Учебный курс

Принципы построения и функционирования ЭВМ

Лекция 1

История развития вычислительной техники и архитектура Фон-Неймана

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Что делает вычислительная машина

- Компьютер преобразует коды, и эти преобразования должны соответствовать принятым правилам выполнения арифметико-логических действий.

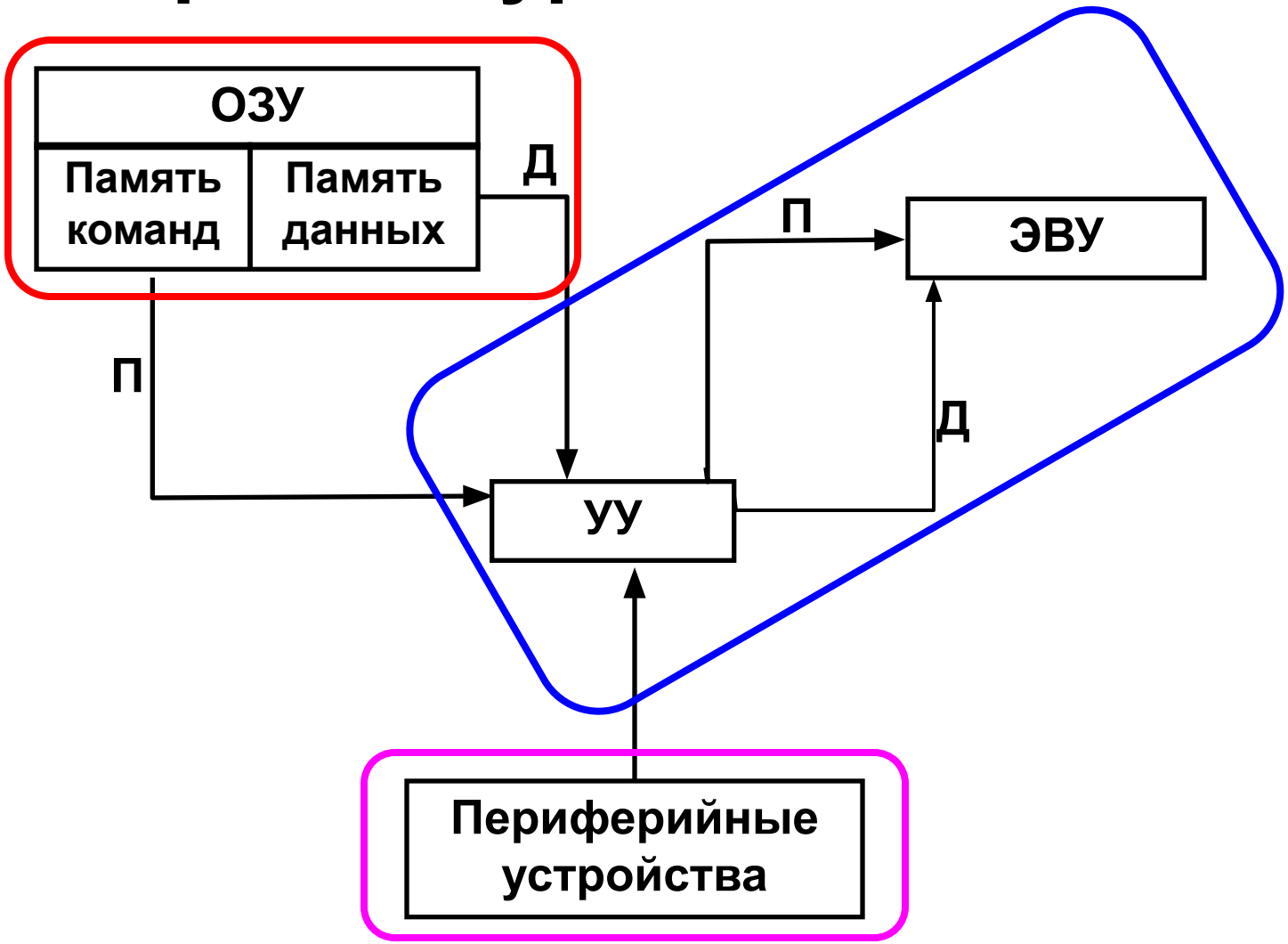
Основные понятия

- Цифра (символ) – обозначает предмет или явление и является необходимым в любой информационной системе.
- Число – упорядоченная последовательность символов, которая выражает количественное соотношение между предметами или явлениями.
- Код – упорядоченная последовательность символов, которая представляет предметы или явления.

Предпосылки создания и развития вычислительной техники

- С самого начала вычислительная техника была нацелена на устранение угроз и должна была обеспечить качественный скачок существующих электромеханических устройств.
- Для реализации подобного рода инновационных проектов требуется интеграция усилий специалистов различных областей человеческого знания.

