5) Если все цифры то конец
иначе пункт 2.

Деление без восстановления остатка

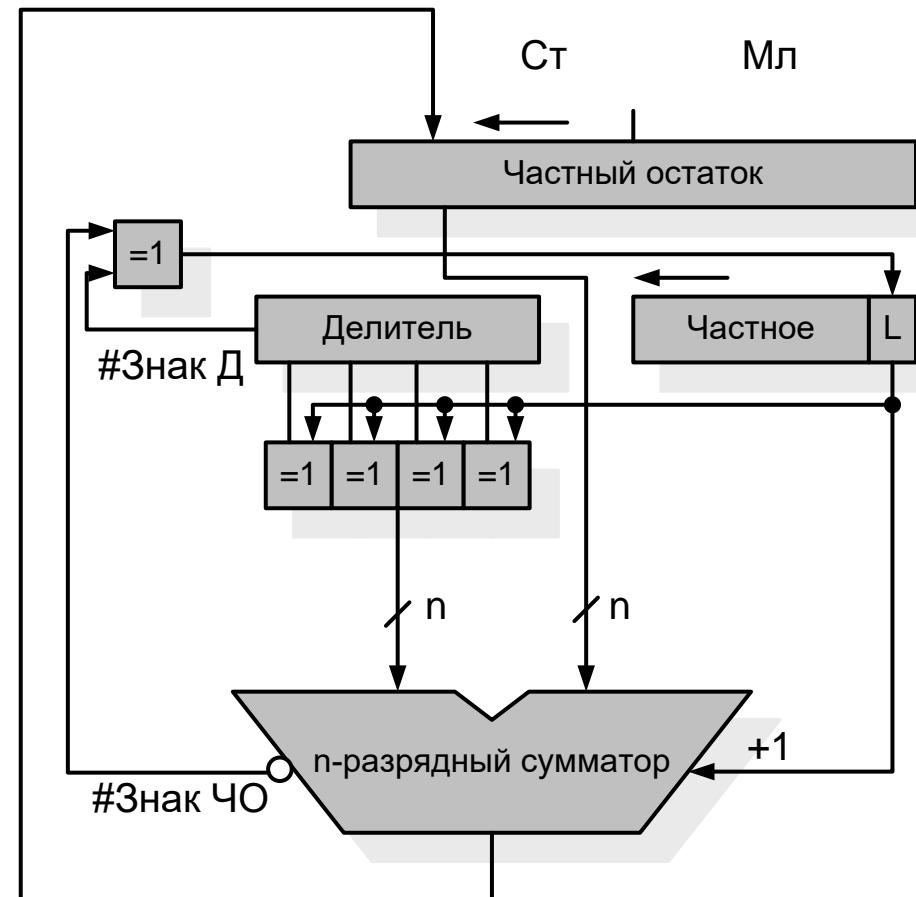
	17		
Делимое	0 1 0 0 0 1 0 1 1	3	
ЧО	0 1 0 0 0 1 1 0 1	5	
<-ЧО	0 1 0 0 1 0		
-Делитель	0 1 1		
ЧО>0	0 0 1 0 1 0		
<-ЧО	0 0 0 1 0 1 0 0		
-Делитель	0 1 1		
ЧО<0	1 1 1 1 1 1 0 0		
<-ЧО	1 1 1 0 0 0		
+Делитель	0 1 1		
ЧО>0	0 0 0 0 1 0 0 0 0		

Алгоритм:

- 1) ЧО = Делимое * 2;
 - 2) ЧО = ЧО – Делитель * 2^n ;
 - 3) ЧО = ЧО * 2;
 - 4) Если ЧО<0 то
Ч<0,
ЧО = ЧО + Делитель * 2^n
- иначе
- Ч<-1;
ЧО = ЧО – Делитель * 2^n

- 5) Если все цифры то конец
иначе пункт 3.

Схема АЛУ для целочисленного деления

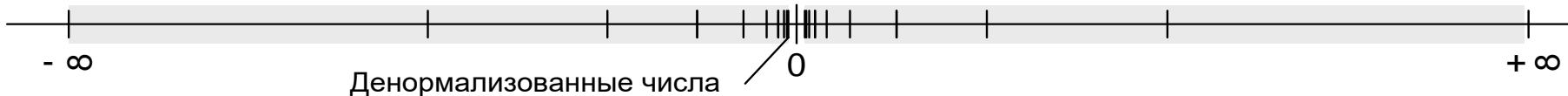


Форматы представления чисел с плавающей запятой (по стандарту IEEE 754 и 784).

Мантисса М числа представляется в нормализованном виде (старший разряд не сохраняется).

Формат	Длина числа	Длина мантиссы	Длина порядка	Смещение порядка	Диапазон чисел
Короткий формат	32	24	8	+127	$10^{-38}..10^{+38}$
Длинный формат	64	53	11	+1023	$10^{-308}..10^{+308}$
Расширенный формат	80	64	15	+16383	$10^{-4932}..10^{+4932}$

Специальные числовые значения.



Переполнение

Потеря значимости

Переполнение

Числовые значения

The diagram illustrates the floating-point representation of a number. It consists of three main parts: a sign bit ('Знак M') with a value of '+Ноль' above it; a fraction ('Порядок + C') with a value of '0 00000000' above it; and a normalized mantissa ('Нормализованная мантисса') with a value of '00000...0' above it.

1	00000000	0000...0
Знак	Порядок	Нормализованная
M	+ C	мантисса

Diagram illustrating the floating-point representation of a number:

- Sign**: 0 (leftmost)
- Exponent**: 11111111 (middle, underlined)
- Mantissa**: 00000...0 (rightmost, underlined)
- Normalizing factor**: 1 (red arrow pointing to the first digit of the mantissa)

-Бесконечность

1 11111111 00000...0

Знак Порядок Нормализованная мантисса

M + C 1

“Нечисла”

Денормализованное число >0

Денормализованное число <0

The diagram illustrates the structure of a floating-point number. It consists of three main parts: a sign bit labeled 'Знак' (Sign) with value 'M'; a 10-bit exponent labeled 'Порядок + C' (Exponent); and a 10-bit mantissa labeled 'Мантисса' (Mantissa). The mantissa is shown as '1xx...1...x'. A red arrow points to the first digit of the mantissa, which is '1', indicating the normalized form of the number.

Операции над числами с плавающей запятой.

1. Подготовительный этап.

- Разделение упакованного ЧПЗ на группы М,П,З.
- Проверка на специальное числовое значение.

2. Выполнение операции.

- Приведение порядков.
- Определение знака результата.
- Определение мантиссы результата.
- Определение порядка результата.
- Проверка на переполнение, потери значимости мантиссы, потери значимости порядка, неточности, деления на 0.

3. Заключительный этап.

- Проверка на специальное числовое значение.
- Нормализация результата.
- Проверка на переполнение, потери значимости мантиссы, потери значимости порядка, неточности.
- Упаковка полей З,П,М в ЧПЗ.

Организация операций сложения и вычитания над числами с плавающей запятой.

1. Подготовительный этап
2. Определение меньшего из двух порядков и проведение операции выравнивания порядков (сдвиг вправо на разность порядков).
3. Проверка на потерю значимости одного операнда (неточность).
4. Определение результирующего порядка как максимума.
5. Сложение мантисс и определение знака результата.
6. Проверка на переполнение мантиссы. Если да, то сдвигаем мантиссу вправо и увеличиваем порядок на 1.
7. Проверка на переполнение порядка.
8. Заключительный этап.

Организация операций умножения чисел с плавающей запятой.

1. Подготовительный этап
2. Проверка ($M_1=0$ или $M_2=0$). Если да, то $R=0$.
3. Определение порядка результата: $Pr = P_1 + P_2 - C$.
4. Проверка на переполнение порядка.
5. Определение мантиссы результата: $M_r = M_1 * M_2$.
6. Определение знака результата.
7. Заключительный этап.

Организация операций деления чисел с плавающей запятой.

1. Подготовительный этап
2. Проверка ($M1=0$ или $M2=0$). Если деление на ноль, то +/-бесконечность или ошибка.
3. Определение порядка результата: $Pr = P1-P2+C$.
4. Проверка на переполнение порядка.
5. Определение мантиссы результата: $Mp = M1*(1/M2)$.
6. Определение знака результата.
7. Заключительный этап.

TABLE 6-5 Floating-Point Multiplication

MULTIPLICATION Instruction FMUL $rs_1, rs_2 / rs_2, rs_3 \rightarrow rd$	Result from the operation includes one or more of the following:			
	Masked Exception, TEM = 0		Enabled Exception, TEM = 1	
	Destination Register Written (rd)	Flag(s)	Destination Register Written (rd)	Flag(s), Trap
+0, [+0]+Normal]	+0	None set.	+0	None set.
+0, [-0]-Normal]	-0	None set.	-0	None set.
-0, [+0]+Normal]	-0	None set.	-0	None set.
-0, [-0]-Normal]	+0	None set.	+0	None set.
+0, +Infinity	QNaN	Asserts nvc, nva.	No	Asserts nvc. IEEE trap ¹ enabled.
+0, -Infinity	QNaN	Asserts nvc, nva.	No	Asserts nvc. IEEE trap enabled.
-0, +Infinity	QNaN	Asserts nvc, nva.	No	Asserts nvc. IEEE trap enabled.
-0, -Infinity	QNaN	Asserts nvc, nva.	No	Asserts nvc. IEEE trap enabled.
±Normal, ±Normal	Can underflow/ overflow. See 6.5.		Can underflow/ overflow. See 6.5.	
[+Normal]+Infinity], +Infinity	+Infinity	None set.	+Infinity	None set.
[+Normal]+Infinity], -Infinity	-Infinity	None set.	-Infinity	None set.
[-Normal]-Infinity], +Infinity	-Infinity	None set.	-Infinity	None set.
[-Normal]-Infinity], -Infinity	+Infinity	None set.	+Infinity	None set.

1.IEEE trap means *fp_exception_IEEE_754*.

TABLE 6-3 Floating-Point Addition

ADDITION Instruction	Result from the operation includes one or more of the following:			
	Masked Exception, TEM = 0		Enabled Exception, TEM = 1	
	Destination Register Written (rd)	Flag(s)	Destination Register Written (rd)	Flag(s), Trap
+0, +0	+0	None set.	+0	None set.
+0, -0	+0 (FSR.RD = 0,1,2) -0 (FSR.RD = 3)	None set.	+0 (FSR.RD = 0,1,2) -0 (FSR.RD = 3)	None set.
-0, -0	-0	None set.	-0	None set.
±0, +Normal	+Normal	None set.	+Normal	None set.
±0, -Normal	-Normal	None set.	-Normal	None set.
±0, +Infinity	+Infinity	None set.	+Infinity	None set.
±0, -Infinity	-Infinity	None set.	-Infinity	None set.
±Normal, +Infinity	+Infinity	Asserts ofc, ofa, nvc, nva.	No	Asserts ofc, nvc. IEEE trap ¹ enabled.
±Normal, -Infinity	-Infinity	Asserts ofc, ofa, nvc, nva.	No	Asserts ofc, nvc. IEEE trap enabled.
+Normal, +Normal	Can overflow. See 6.5.3.		Can overflow. See 6.5.3.	
+Normal, -Normal	±Normal		Normal	
-Normal, +Normal	±Normal		Normal	
-Normal, -Normal	Can underflow. See 6.5.4.		Can underflow. See 6.5.4.	
+Infinity, +Infinity	+Infinity	None set.	+Infinity	None set.
+Infinity, -Infinity	QNaN	Asserts nvc, nva.	No	Asserts nvc. IEEE trap enabled.
-Infinity, +Infinity	QNaN	Asserts nvc, nva.	No	Asserts nvc. IEEE trap enabled.
-Infinity, -Infinity	-Infinity	None set.	-Infinity	None set.

1.IEEE trap means *fp_exception_IEEE_754*.

TABLE 6-6 Floating-Point Division

DIVISION Instruction $rs_1\ rs_2$ FDIV $rs_1, rs_2 \rightarrow rd$	Result from the operation includes one or more of the following:			
	Masked Exception, TEM = 0		Enabled Exception, TEM = 1	
	Destination Register Written (rd)	Flag(s)	Destination Register Written (rd)	Flag(s), Trap
$\pm 0, \pm 0$	sign=0, expo=111...111, frac=111...111 (QNaN)	Asserts nvc, nva.	No	Asserts nvc. IEEE trap ¹ enabled.
$\pm 0, \pm$ Normal	± 0	None set.	± 0	None set.
$\pm 0, \pm$ Infinity	± 0	None set.	± 0	None set.
+Normal, +0	+Infinity	Asserts nvc, nva.	No	Asserts dzc, nvc. IEEE trap enabled.
+Normal, -0	-Infinity	Asserts nvc, nva.	No	Asserts dzc, nvc. IEEE trap enabled.
-Normal, +0	-Infinity	Asserts nvc, nva.	No	Asserts dzc, nvc. IEEE trap enabled.
-Normal, -0	+Infinity	Asserts nvc, nva.	No	Asserts dzc, nvc. IEEE trap enabled.
\pm Normal, \pm Normal	Can underflow/overflow. See 6.5.		Can underflow/overflow. See 6.5.	
\pm Infinity, \pm Infinity	QNaN	Asserts nvc, nva.	No	Asserts nvc. IEEE trap enabled.
+Infinity, +Normal	+Infinity	None set.	+Infinity	None set.
+Infinity, -Normal	-Infinity	None set.	-Infinity	None set.
-Infinity, +Normal	-Infinity	None set.	-Infinity	None set.
-Infinity, -Normal	+Infinity	None set.	+Infinity	None set.

1.IEEE trap means *fp_exception_IEEE_754*.

Устройства выполнения векторных операций (Эльбрус1,Intel,AMD,Sun,IBM,MIPS).

Устройство выполнения целочисленных MMX операций (MultiMedia eXtensions, Intel) и SSE операций (Streaming SIMD Extension) предназначены для ускорения приложений, ориентированных на выполнение однотипных действий с большими массивами целочисленных и вещественных данных. С данными такого типа обычно работают мультимедийные, графические и коммуникационные программы.

Операнды MMX и SSE операций упакованы в группы по 32,64,80,128 разрядов. Выполнение арифметических операций над операндами группы выполняются параллельно.

Технология SIMD (ИТМ и ВТ, Эльбрус 1) 1978 год

Технология MMX (Intel Pentium MMX, Intel P6, ...) 1992 год

Технология SSE (Intel P6, Intel NetBurst, ...)

Технология 3DNow (AMD K6, ...)

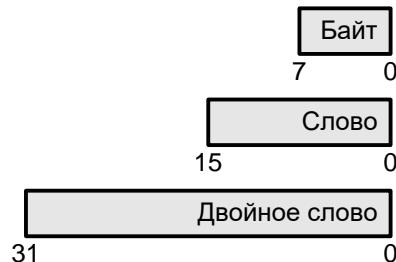
Технология AltiVec (IBM PowerPC)

Технология VIS (Sun UltraSPARC II)

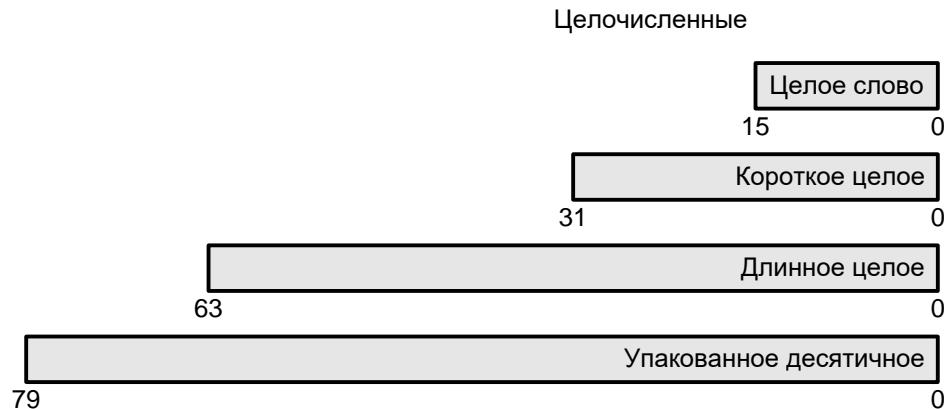
Технология ASE (MIPS 24KE, 74K)

Форматы чисел в микропроцессорах Intel, AMD

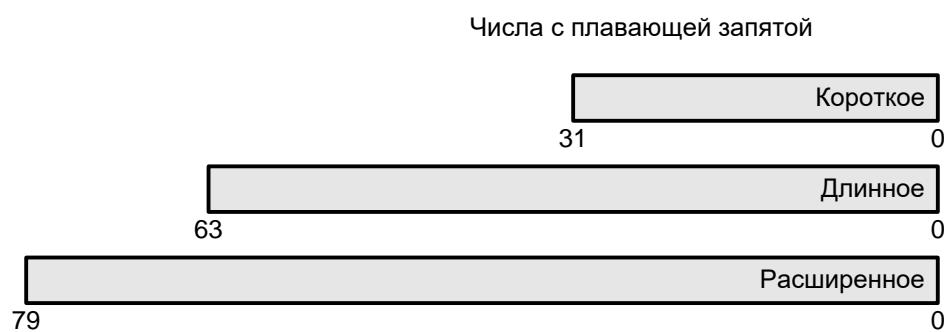
Форматы целочисленного АЛУ



Форматы блока обработки ЧПЗ



Форматы MMX

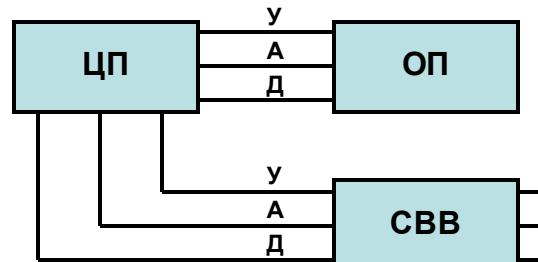


Форматы SSE



XI. Организация ввода-вывода

Совокупность технических и программных средств, обеспечивающих обмен данными между центральным процессором и внешними устройствами называется системой ввода вывода.

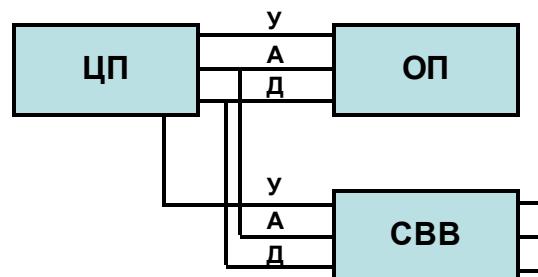


Шины полностью независимы.

(+) Обмен можно осуществлять параллельно.

(+) Каждую шину можно оптимизировать независимо.

(-) Увеличивается количество выводов ЦП.

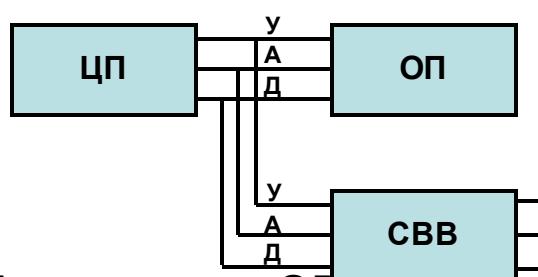


Совместно используются ШД и ША. ШУ независимы.

(+) Управление обменом выполняется параллельно.

(+) Экономично используются выводы.

(-) Одновременный обмен данными невозможен.



Совместно используются ШД и ША и ШУ.

(+) Минимальные аппаратные затраты.

(-) Одновременная работа ЦП с СВВ и с ОП невозможна.

Способы адресации внешних устройств

Для выбора конкретного ВУ используются адреса.

Адресное пространство ОП и СВВ может быть совмещенным и раздельным.

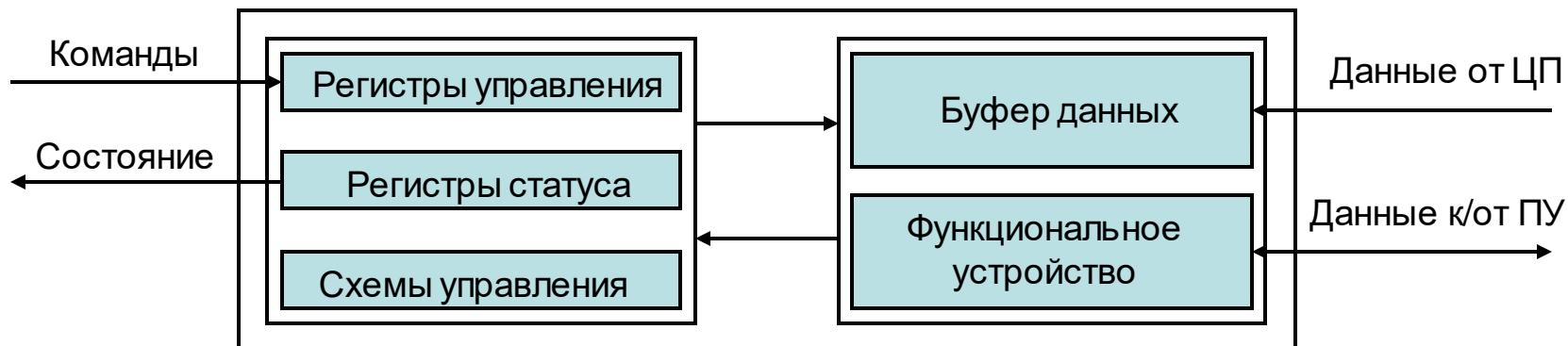
При совмешенном адресном пространстве обращение за чтением и записью в некоторый диапазон адресов приводит к обращению в СВВ.

- (+) Возможное количество ВУ может быть велико. Нет жестких ограничений на объем адресуемых портов ввода вывода (гибкость СВВ).
- (+) Команды для доступа к внешним устройствам не отличаются от команд доступа к ОП.
- (-) Потребность в дополнительных средствах декодирования адресов СВВ.
- (-) Сложность организации виртуальной памяти и поддержки когерентности кэш-памяти.
- (-) Непредсказуемость времени выполнения простых команд.
- (-) Сокращение адресного пространства ОП.

При раздельном адресном пространстве используются отдельные команды при обращении к ОП и СВВ.

- (+) Команды ввода/вывода короткие.
 - (+) Легкость дешифрации запросов к СВВ для ЦП.
 - (+/-) Изменение СВВ не затрагивает память
 - (-) Малое количество способов адресации и обработки информации (Аккумулятор - СВВ).
 - (-) Сложность работы с быстродействующими устройствами и устройствами с большим количеством адресуемой памяти.
- Устройство вычислительной машины, осуществляющее связь ЦП и ПУ называется модулем ввода-вывода.

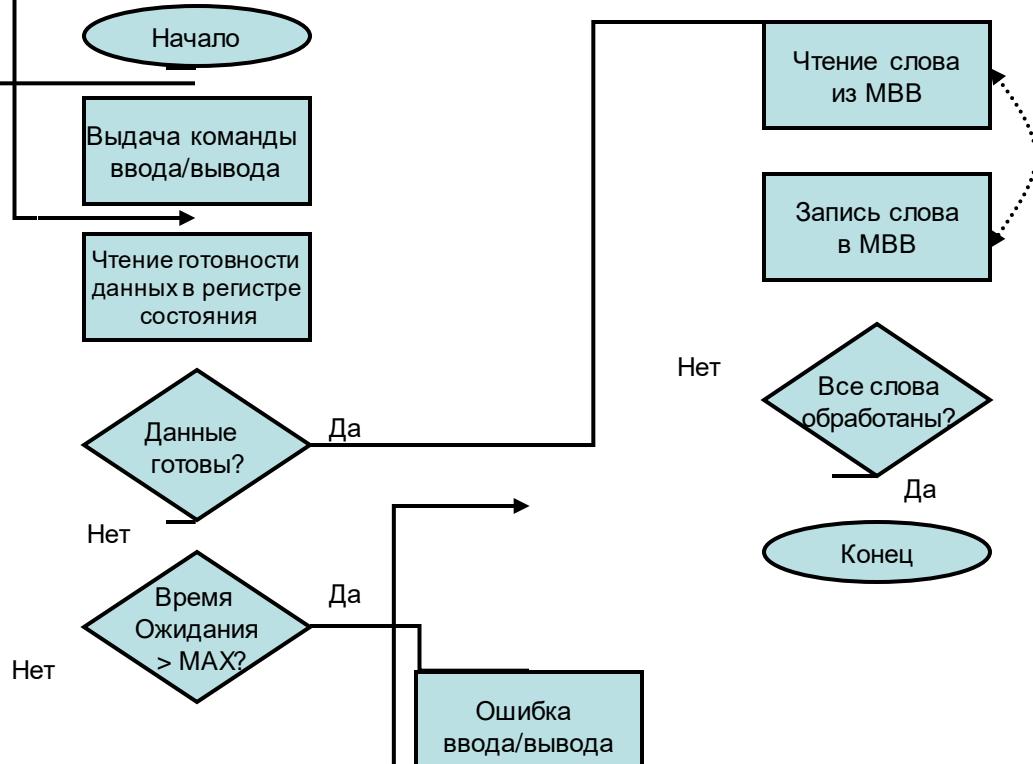
Обобщенная схема МВВ



Методы управления вводом/выводом

- Программно-управляемый ввод/вывод.
- Ввод/вывод по прерыванию.
- Прямой доступ к памяти.

Программно-управляемый ввод/вывод.

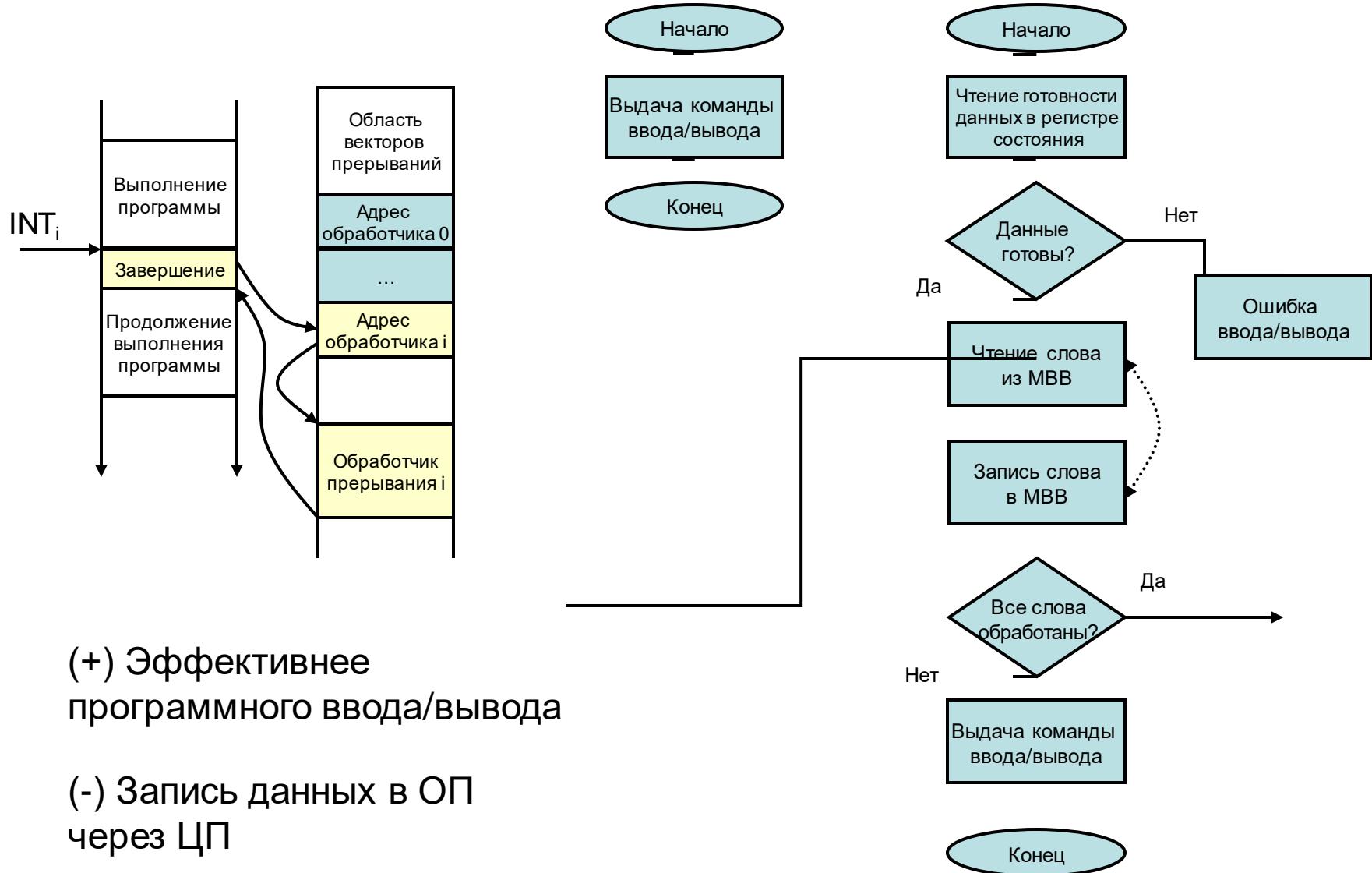


(+) Минимальные аппаратные затраты

(+) Простота изменения процедур ввода/вывода

(-) Простой ЦП из-за ожидания готовности МВВ к передаче.

Ввод/вывод по прерыванию.