Архитектура Фон-Неймана



Взаимосвязь компонентов

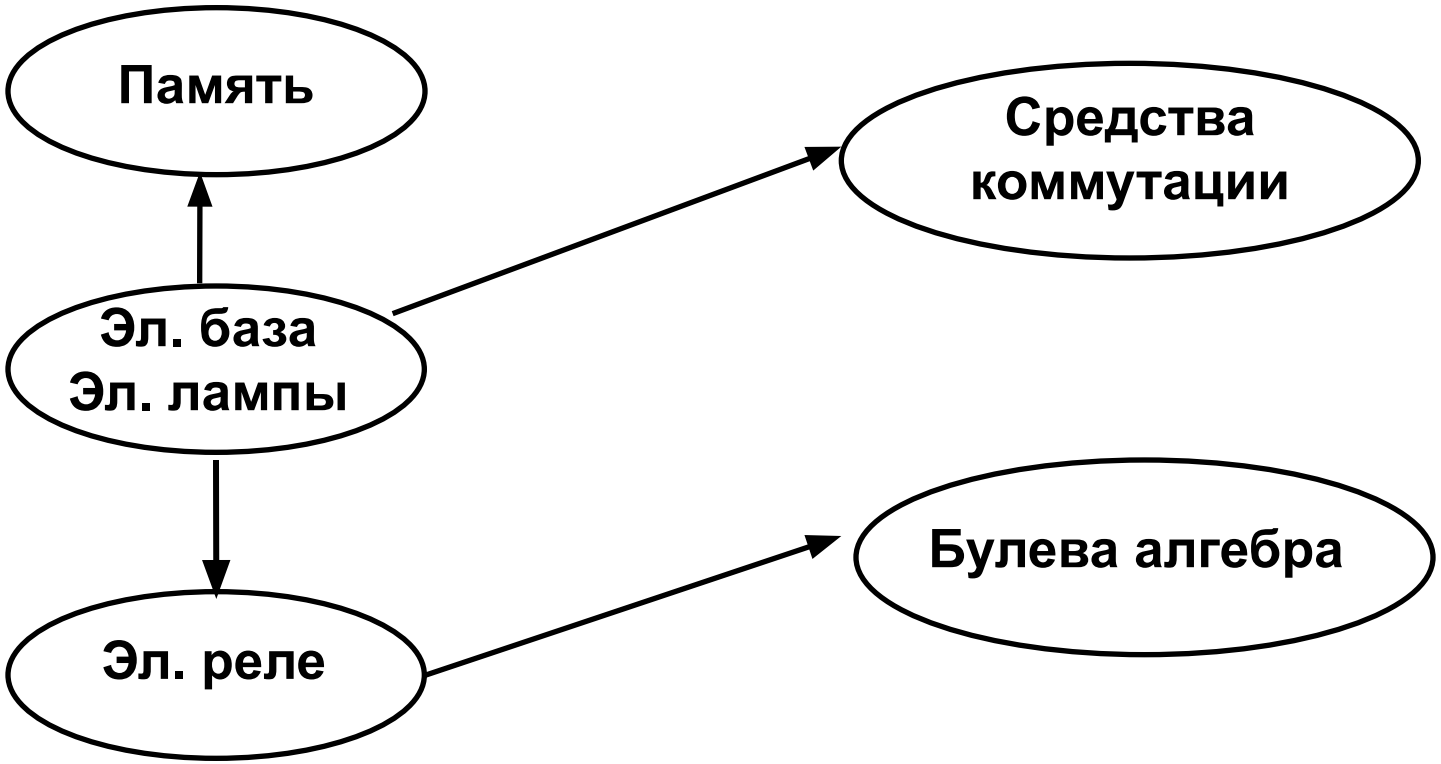
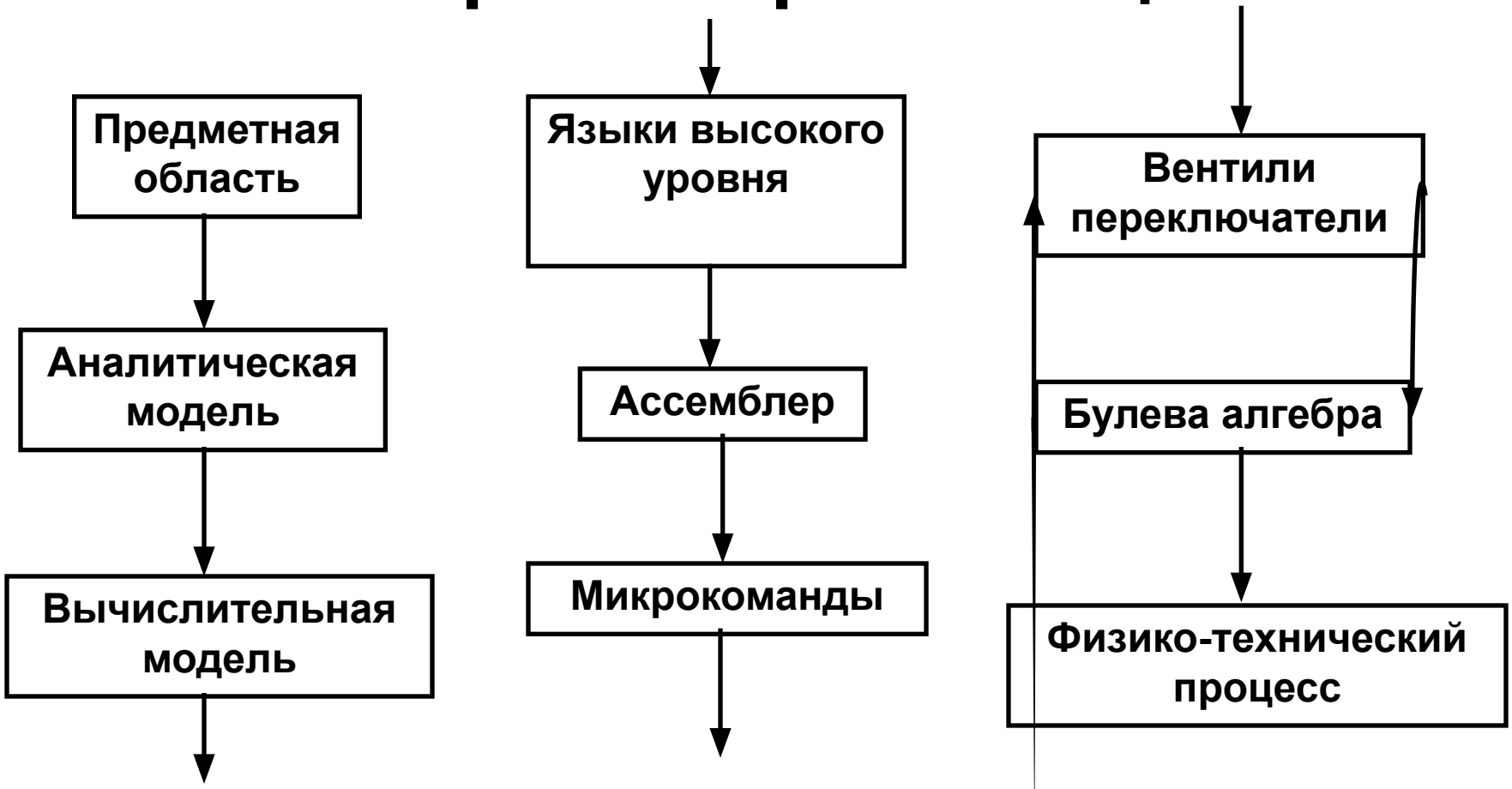


Схема погружения задания пользователя на уровень аппаратной реализации



Вычислительный процесс

- Вычислительный процесс с точки зрения автоматизации его кодов представляет собой взаимодействие на операционном устройстве двух потоков: потока команд и потока данных.

Существуют два процесса:

- Реализуемый операционным устройством;
- Перечислительный процесс, исполняемый потоком инструкций и потоком данных.

Учебный курс

Принципы построения и функционирования ЭВМ

Лекция 2

Алгоритмы и машина Тьюринга

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Основное положение

Вычислительный процесс можно реализовать только за N шагов

Проблемы:

- Почему решить задачу пользователя за один такт (цикл работы машины) практически невозможно?
- Можно ли сформулировать в предметной области задачу так, чтобы на уровне вентилей она решилась за один такт?

10

Этапы решения задачи

Центральная проблема

Центральная проблема, связанная с организацией вычислений, сосредоточена в предметной области:

- невозможно сформулировать аналитическую формулу из одного действия;
- необходимо привести задачу к стандартным действиям, реализуемым данной машиной;
- через эти стандартные действия выразить задачу предметной области.

Алгоритм

- **центральное понятие в теории вычислений представляет собой конечный набор правил выполнения некоторой процедуры, которая должна удовлетворять 3 требованиям:**

1. *массовость – гарантирует выполнение целого класса однородных и однотипных процедур*
2. *детерминированность – предполагает однозначное понимание каждой инструкции алгоритма; гарантирует воспроизводимость результата вычислений, проводимых в одинаковых условиях.*
3. *результативность – гарантирует конечность алгоритма, а значит, и времени решения задач.*

Теория алгоритмов

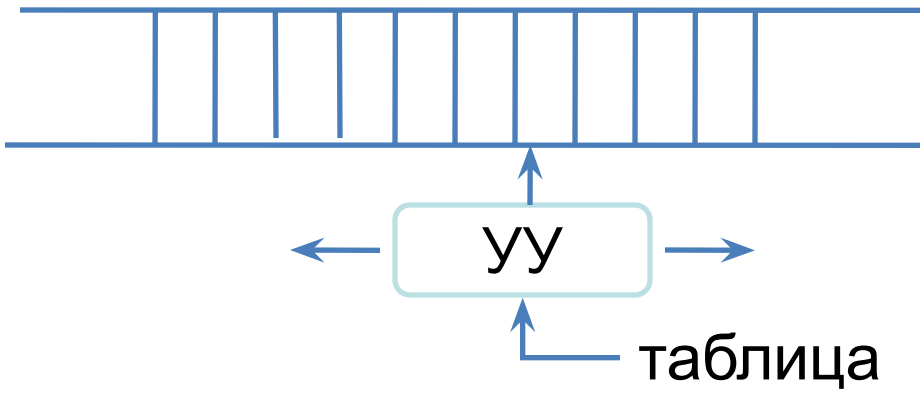
- Восходит к Давиду Гильберту
- На рубеже 20 века сформулировал мировую проблему:
  - **Можно ли построить алгоритм, создающий необходимый алгоритм к любой, точно поставленной задаче?**
- Такая постановка задачи неразрешима, т.к. фактически проблема сводится к логическому парадоксу брадобрея: для того, чтобы такой алгоритм существовал, его не должно быть

Соглашение об алгоритме

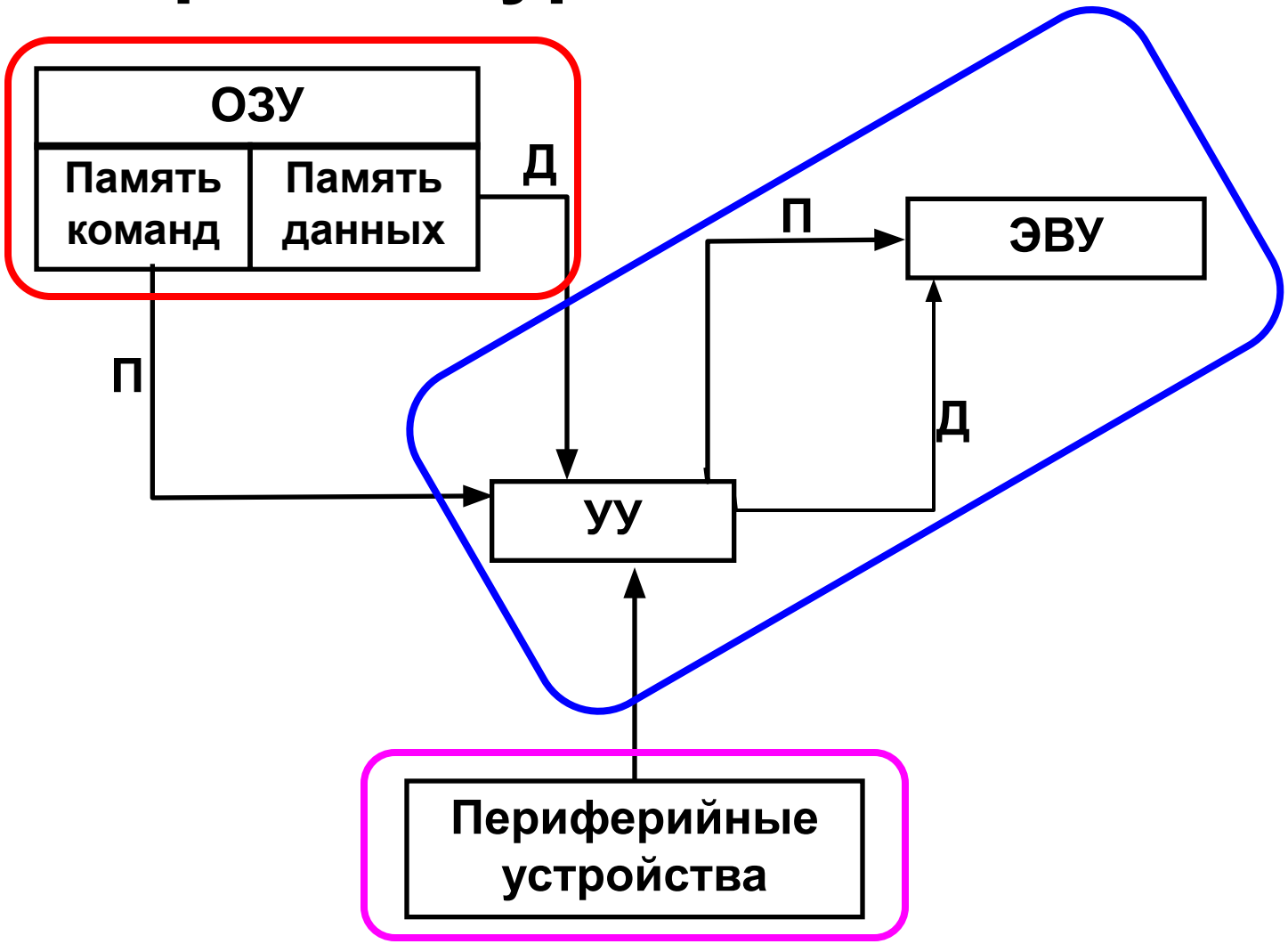
- Вычислительным алгоритмом принято считать все то, что представимо в виде машины Тьюринга