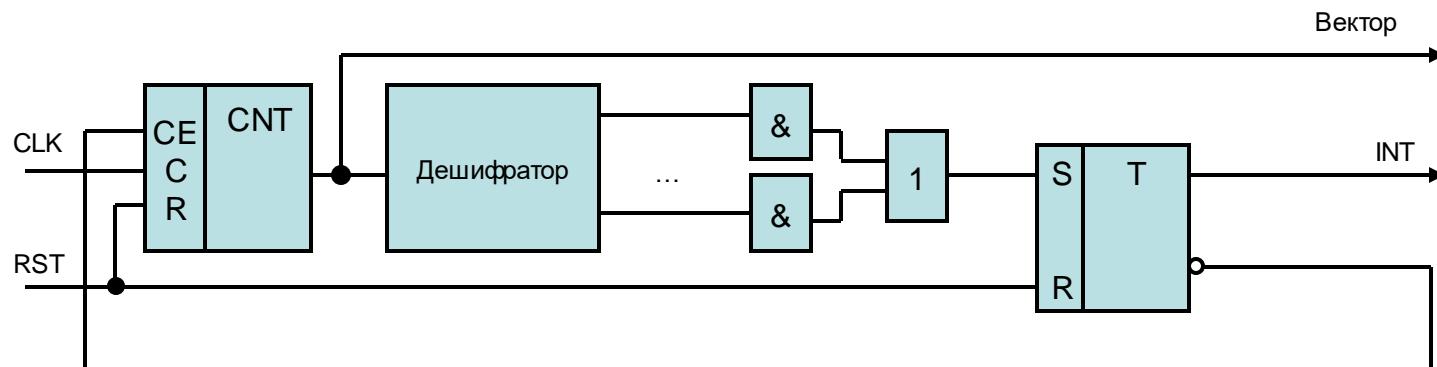
(+) Эффективнее
программного ввода/вывода

(-) Запись данных в ОП
через ЦП

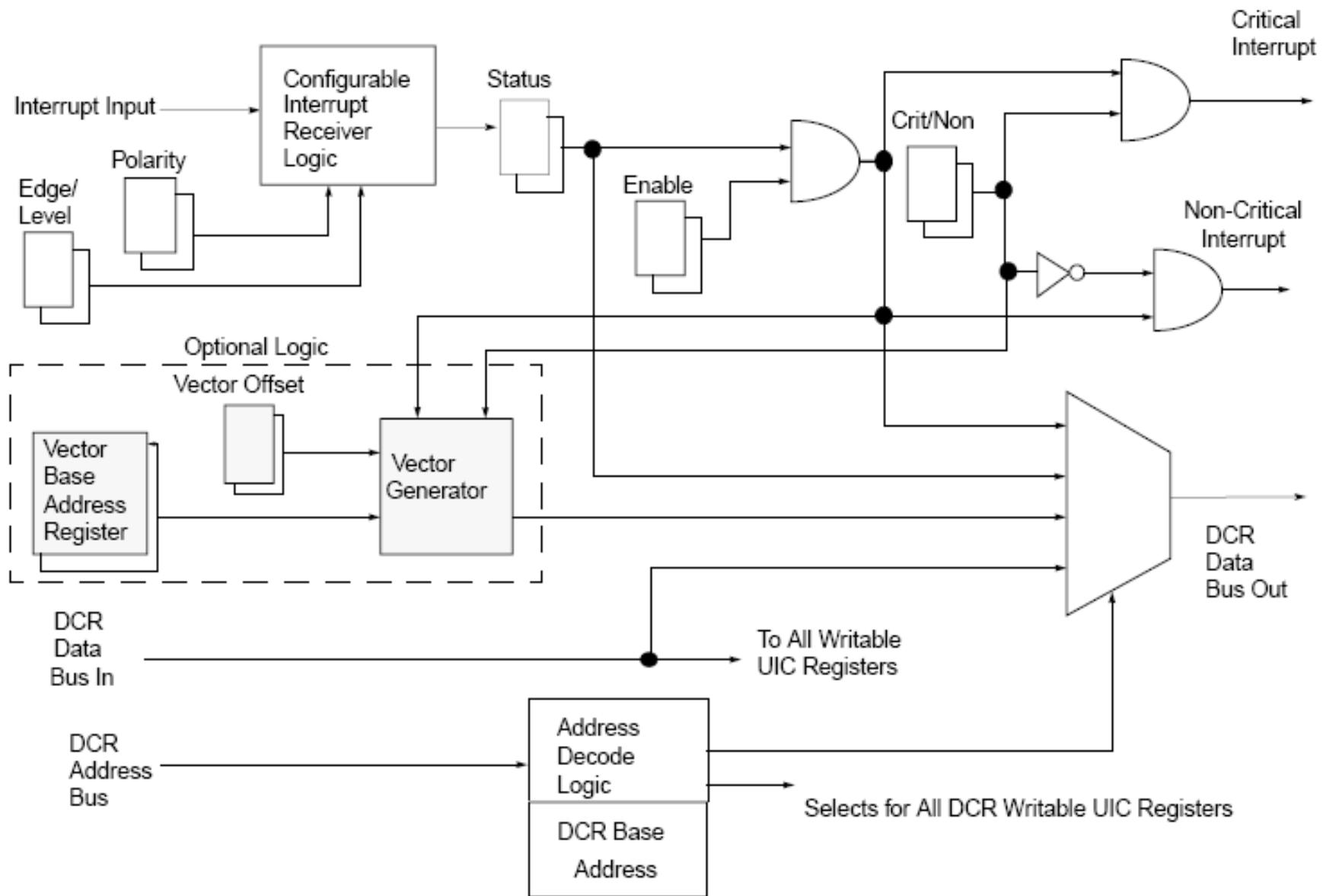
Способы определения адреса источника прерывания.

- По номеру линии прерывания (необходимо много линий).
- Программным опросом всех МВВ (необходимо много времени на обработку).
- Векторизация прерываний (необходимо использовать устройство для передачи запросов прерываний).

Схема циклического опроса



Универсальный контроллер прерываний (IBM PowerPC405 UIC).

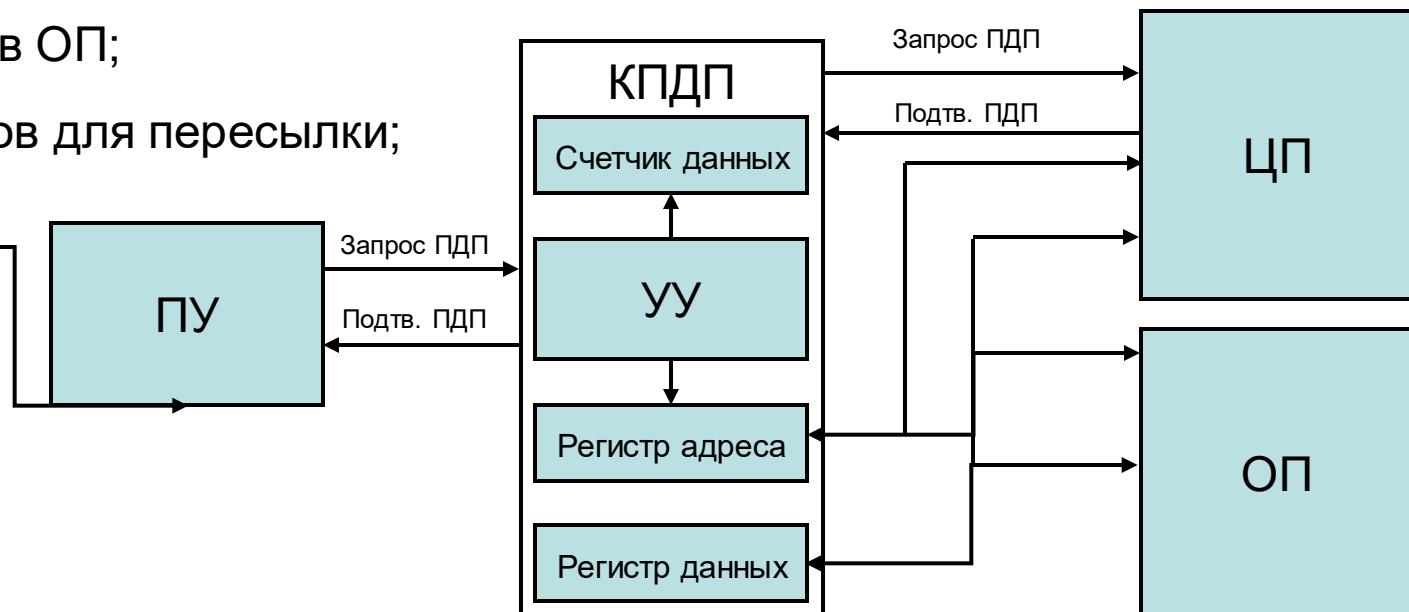


Прямой доступ к памяти

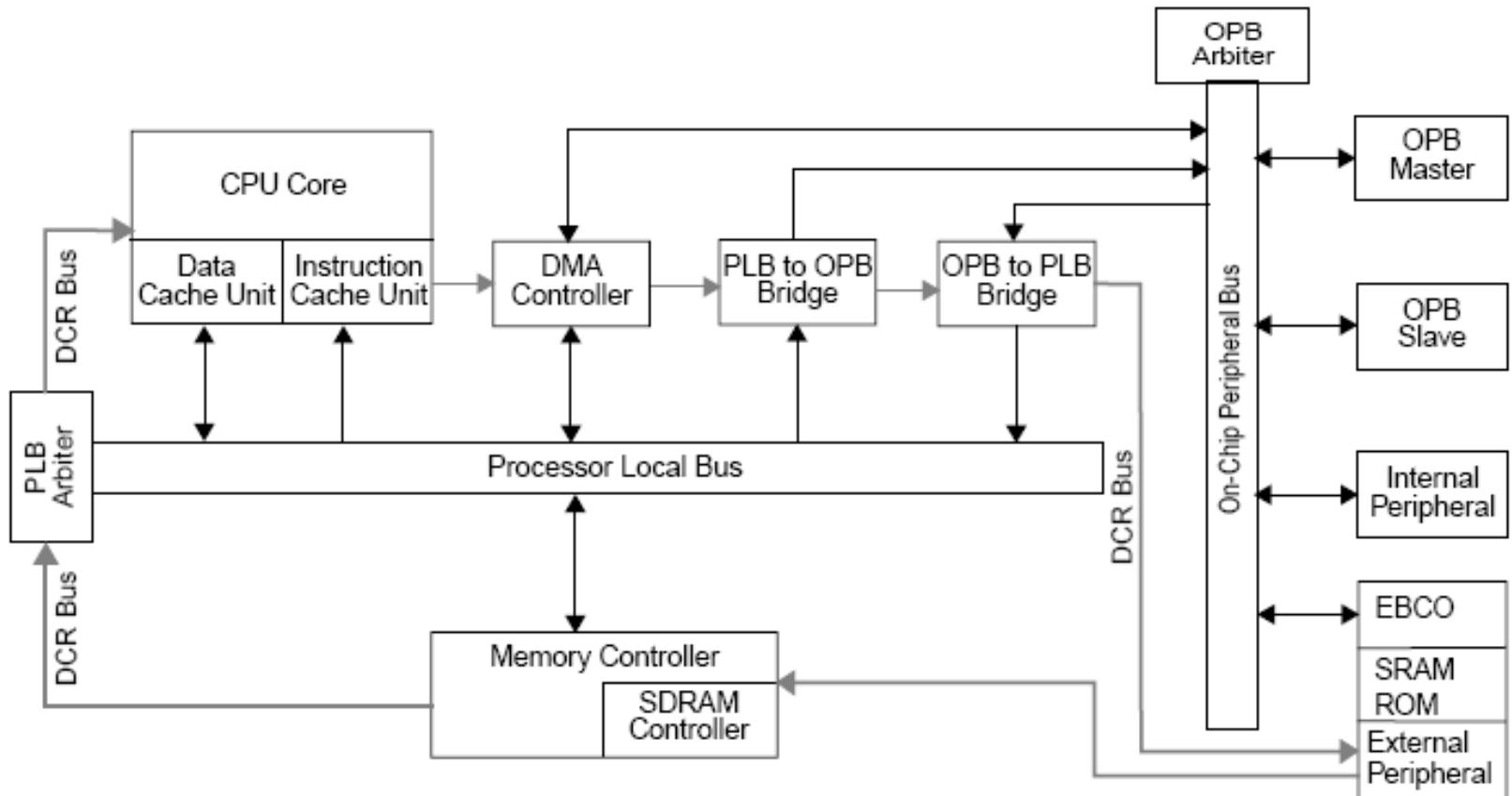
Для пересылки информации между ОП и быстродействующими устройствами необходимо обеспечивать независимую от ЦП передачу данных без использования ввода/вывода по прерыванию. Для этих целей используется специальное устройство – контроллер прямого доступа к памяти. ЦП занят только инициализацией ПДП.

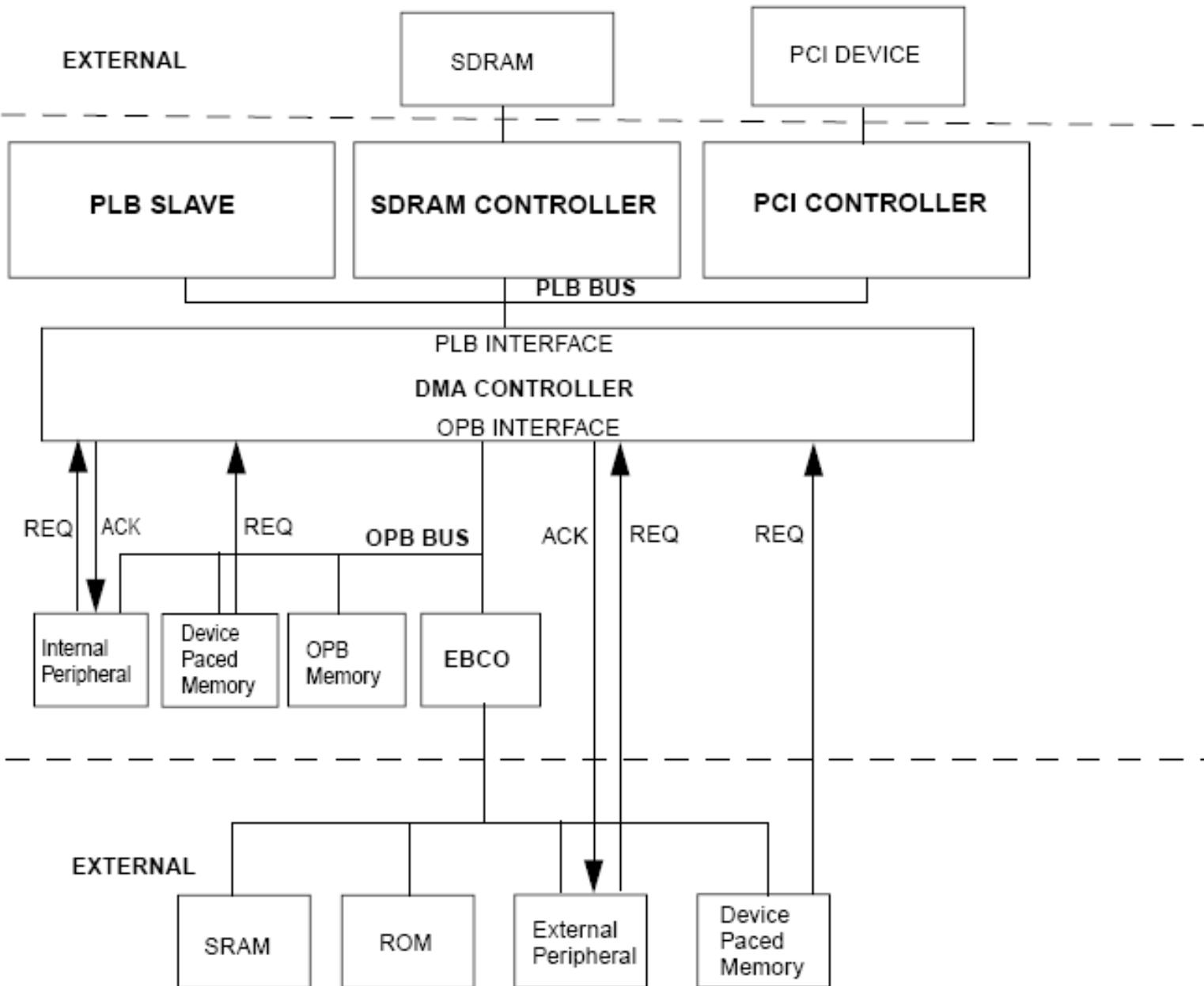
По инициативе ЦП или ПУ в КПДП передается:

- Вид запроса (чтение или запись);
- Адрес буфера в ОП;
- Количество слов для пересылки;
- Адрес ПУ.