Машина Тьюринга

- Представляет собой бесконечную ленту, разделенную на ячейки.
- Имеет управляющее устройство, которое перемещается в двух направлениях.
- В управляющем устройстве содержится таблица, которая описывает порядок действий.



Архитектура Фон-Неймана



Этапы решения задачи

Отличия ЭВМ и машины Тьюринга

- Главное отличие машины Тьюринга от ЭВМ – бесконечная лента
- В отличие от машины Тьюринга память реальных машин всегда конечна и ее ограничения удается преодолеть путем организации циклов if – если и for – делай до тех пор пока

Как работает машина Тьюринга

На ленту можно записать слова – упорядоченную последовательность символов: | - единица, * - ноль

* * | | | - код некоторого явления

На каждом шаге своей работы машина может извлекать символы из ячейки, освобождая ее, и записывать другой символ, в соответствии с правилом, отвечающим как состоянию УУ, так и считанному символу

Управляющая таблица

- В современных процессорах “управляющая таблица” хранится в ПЗУ, в которое информация заносится изготовителями процессора, т. к. она регламентирует порядок выполнения ассемблерных команд

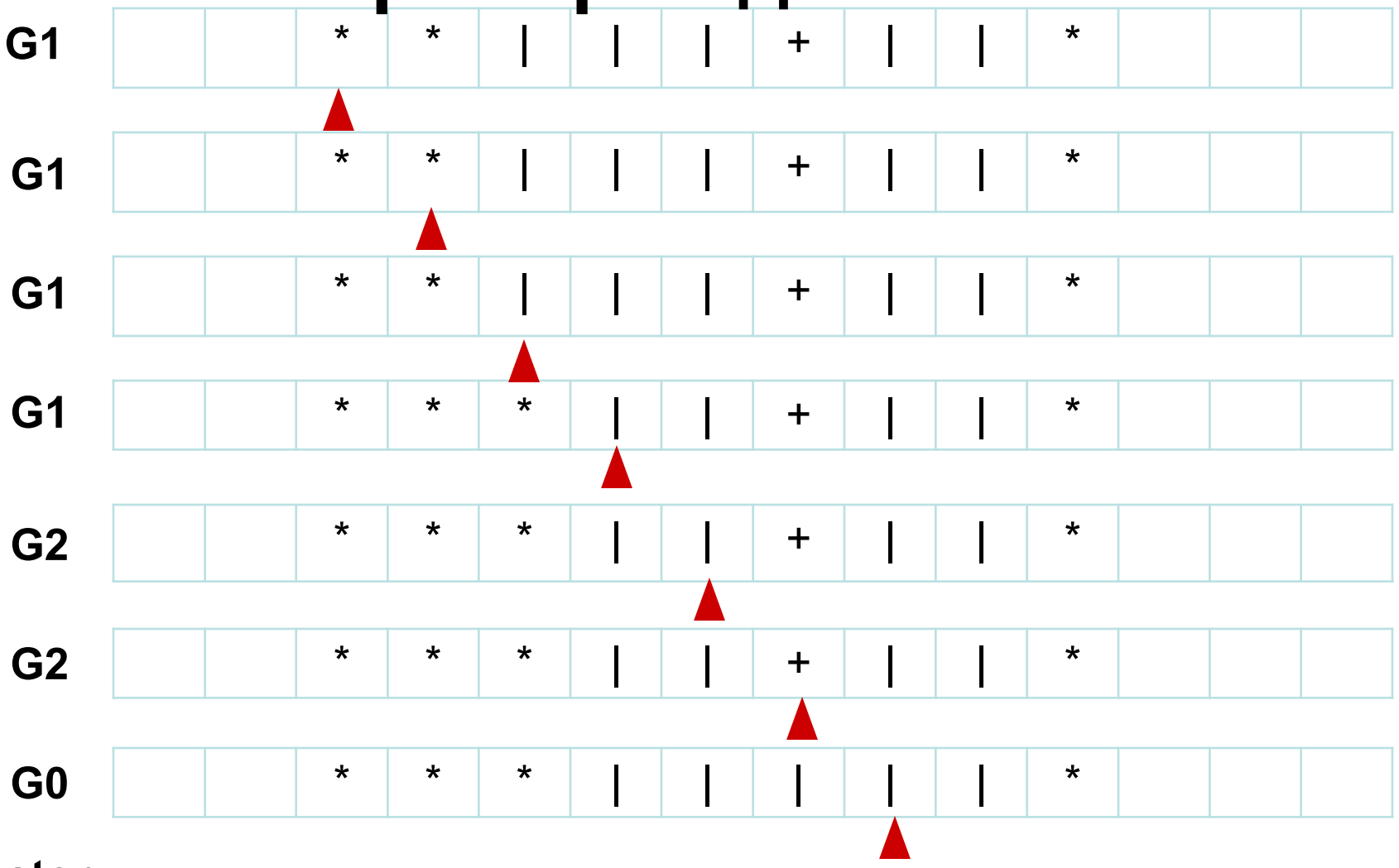
Пример работы машины Тьюринга

- G0, G1, G2 – состояния устройства управления

символы	G0	G1	G2
*	*, G0, stop	*, G1, right	|, G2, stop
|	|, G0, stop	*, G2, right	|, G2, right
+	+, G0, stop	-	|, G0, stop

- G0 – исходное состояние любой машины, абсолютный сброс
- Существует внешнее воздействие – BIOS, которое сдвигает машину из состояния G0 в другое состояние

Пример вида ленты



Результат работы

- Машина заменяет одни символы на другие в соответствии с некоторой таблицей, но не вычисляет
- Результат, полученный машиной, принадлежит интерпретации
- Вывод: поскольку любая машина занимается преобразованиями символов, то после любого этапа вычислений наступает этап интерпретации его результатов
- Герменевтика – наука об интерпретации


Угрозы
Задачи
Требуемая производительность
Предоставляемая производительность
Предпосылки
Электромеханические вычислительные устройства
Схемотехнические
Системные
Распараллеливание вычислений
Минимизация системных, временных издержек на управления ходом вычислительного процесса

Общие выводы

- Схемотехнические факторы повышения производительности предполагают развитие технологий производства элементной базы
- Глобальные национальные задачи, связанные с развитием вычислительной техники и микроэлектроники, можно реализовать только с использованием консолидирующей роли военных бюджетов

Учебный курс

Принципы построения и функционирования ЭВМ

Лекция 3

Механизмы реализации алгоритмов на низшем уровне

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Список литературы

- «Организация ЭВМ и систем», Цилькер Б.Я., Орлов С.А., 2004 – сегодня.
- «Электронно-вычислительные машины и системы», Каган, 1985 – 20 лет назад.
- «Введение в отказоустойчивые технологии высокопроизводительных вычислительных систем субмикронного, супромолекулярного и нанометрового диапазона», под ред. Алакоза Г.М., 2008 – на 20 лет вперед.

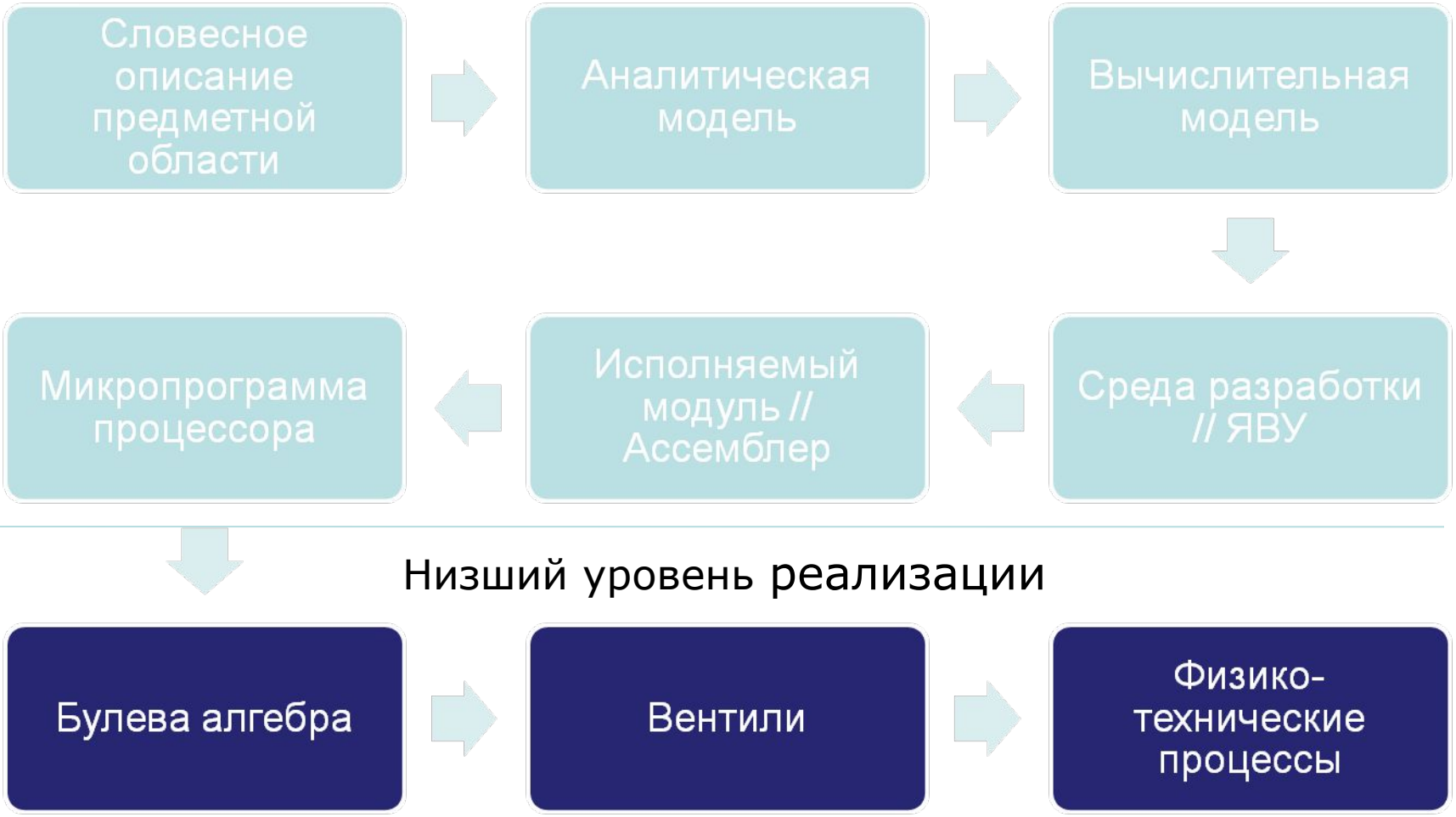
Для сравнения

Проекты молодых японских ученых в рамках проекта ERATO в 80-х – начале 90-х годов:

- Создание совершенного кристалла
- Работа с ультрамалыми частицами (4-20нм)
- Передача биоинформации
- Наномеханизмы
- Сверхчувствительные фотоприемники
- Молекулярно-динамические ансамбли

Тогдашняя теория обгоняет сегодняшнюю практику.

Схема погружения задания пользователя



Общий принцип производства

Технологические возможности любого оборудования нужно использовать не более чем на 40-60%

Страна	Используемые возможности оборудования	Годный выход
Япония	Не более 40%	Не менее 80%
США	Не более 60%	Не менее 65%
СССР (на момент распада)	Более 80%	Приблизительно **2-5%**

Логика

- Булева алгебра является частью символьной логики.
- Символьная логика занимается разработкой и изучением правил преобразования символов.
- Символьная логика имеет своей основой Аристотелеву логику (далее - АЛ)

Правила (аксиомы) АЛ

Название	Краткая формулировка	Применение в области ВС
Аксиома тождества	Мысль принимается однозначной и неизменной	Отвлечение от единиц, сведение понятий к числам
Исключение третьего	Один логический вопрос – однозначный верный ответ	Позволяет сведение к булевой алгебре
Аксиома непротиво-речивости	Верное – верно; мысль протекает непротиворечиво	Позволяют и регламентируют построение символьных формул
Правило вывода	Мысль верна только если вытекает из другой верной мысли	

Схема создания вычислительных устройств

Преобразования осуществляет **кремниевый компилятор**

Функции двух переменных

x_2	x_1	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1
-	-	0	$x_2 * x_1$	$x_2 * \overline{x}_1$	x_2	$\overline{x}_2 * x_1$	x_1	$\overline{x}_2 x_1 + x_2 \overline{x}_1$	$x_1 + x_2$

Недостающие функции (с 8й по 15ю) формируются по правилу

$$F_i = \overline{F_{15-i}}$$

В общем случае количество функций не более n аргументов

$$\left|\left\{F_i(x_1,\ldots,x_n)\right\}\right| = 2^{2^n}$$

Требования к устройству

Необходима реализация **всех** возможных функций

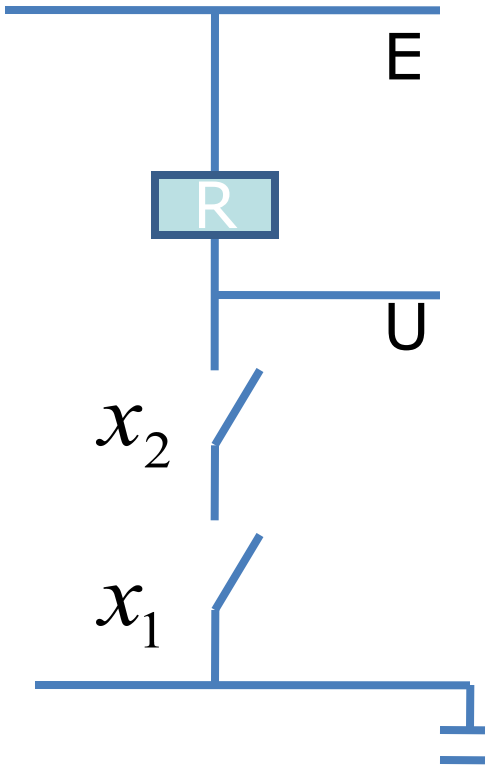
Используется **базисный** набор

Используемые базисы

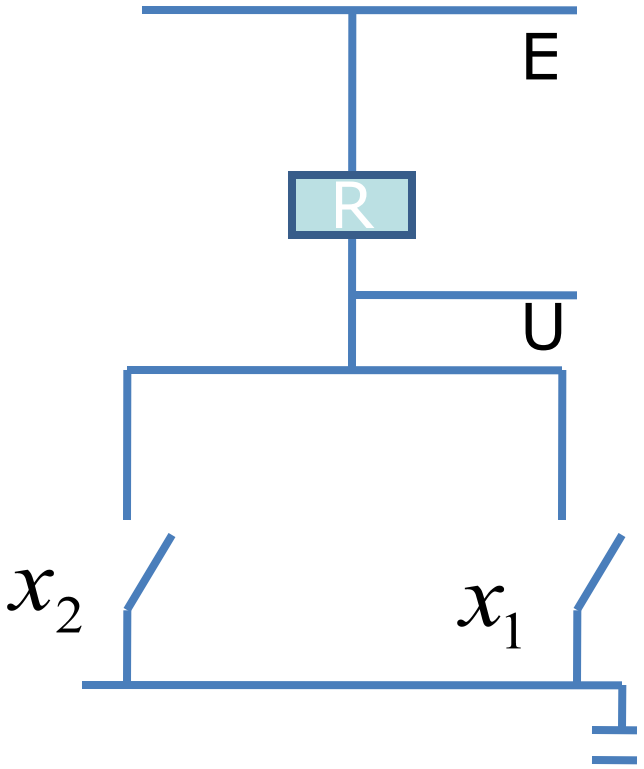
«И-НЕ» «ИЛИ-НЕ»