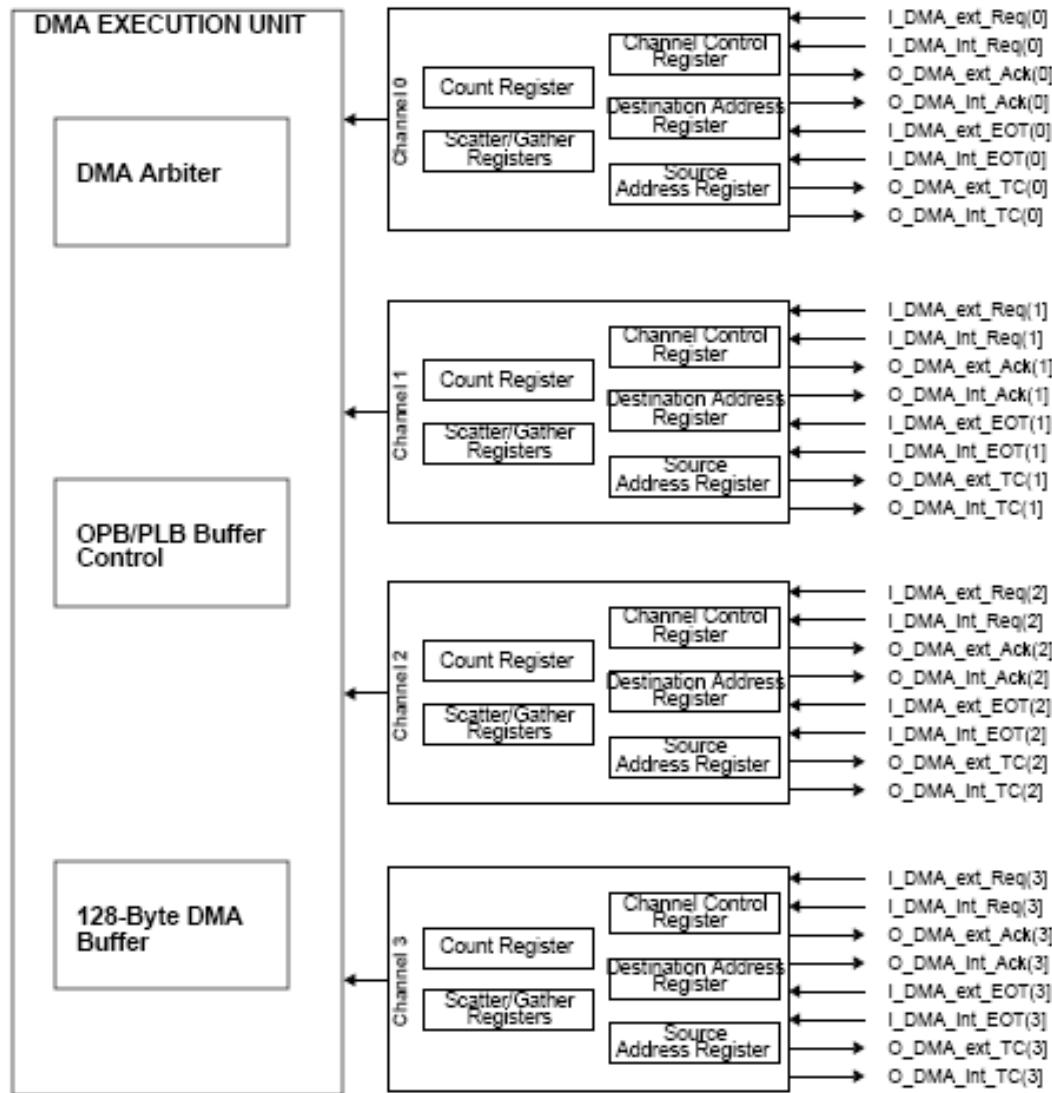


Пример построение системы с контроллером прямого доступа в память (IBM PowerPC405).





Контроллер прямого доступа в память (IBM PowerPC405 DMA).



XII. Организация шин

Совокупность устройств и сигнальных линий, обеспечивающих взаимосвязь всех частей ЭВМ образуют систему шин.

Состав системы:

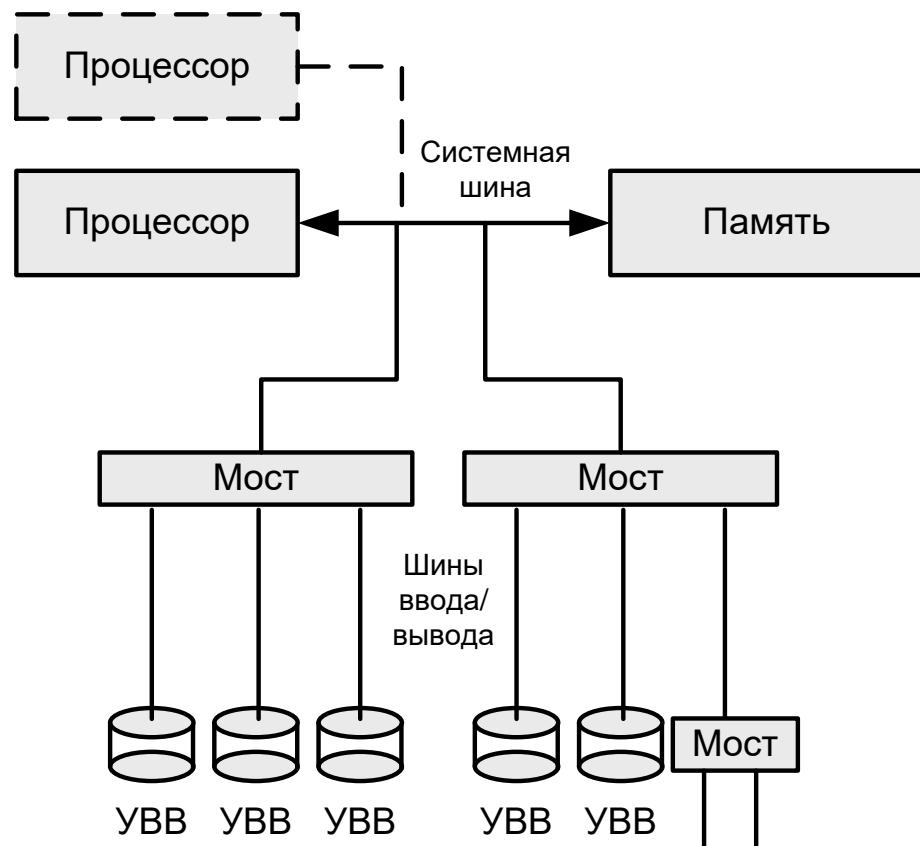
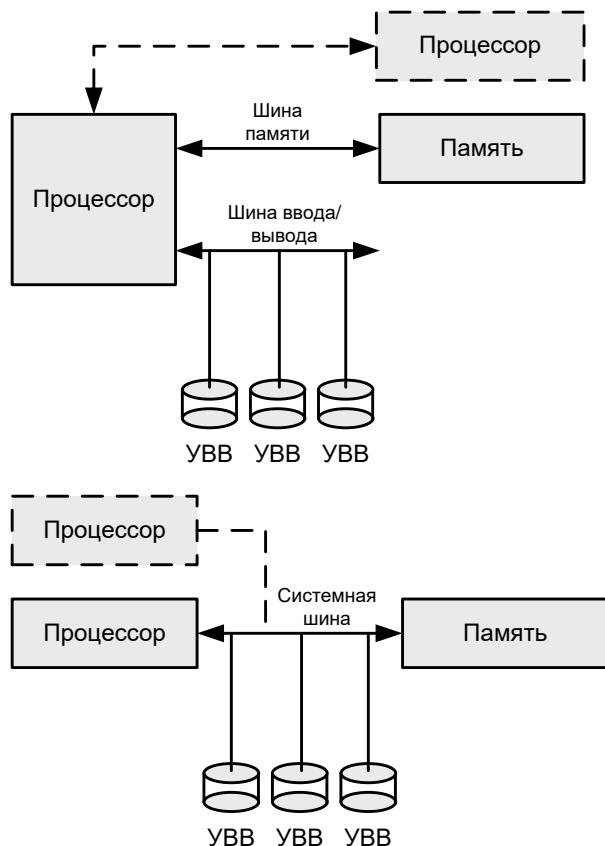
- Сигнальные линии
- Устройства арбитража
- Разъемы

Для описания (спецификации) конкретной шины необходимо описать:

- Совокупность линий (сигналов) и их назначение.
- Протокол: правила взаимодействия устройств с помощью шины (диаграммы, алгоритмы, автоматы).
- Физические, механические и электрические параметры шины и подключаемых устройств (частота, уровни сигналов, способ согласования волновых сопротивлений, длины линий, характеристики разъемов и т.д.).

Типы шин:

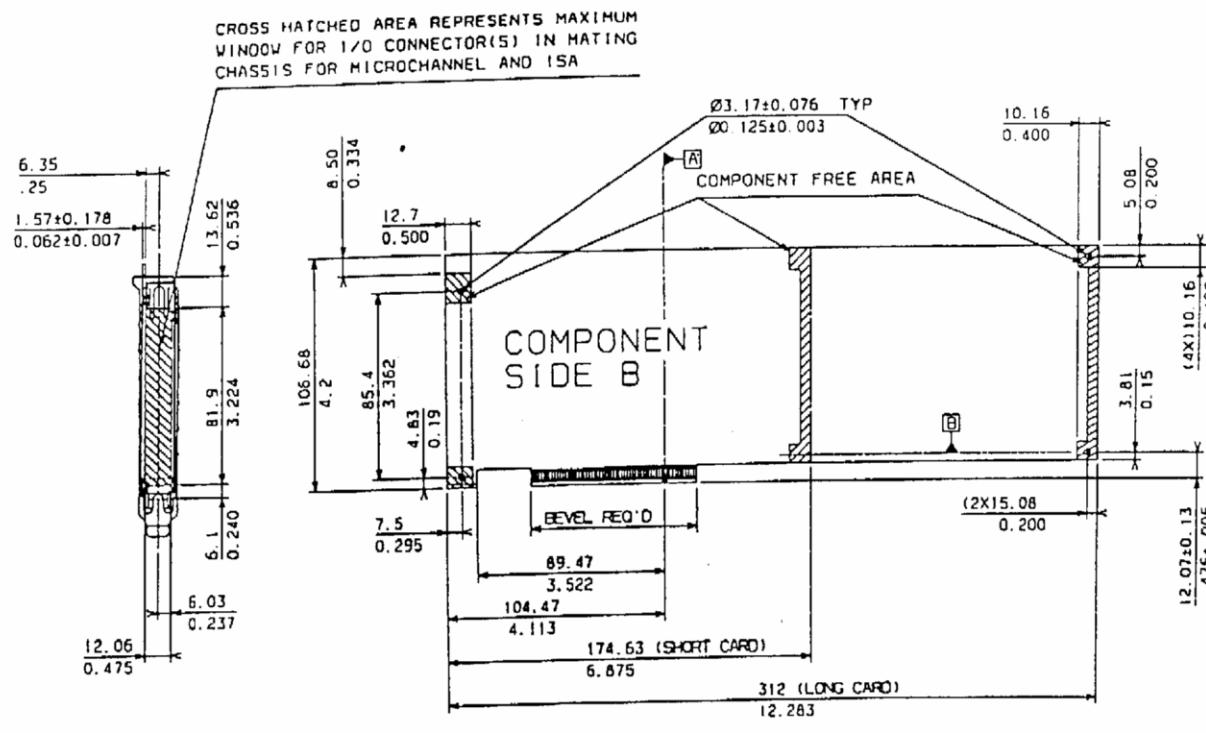
- Шины «процессор-память», «процессор-кэш», «процессор-процессор» (HyperTransport, P6 Back-Side Bus, ...)
- Шины ввода-вывода (PCI, SCSI, PCI-X, IDE, GX, OPB ...)
- Системные шины (EISA, PLB, Pentium 4 FSB, Multibus II, NuBus, ...)



Механические аспекты спецификации шин

Механические аспекты спецификации шин включают описание вариантов исполнения, чертежи разъемов, размеры и допуски печатных плат, описание направляющих и ключей, описание способов испытаний и т.д.

ПРИМЕР



Электрические аспекты спецификации шин

- Спецификация динамических и статических характеристик
- Спецификация синхронизации
- Спецификация инициализации
- Спецификация нагрузки
- Спецификация питания
- Назначения контактов разъема

Для каждого сигнала должны быть определены параметры:

- Уровни сигналов логического «0» и логической «1» (ТТЛ, ЭСЛ, КМОП и др.).
- Время переключения
- Моменты фиксации состояния относительно сигнала синхронизации
- Условия перевода в третье состояние
- Способ согласования.

При распространении сигналов по параллельным линиям необходимо учитывать:

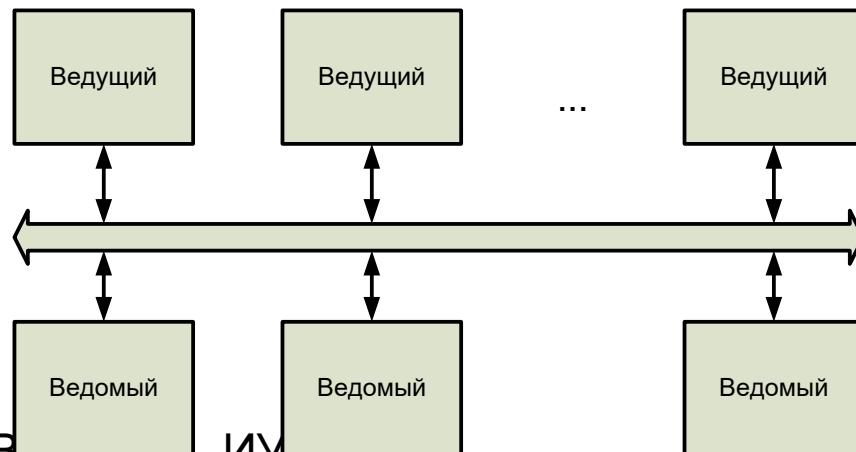
- Скорость распространения (обычно составляет ~70% скорости света $0.7 \cdot 300$ мм/нс).
- Отражение сигнала
- Перекос сигналов
- Архитектура ПОМи

Типы сигнальных линий:

- Линии адреса
- Линии данных (линии адреса и данных могут объединяться).
- Линии управления для передачи типа транзакции, статуса устройства, сигналов арбитража, запросов прерываний, резервных линий, линий константных значений.

Арбитраж шин

Ведущие устройства используют шину в разное время и должны отдавать и захватывать ее. Для определения очередности подключения ведущих устройств и учета их приоритетности используются: статические приоритеты, динамические приоритеты.

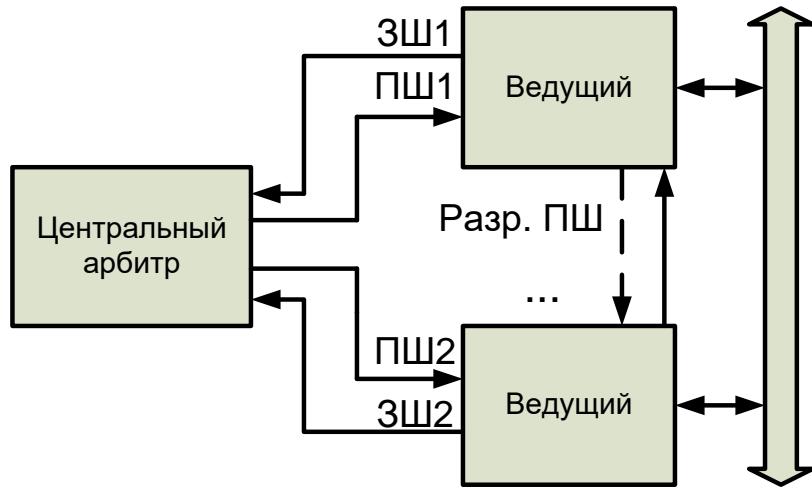


По способу арбитража: централизованный арбитраж, децентрализованный арбитраж.