Устройство вентилей

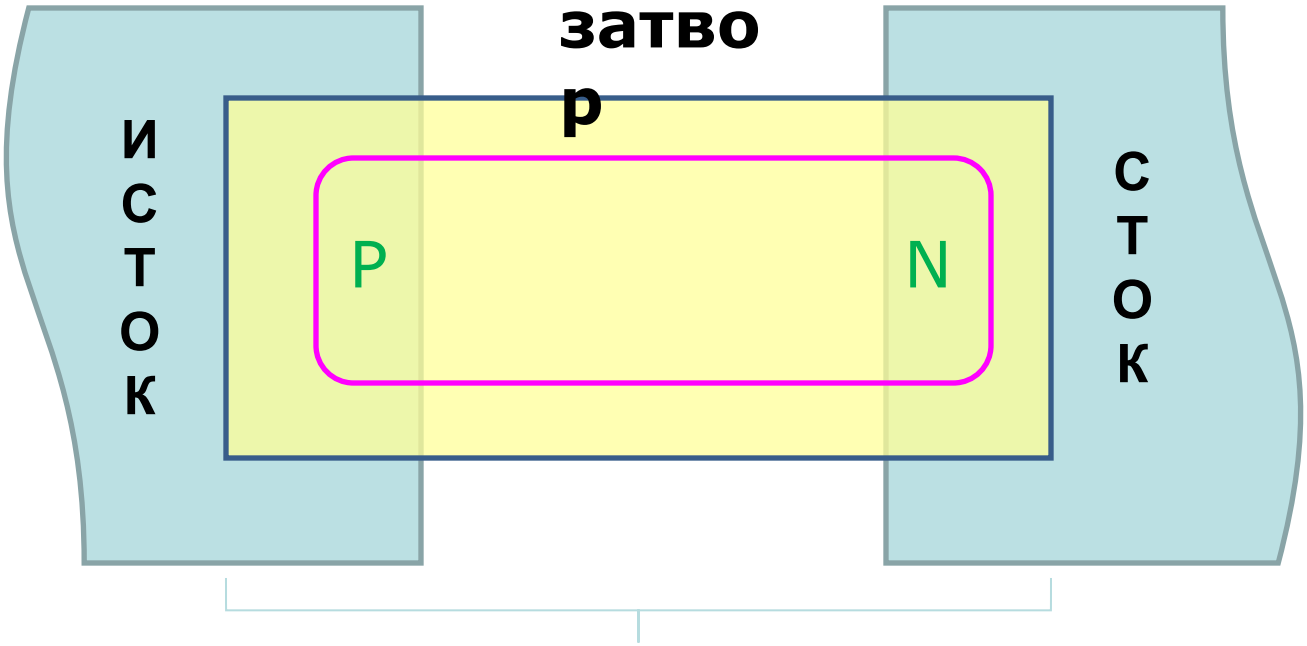
«и
»



«или
»



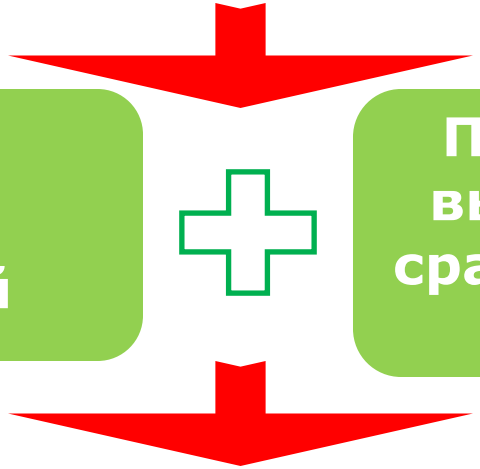
Полупроводниковый вентиль



Размер имеет значение

Ширина контактов = уровень технологии =
= топологическая норма

Уменьшение линейных размеров



Квадратичный выигрыш в пространстве/ функциональной интеграции

Пропорциональный выигрыш в скорости срабатывания/тактовой частоте

Интегральный показатель качества возрастает на 3 порядка

Учебный курс

Принципы построения и функционирования ЭВМ

Лекция 4

Схемы вентилей

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Полупроводниковый вентиль

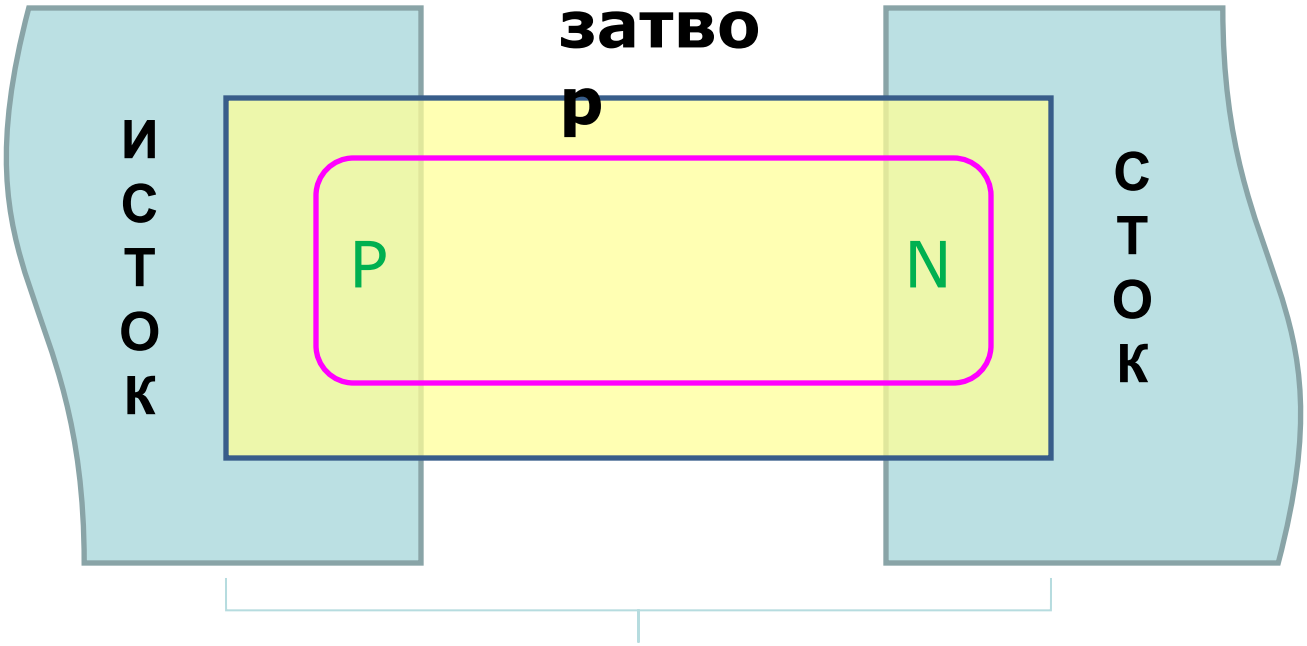


Схема вентиля «и-не» (последовательное соединение)