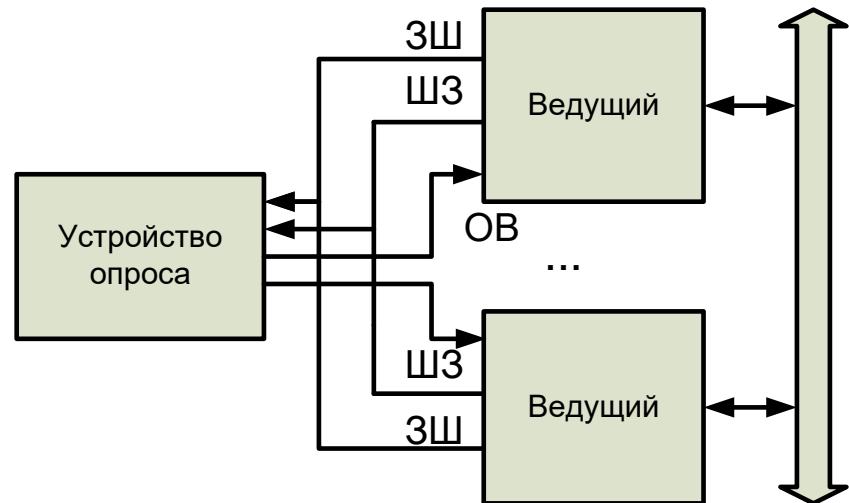
Схемы арбитража: циклическая схема, циклическая с учетом последнего, рандомизированный, схема с равными приоритетами, схема LRU

Централизованный арбитраж

Параллельный арбитраж

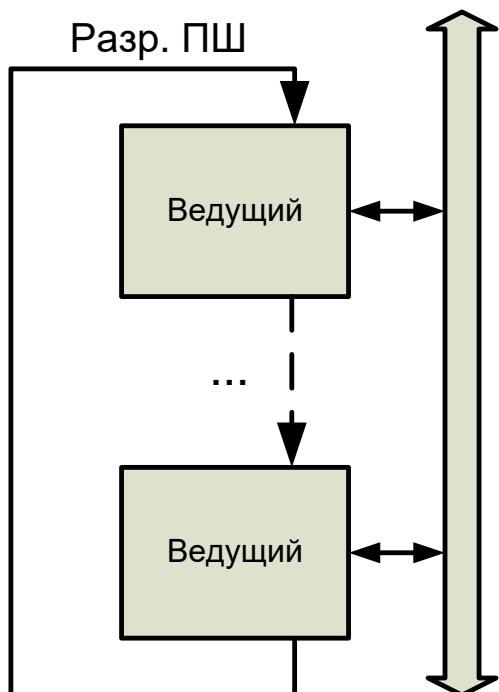


Централизованный опрос

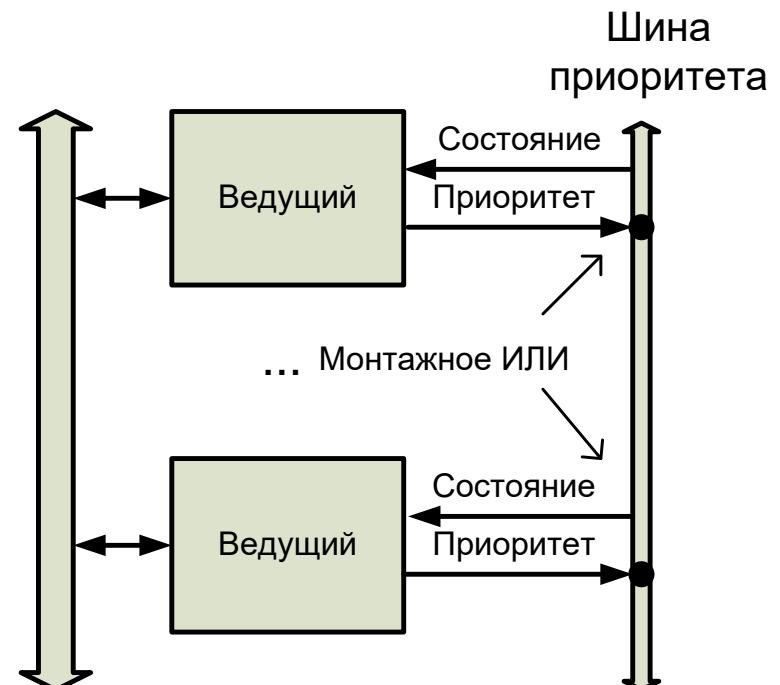


Децентрализованный арбитраж

Цепочечная схема без централизованного арбитража



Распределенный арбитраж



B1	1	0	→	B1	1	0
B2	0	1		В2	0	0
ШУ	1	1		ШУ	<1>	0>

Захват шины В1

Системная шина процессоров Р6

Частота: 66, 100, 133.

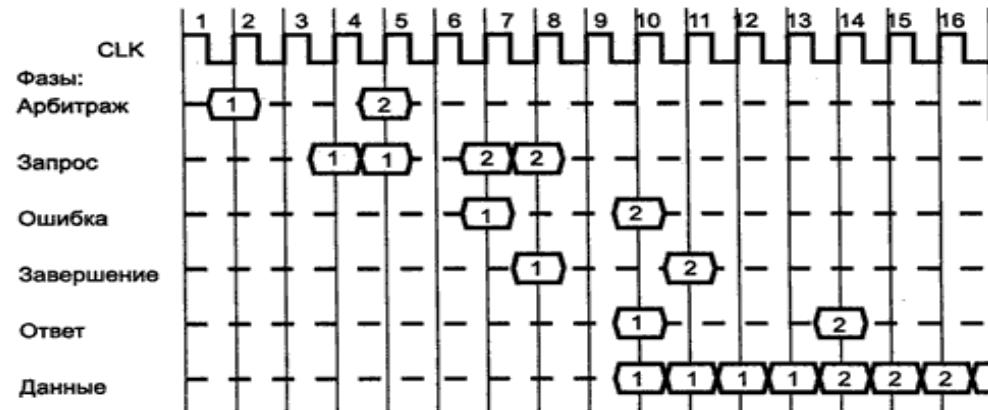
Количество абонентов: 16

Разрядность адреса: 32

Разрядность данных: 64

Контроль информации: РЕ и ECC

Поддержка операций с кэш



Пакетная и транзакционная передача

Каждое устройство-агент, подключенное к этойшине (например, любой из процессоров), до инициализации запроса должно получить через механизм арбитража право на использование шины запроса. Запрос выходит за два смежных такта: в первом такте передается адрес, тип обращения (чтение-запись памяти или ввода/вывода) и тому подобная информация. Во втором такте передается уникальный идентификатор транзакции, длина запроса, разрешенные байты шины и т. п. Через три такта после запроса проверяется состояние ошибки (error status) для защиты от ошибок передачи или нарушений протокола.

Любая обнаруженная ошибка вызывает повтор запроса, а вторая ошибка для транзакции вызывает исключение контроля (machine check exception).

Шина PCI

Частота: 33,66.

Количество абонентов: 21

Разрядность адреса/данных: 32,64 (132...528 МБ/сек)

Контроль информации: по четности

Поддержка операций с кэш

Пакетная передача

Поддержка иерархии шин (до 256)

Plug and Play технология.

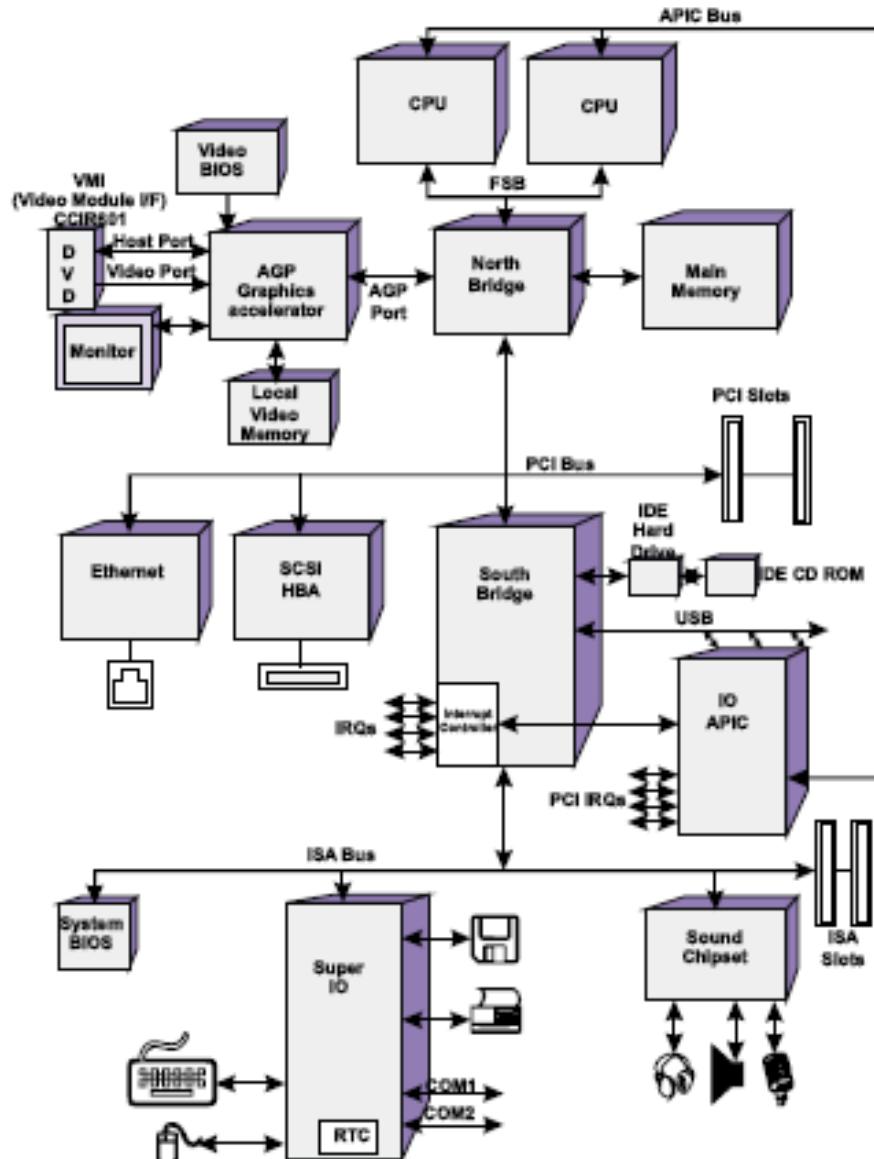
Поддержка многих управителей шины (Masters) и ведомых (Targets)/

Спецификации: 1.0 (1992), 2.0(1993), 2.1(1995), 2.2(1999)

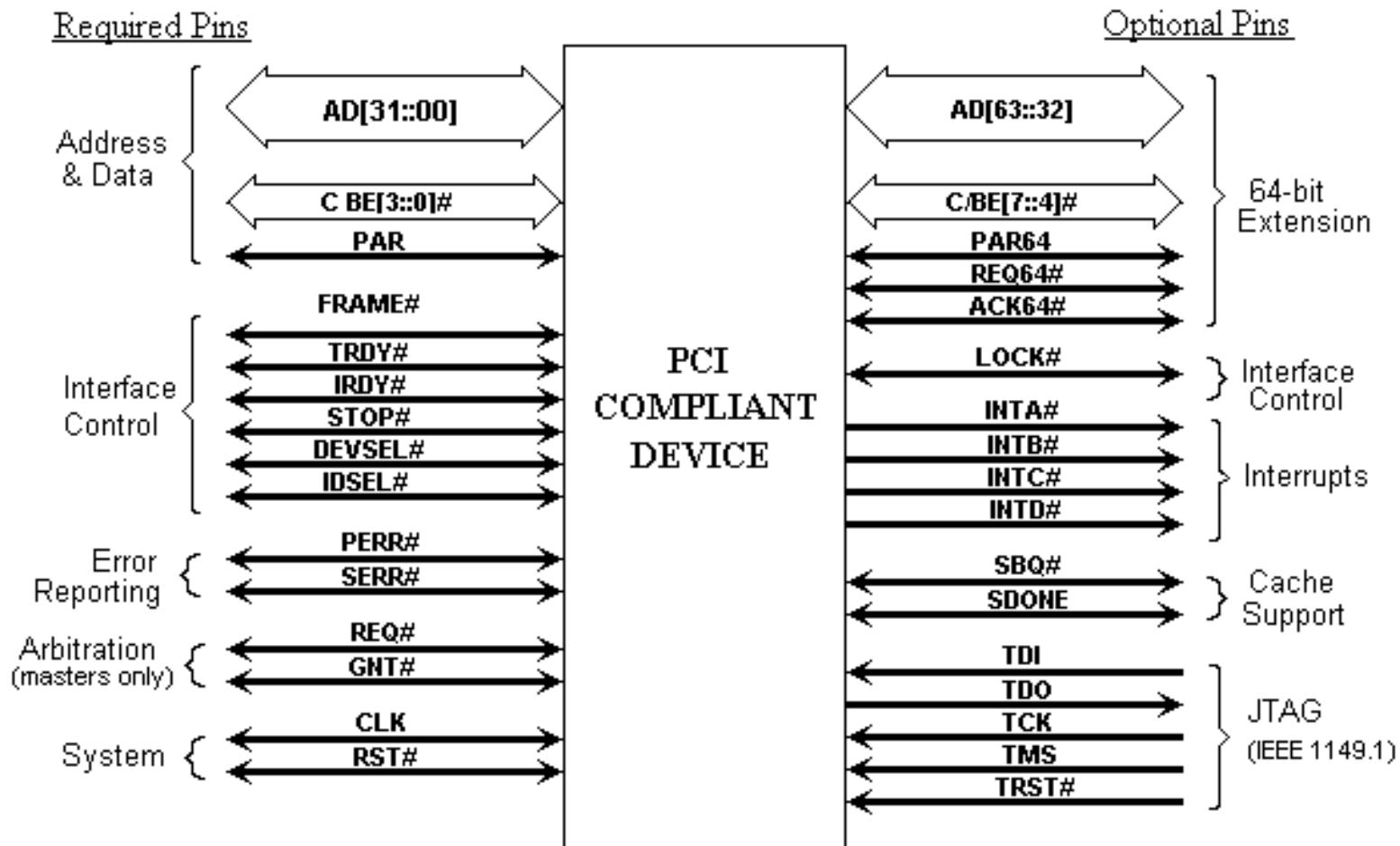
Локальная шина PCI - это высокопроизводительная 32-битная или 64-битная шина с мультиплексированными линиями адреса и данных. Она предназначена для использования в качестве связующего механизма между высокоинтегрированными периферийными контроллерами ввода-вывода, периферийными встраиваемыми платами и системами процессор/память.

Спецификация локальной шины PCI, реализация 2.0, включает протокол, электрическую, механическую и конфигурационную спецификации для локальной шины PCI и плат расширения. Описания электрических сигналов приводятся для напряжений питания 3.3В и 5.0В.

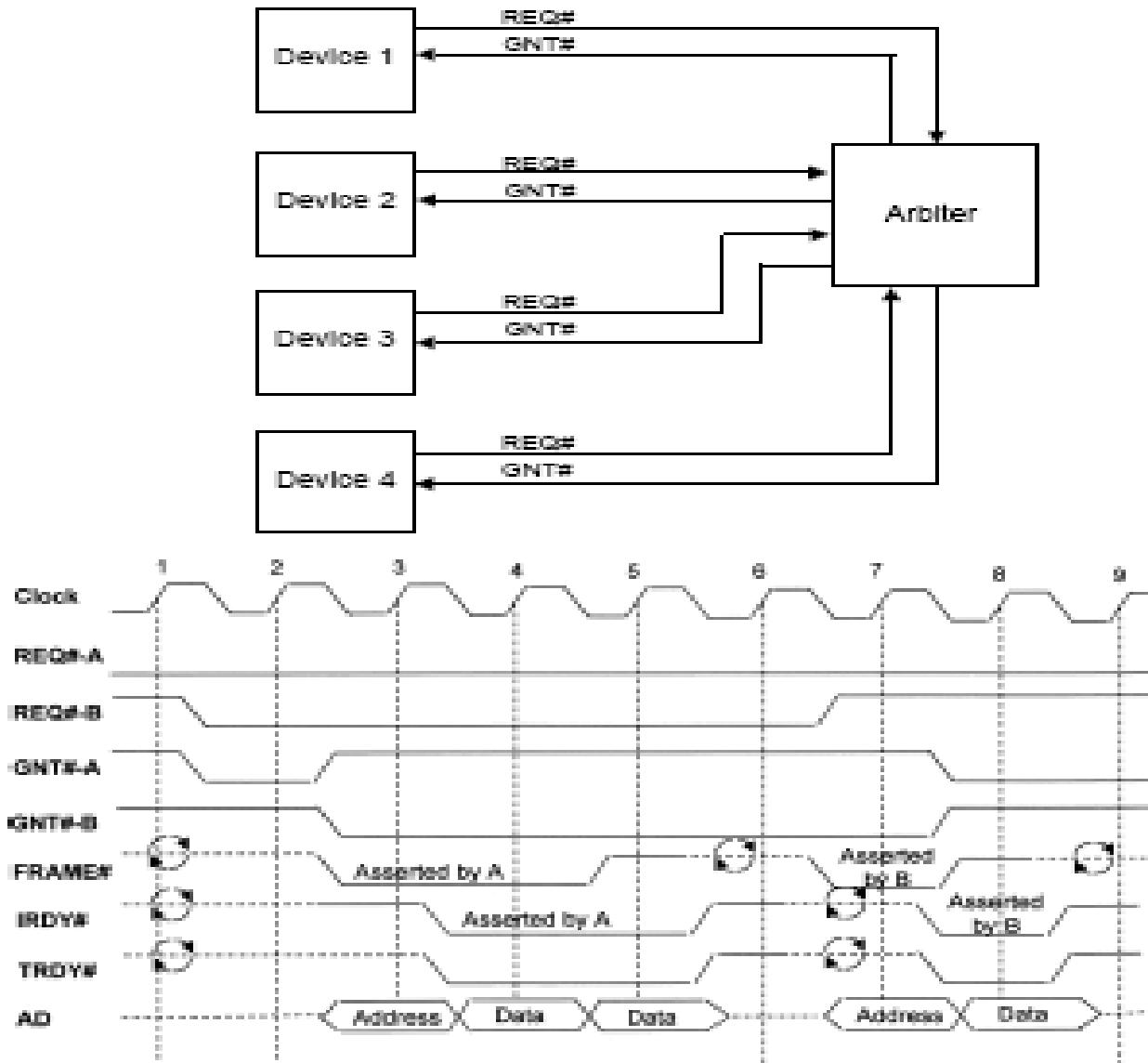
Пример построения систем на основе шины PCI*



Сигналы шины PCI



Арбитраж PCI шины



Операции на шине PCI

C/BE[3::0]#	Тип операции
0000	Interrupt Acknowledge (подтверждение прерывания)
0001	Special Cycle (специальный цикл)
0010	I/O Read (чтение при вводе - выводе)
0011	I/O Write (запись при вводе - выводе)
0100	Зарезервировано
0101	Зарезервировано
0110	Memory Read (чтение памяти)
0111	Memory Write (запись в память)
1000	Зарезервировано
1001	Зарезервировано
1010	Configuration Read (чтение конфигурации)
1011	Configuration Write (запись конфигурации)
1100	Memory Read Multiple (множественное чтение памяти)
1101	Dual Address Cycle (двойной цикл адреса)
1110	Memory read Line (линия чтения памяти)
1111	Memory Write and Invalidate (запись в память и недействительные данные)

FRAME# - Управляется мастером для того, чтобы он мог указать начало и конец транзакции.

IRDY# - Управляется мастером, чтобы он мог инициировать циклы ожидания.

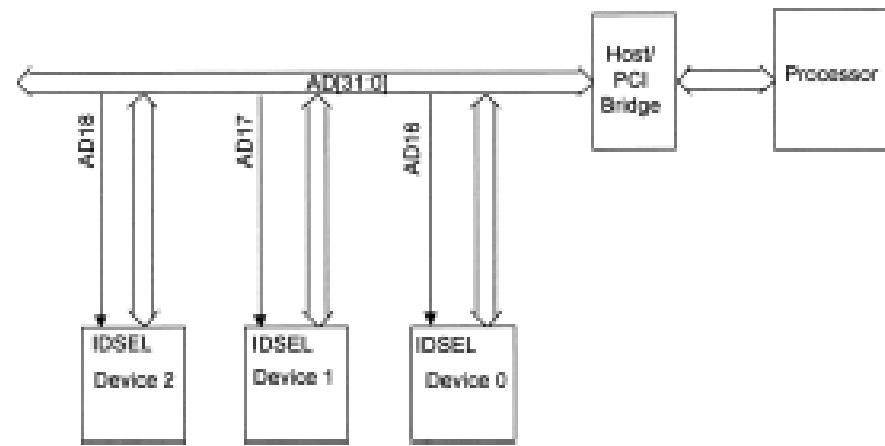
TRDY# - Управляется целевым устройством, чтобы оно могло инициировать циклы ожидания.

Адресация на PCI шине

Все определено три физических адресных пространства. Пространства адресов памяти и ввода-вывода объединены. Адресное пространство конфигураций было введено для обеспечения аппаратной конфигурации PCI.

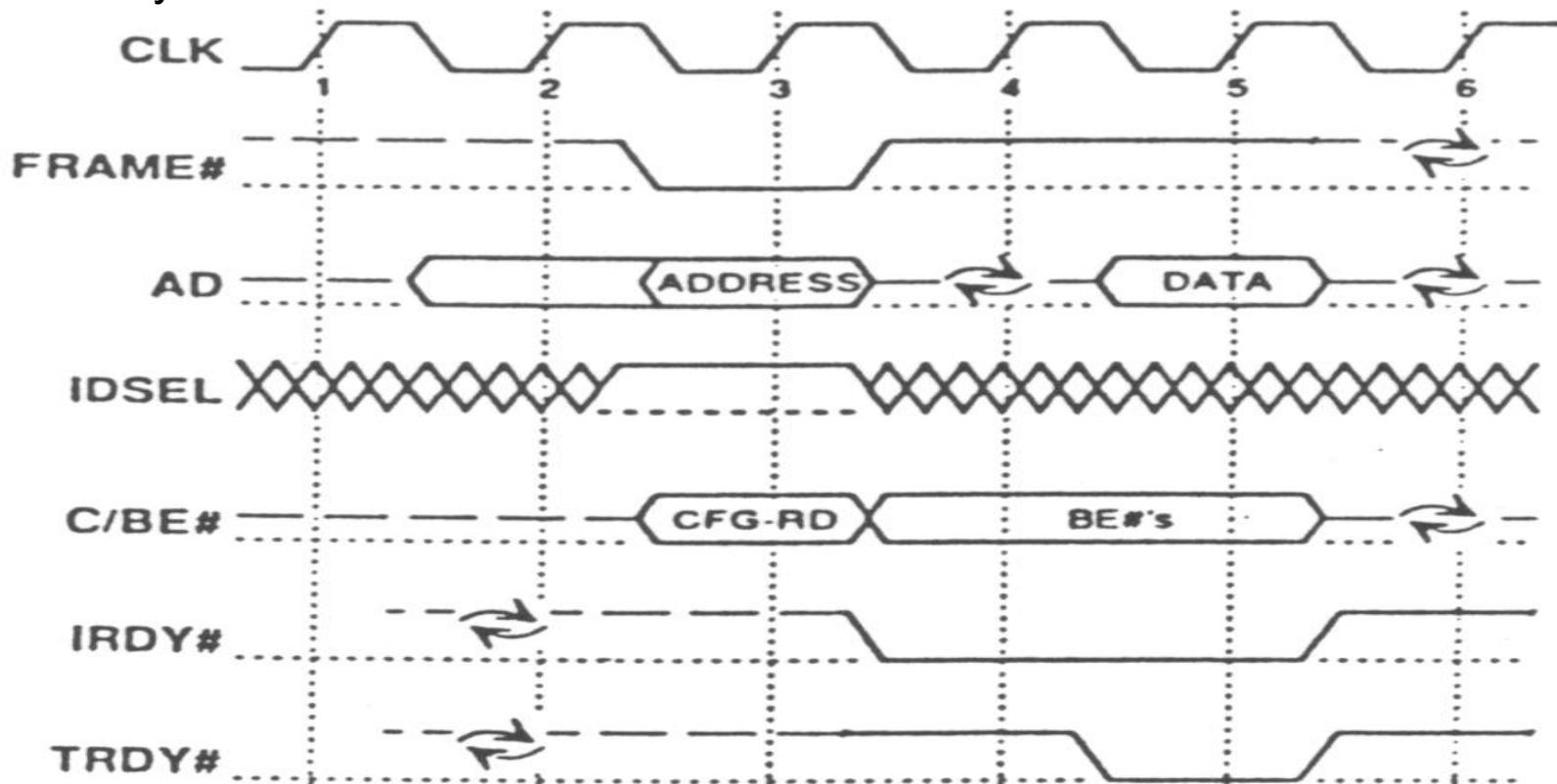
Дешифрирование адреса на шине PCI распределено; это означает, что оно выполняется на каждом устройстве. Это устраняет проблемы для центральной дешифрирующей логики, а также для сигналов выбора устройства, независимо от их использования для конфигурации. Каждый агент отвечает только на свой дешифрированный адрес.

PCI обеспечивает полную программно управляемую инициализацию и конфигурацию через отдельное адресное пространство конфигурации. PCI устройства должны обеспечить 256 байтов регистров конфигурации для этой цели.



Чтение конфигурации

Для поддержки иерархии PCI шины используются два типа доступа конфигурации. Тип 1 и тип 0 доступа конфигурации различаются значениями на AD[1::0]. Тип 0 цикла конфигурации (когда AD[1::0] = "00") используется, чтобы выбрать устройство на PCIшине, где цикл выполняется. Тип 1 цикла конфигурации (когда AD[1::0] = "01") используется, чтобы передать запрос конфигурации на другую PCIшину.

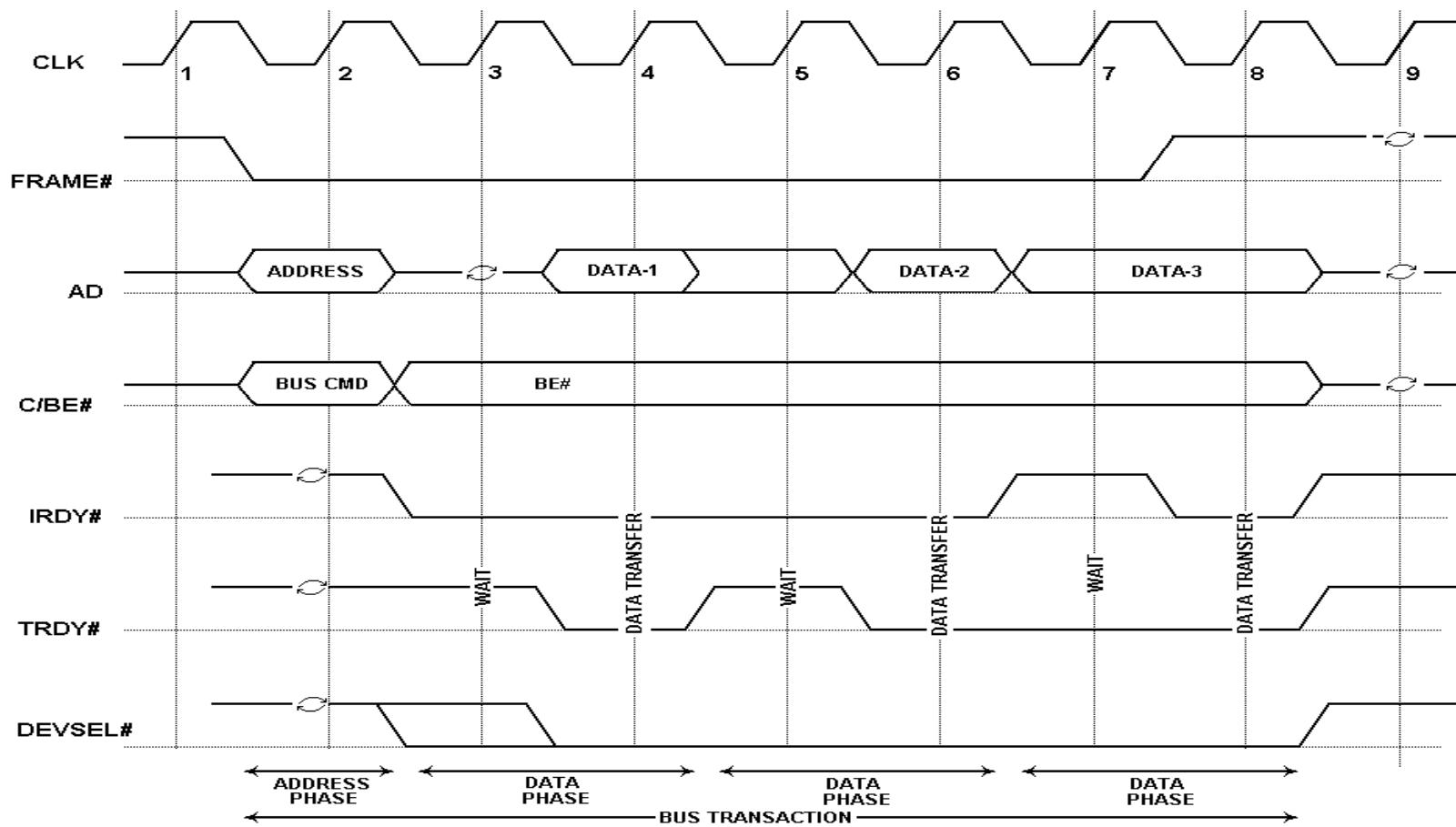


Пространство конфигурации Тип 0

31	16 15	0
Device ID	Vendor ID	00h
Status	Command	04h
Class Code	Revision ID	08h
BIST	Header Type	Cache Line Size
		0ch
		10h
		14h
		18h
	Base Address Registers	1Ch
		20h
		24h
	Cardbus CIS Pointer	28h
Subsystem ID	Subsystem Vendor ID	2ch
Expansion Bus ROM Base		30h
Reserved	Cap Pntr	34h
Reserved		38h
Max_Lat	Min_Gnt	Interrupt Pin
		Interrupt Line

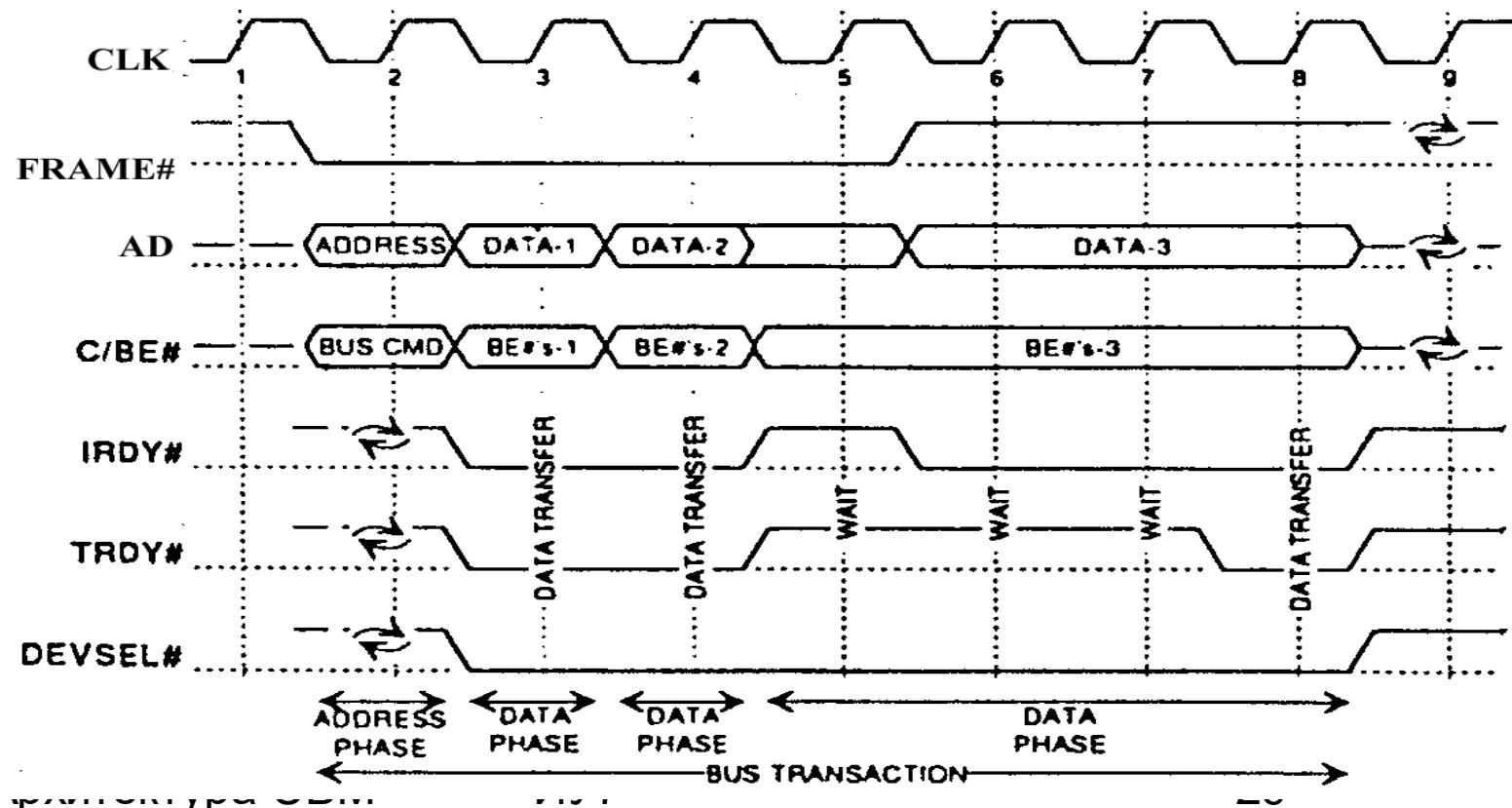
Базовая операция чтения

Транзакция чтения начинается фазой адреса, которая происходит во 2-ом такте, когда впервые устанавливается сигнал FRAME#. В течение фазы адреса AD[31::00] содержит допустимый адрес, а C/BE[3::0]# - допустимую команду шины.



Базовая операция записи

Транзакция записи начинается в такте 2, когда впервые устанавливается в активное состояние сигнал FRAME#. Транзакция записи подобна транзакции чтения, за исключением того, что после фазы адреса не требуется обратный цикл, так мастер обеспечивает и адрес, и данные. Фазы данных для транзакции записи такие, как и для транзакции чтения.

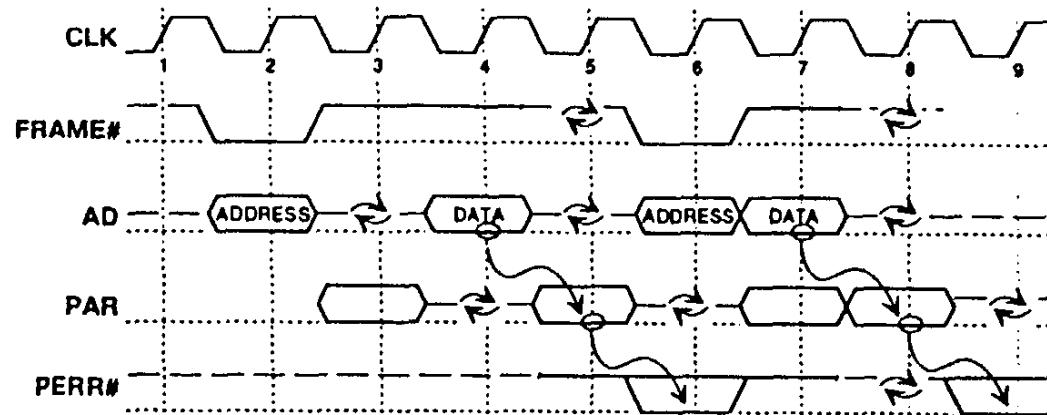


Контроль ошибок

Выводы для сообщения об ошибках требуются всем устройствам:

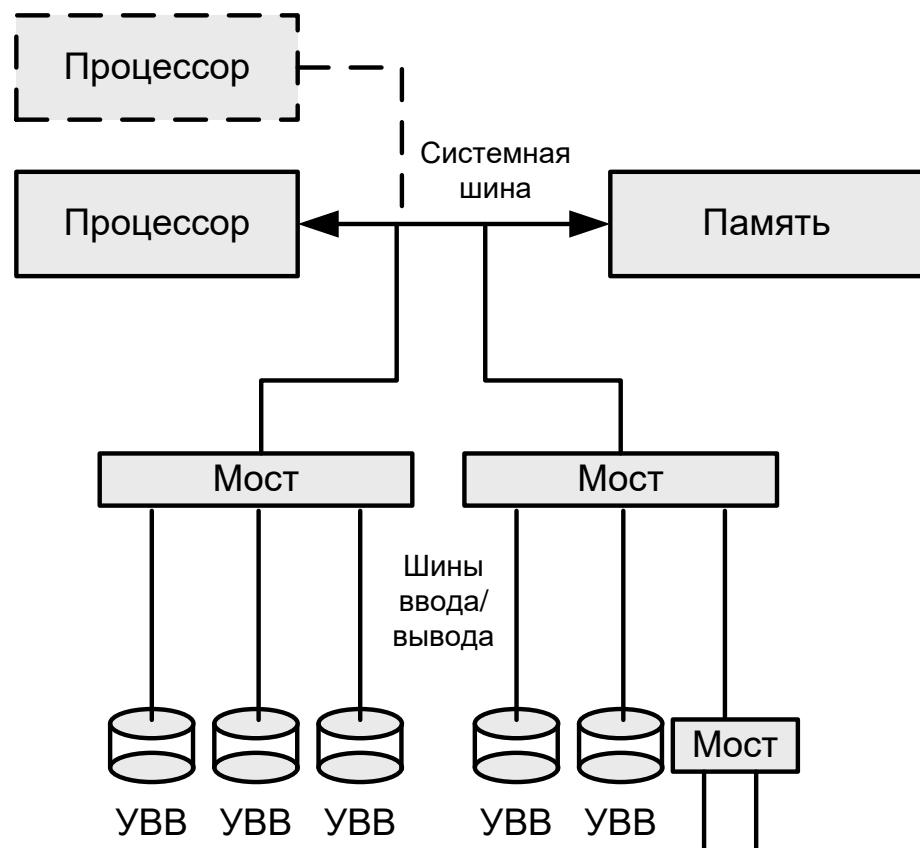
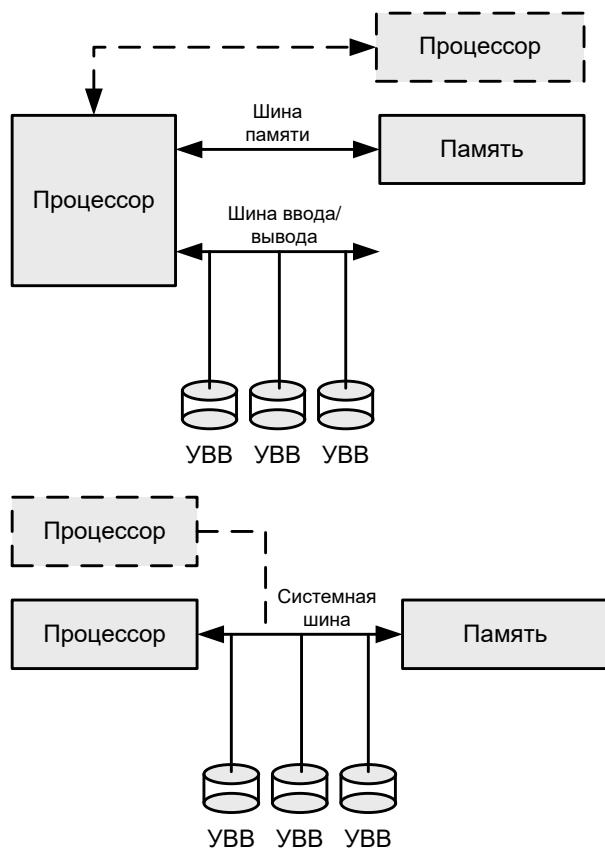
PERR# s/t/s

Вывод Parity Error (ошибка контроля по четности) предназначен только для сообщения об ошибках контроля по четности во время всех транзакций PCI, за исключением специального цикла (Special Cycle). Вывод PERR# - три-стабильный и должен активно управляться агентом, получающим данные в течение двух тактов, после того, как обнаружена ошибка контроля данных по четности. Минимальная продолжительность PERR# - один такт для любой фазы данных, у которой обнаружена ошибка контроля данных по четности (если идут последовательно несколько фаз данных, каждая из которых имеет ошибку контроля данных по четности, то сигнал PERR# будет установлен за более, чем один такт). PERR# должен быть установлен в высокое состояние за один такт прежде, перед тем, как он перейдет в третье состояние со всеми соответствующими тристабильными сигналами. Не существует никаких специальных условий для случая, когда теряется ошибка контроля данных по четности или сообщается об отсроченной ошибке. Агент не может установить PERR#, пока он не разрешил доступ, установив DEVSEL# и завершив фазу данных.



Типы шин:

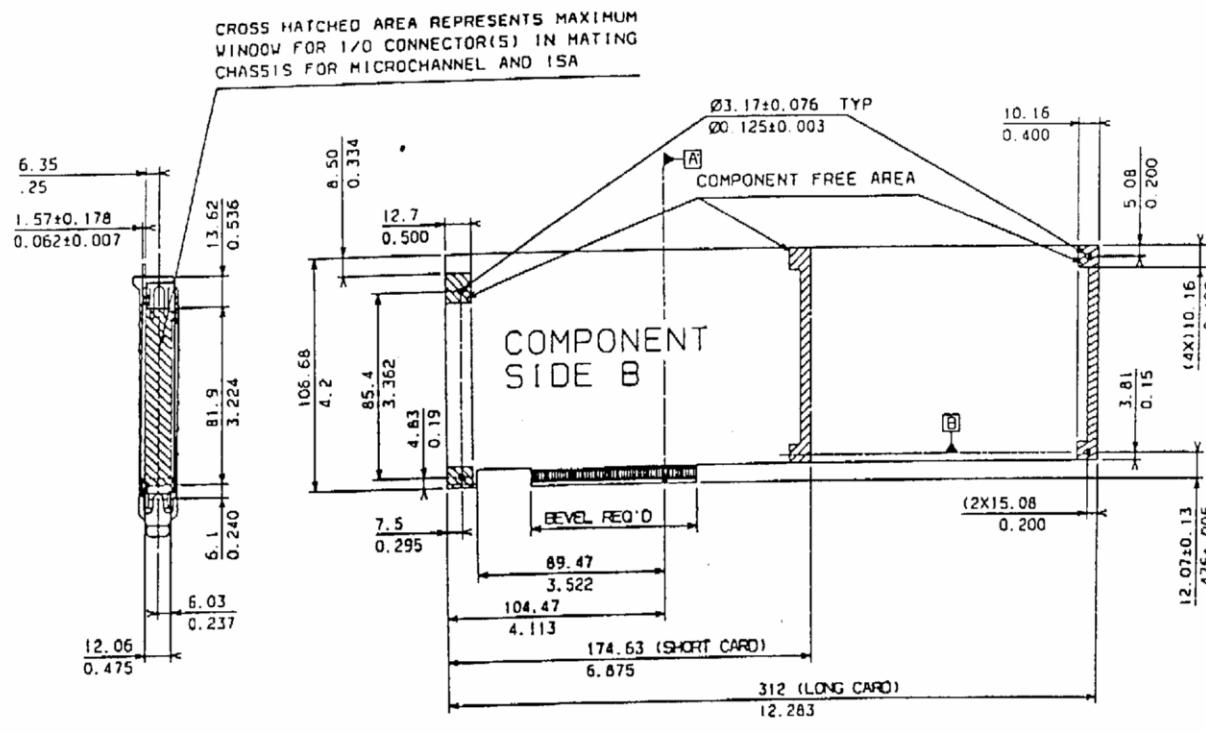
- Шины «процессор-память», «процессор-кэш», «процессор-процессор» (HyperTransport, P6 Back-Side Bus, ...)
- Шины ввода-вывода (PCI, SCSI, PCI-X, IDE, GX, ...)
- Системные шины (EISA, Pentium 4 FSB, Multibus II, NuBus, ...)



Механические аспекты спецификации шин

Механические аспекты спецификации шин включают описание вариантов исполнения, чертежи разъемов, размеры и допуски печатных плат, описание направляющих и ключей, описание способов испытаний и т.д.

ПРИМЕР



Электрические аспекты спецификации шин

- Спецификация динамических и статических характеристик
- Спецификация синхронизации
- Спецификация инициализации
- Спецификация нагрузки
- Спецификация питания
- Назначения контактов разъема

Для каждого сигнала должны быть определены параметры:

- Уровни сигналов логического «0» и логической «1» (ТТЛ, ЭСЛ, КМОП и др.).
- Время переключения
- Моменты фиксации состояния относительно сигнала синхронизации
- Условия перевода в третье состояние
- Способ согласования.

При распространении сигналов по параллельным линиям необходимо учитывать:

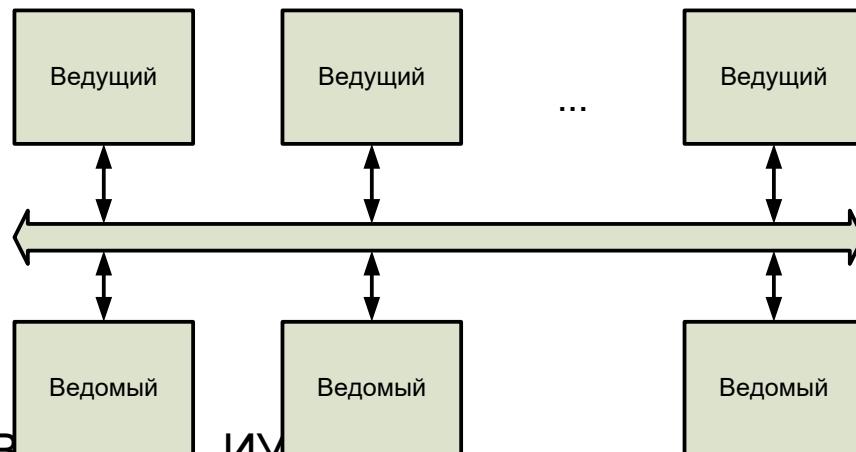
- Скорость распространения (обычно составляет ~70% скорости света $0.7 \cdot 300$ мм/нс).
- Отражение сигнала
- Перекос сигналов
- Архитектура ПОМи

Типы сигнальных линий:

- Линии адреса
- Линии данных (линии адреса и данных могут объединяться).
- Линии управления для передачи типа транзакции, статуса устройства, сигналов арбитража, запросов прерываний, резервных линий, линий константных значений.

Арбитраж шин

Ведущие устройства используют шину в разное время и должны отдавать и захватывать ее. Для определения очередности подключения ведущих устройств и учета их приоритетности используются: статические приоритеты, динамические приоритеты.

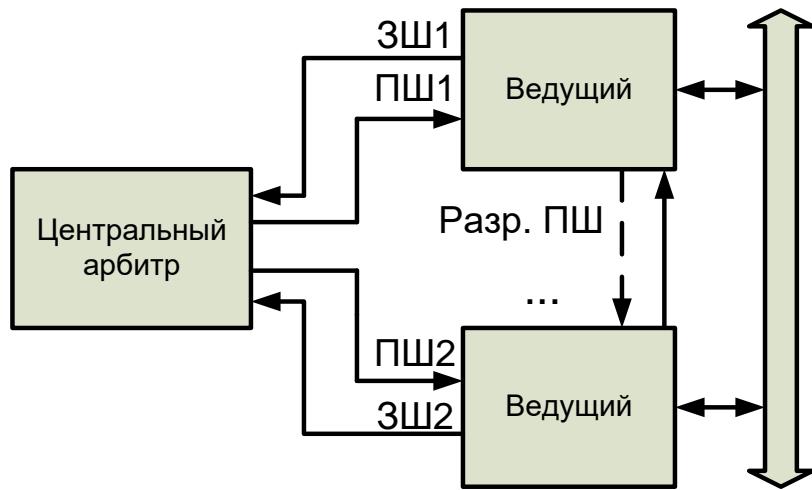


По способу арбитража: централизованный арбитраж, децентрализованный арбитраж.

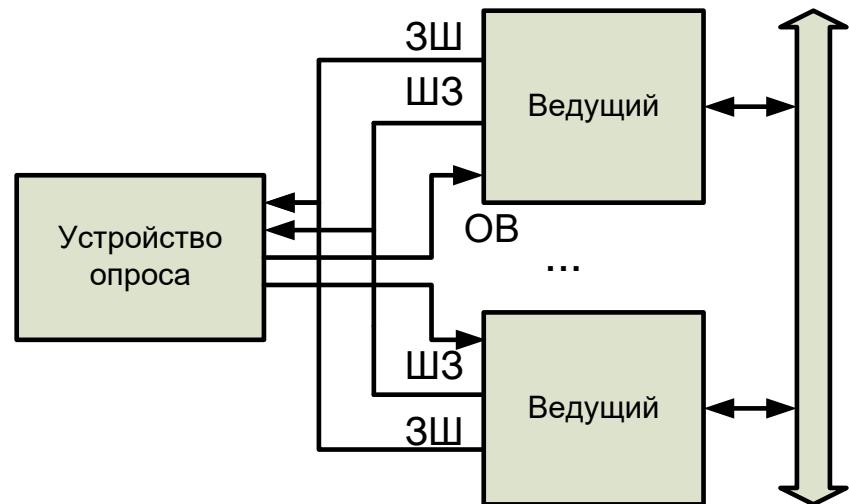
Схемы арбитража: циклическая схема, циклическая с учетом последнего, рандомизированный, схема с равными приоритетами, схема LRU

Централизованный арбитраж

Параллельный арбитраж

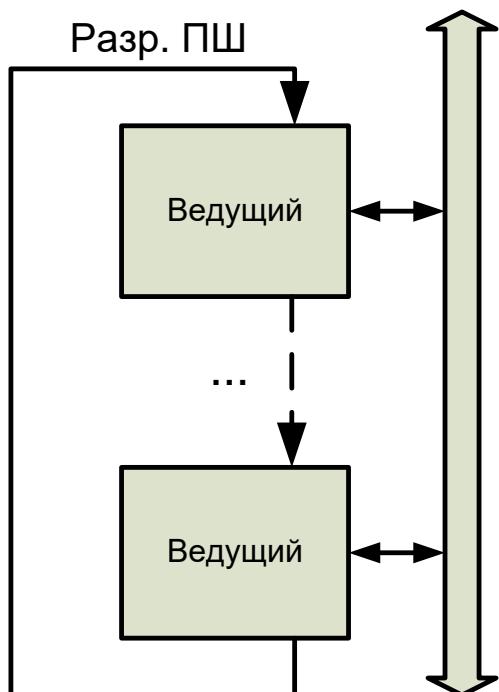


Централизованный опрос

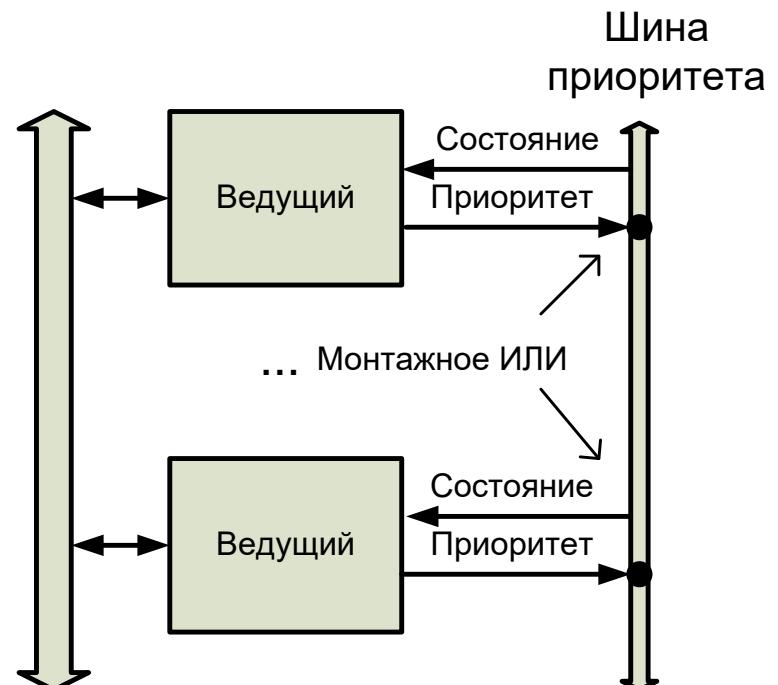


Децентрализованный арбитраж

Цепочечная схема без централизованного арбитража



Распределенный арбитраж



B1	1	0	→	B1	1	0
B2	0	1		B2	0	0
ШУ	1	1		ШУ	<1>	0>

Захват шины В1

Системная шина процессоров Р6

Частота: 66, 100, 133.

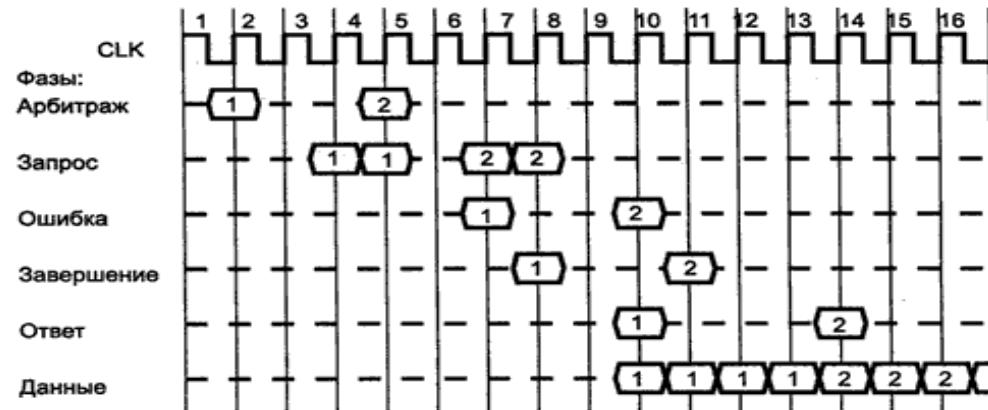
Количество абонентов: 16

Разрядность адреса: 32

Разрядность данных: 64

Контроль информации: РЕ и ECC

Поддержка операций с кэш



Пакетная и транзакционная передача

Каждое устройство-агент, подключенное к этойшине (например, любой из процессоров), до инициализации запроса должно получить через механизм арбитража право на использование шины запроса. Запрос выходит за два смежных такта: в первом такте передается адрес, тип обращения (чтение-запись памяти или ввода/вывода) и тому подобная информация. Во втором такте передается уникальный идентификатор транзакции, длина запроса, разрешенные байты шины и т. п. Через три такта после запроса проверяется состояние ошибки (error status) для защиты от ошибок передачи или нарушений протокола.

Любая обнаруженная ошибка вызывает повтор запроса, а вторая ошибка для тодолжен запроса. Это вызывает исключение контроля (machine check exception).

XIII. Вычислительные системы

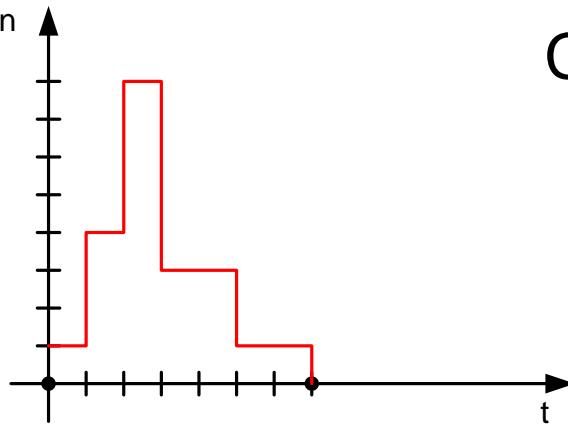
Вычислительная машина, позволяющая выполнять параллельную обработку частей программы или различных программ на нескольких процессорных элементах называется вычислительной системой.

Уровни параллелизма:

- Уровень заданий. ВС решает различные несвязанные задания на разных процессах (многозадачный режим).
- Уровень программ. ВС позволяет обрабатывать части одной программы на различных процессорах.
- Уровень команд. Фазы команды выполняются параллельно при конвейерной обработке.
- Уровень битов. Биты операндов могут обрабатываться параллельно.

По связности параллельных частей программ различают:

- Крупнозернистый параллелизм.
- Среднезернистый параллелизм.
- Мелкозернистый параллелизм.



Средний параллелизм

$$A = \frac{\sum_{i=1}^m i \cdot t_i}{\sum_{i=1}^m t_i} = \frac{21}{7} = 3$$

Ускорение

$$S(n) = \frac{T(1)}{T(n)}$$

$T(n)$ – количество квантов времени для решения задачи на n процессорах
!!! $S(n)$ в некоторых случаях превосходит n .

Ускорение неизбежно уменьшается из-за:

- программных издержек на распределение заданий по процессорам;
- дисбаланса загрузки процессоров;
- издержек на коммуникации.

Эффективность

$$E(n) = \frac{S(n)}{n}$$

Эффективность показывает ускорение в расчете на один процессор.
 $1/n \leq E(n) \leq n$

Закон Амдала

Каждая программа состоит из двух типов команд: исполняющихся исключительно последовательно (доля таких команд - f) допускающих параллельное исполнение (доля таких команд $(1-f)$)



Время последовательного исполнения: $T_{\text{посл}}$

Время при параллельной обработке: $T_{\text{пар}} = f * T_{\text{посл}} + (1-f)*T_{\text{посл}}/n$

Ускорение: $S = T_{\text{посл}}/T_{\text{пар}} = n/(1+(n-1)*f) \rightarrow 1/f (n \rightarrow \infty)$

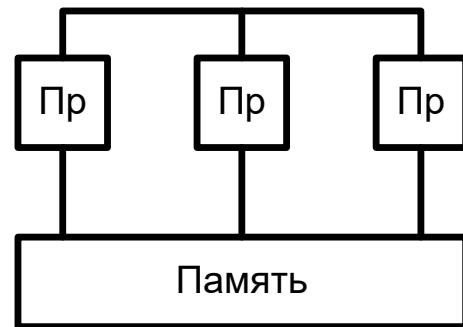
Ускорение обратно пропорционально проценту последовательных команд.

Организация памяти вычислительных систем

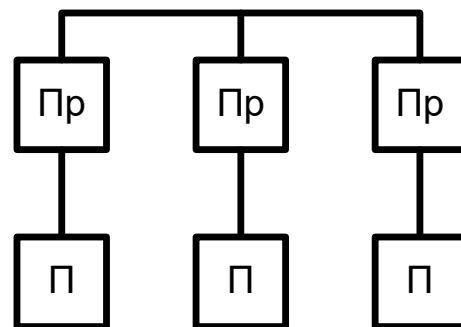
Основа любой вычислительной системы – принципы взаимодействия памяти и процессоров.

Для многопроцессорных ВС память можно сделать общей для всех процессоров (shared memory) и раздельной (distributed memory).

SM



DM



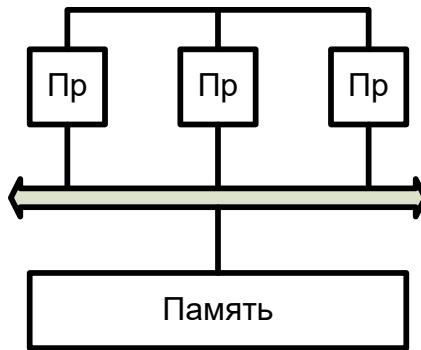
- (+) Упрощается пересылка данных, ускоряется обработка.
- (-) Усложняется память
- (-) Затрудняется использование глобальных системных таблиц (векторов прерываний и т.д.)

- (+) Упрощается программирование, упрощается память
- (-) Усложняется межпроцессорное взаимодействие.

Архитектура с совместной памятью

Системы с однородным доступом к памяти

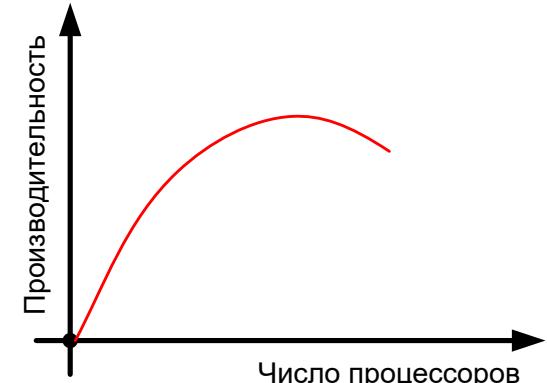
Для массива одинаковых процессоров удобно организовать единственный доступ к памяти.



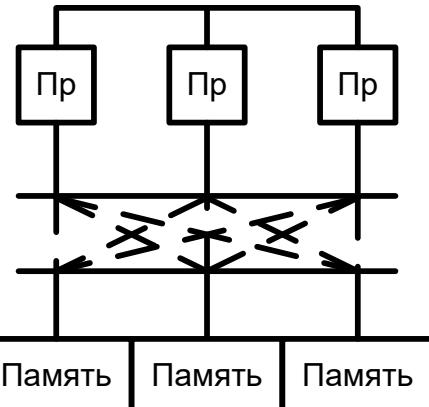
UMA (Unified Memory Access).

(-) В каждый момент времени к памяти может обращаться только один процессор.

Пример: Intel Core Duo, Xeon (SMP), IBM Power5+.



Системы с неоднородным доступом к памяти



NUMA (Non Unified Memory Access).

Каждый процессор обладает собственной памятью (кэш).

(-) В каждый момент времени к одному блоку памяти может обращаться только один процессор.

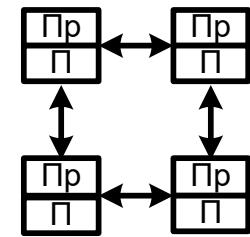
Другие SM архитектуры: СОМА, CCNUMA (AMD Opteron)

Архитектура с распределенной памятью

В вычислительной системе с распределенной памятью каждый процессор имеет свою собственную адресную память

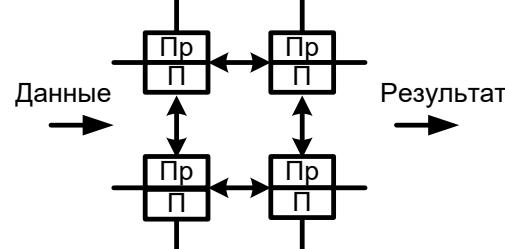
Процессор не обращается к удаленной памяти непосредственно, а посыпает сообщение тому процессору, к которому она относится.

- (+) Локальная шина процессоров меньше используется.
- (+) Количество процессоров в системе можно увеличить.
- (+) Данные в локальной памяти легко согласовывать.
- (-) Дополнительные издержки на обмен данными
(передача запроса, подготовка данных, передача данных).

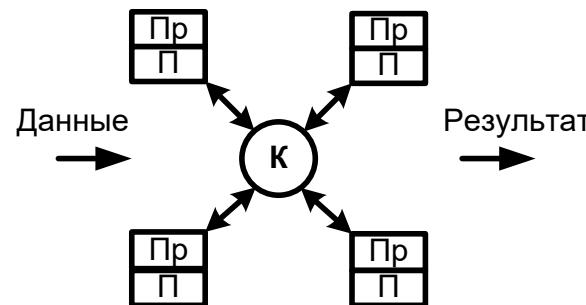


Некоторые типы архитектур ВС

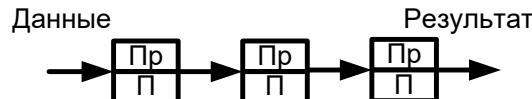
Матричные ВС



ВС с программируемой структурой



Конвейерные ВС



Архитектура ЭВМ

Кластерные ВС