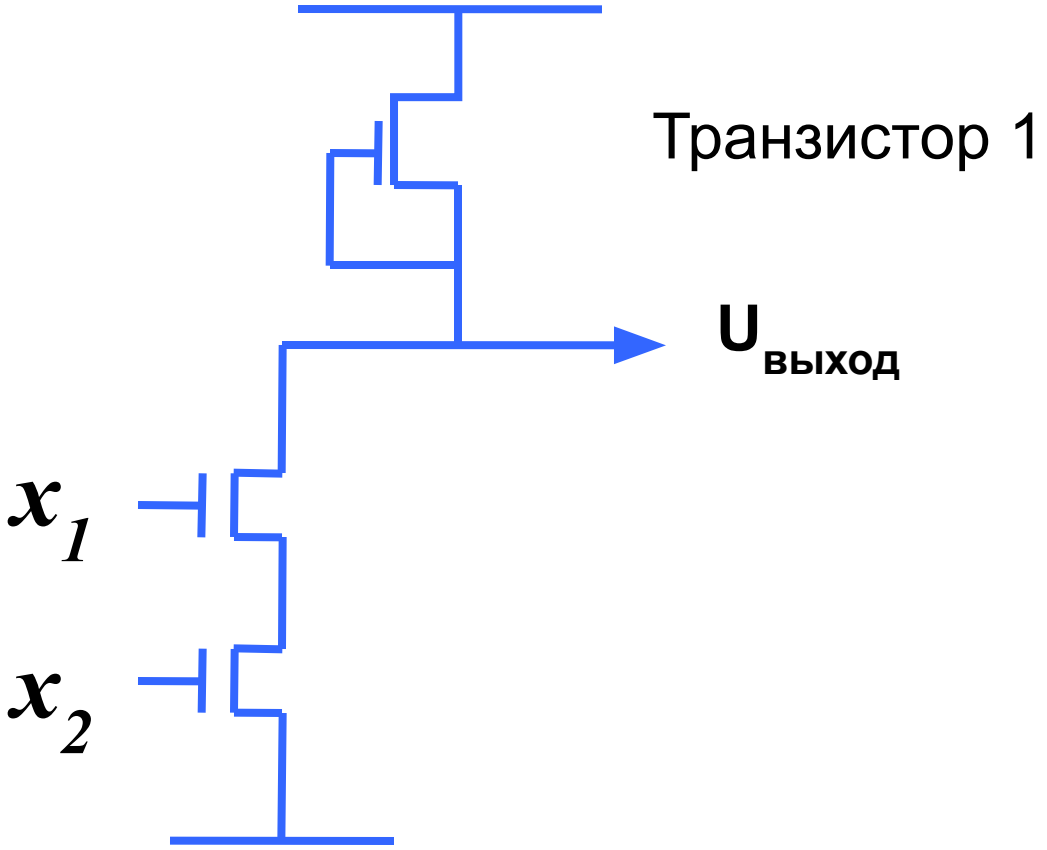
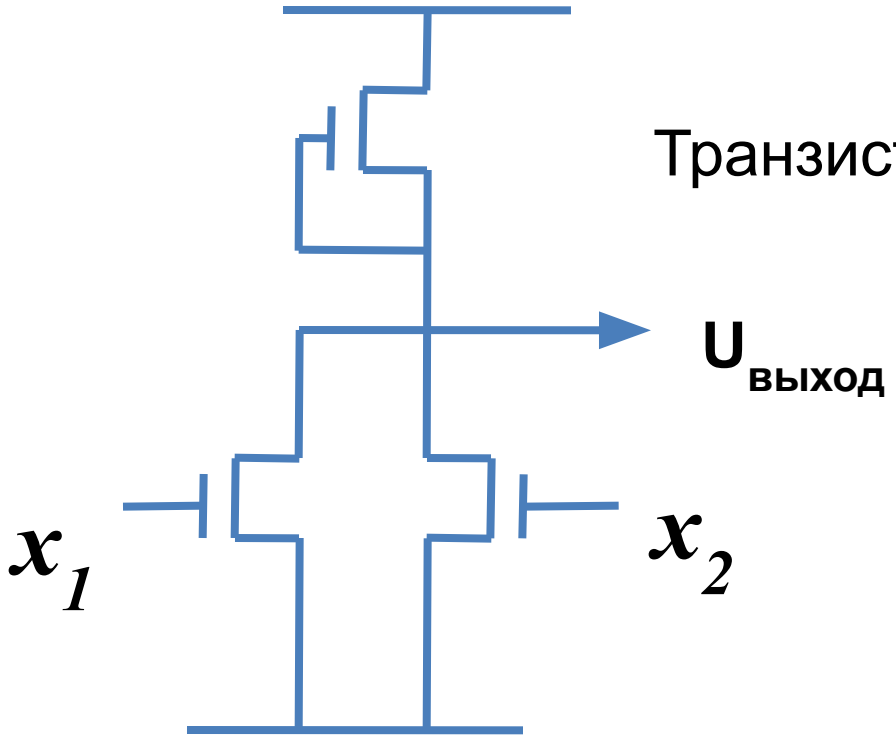
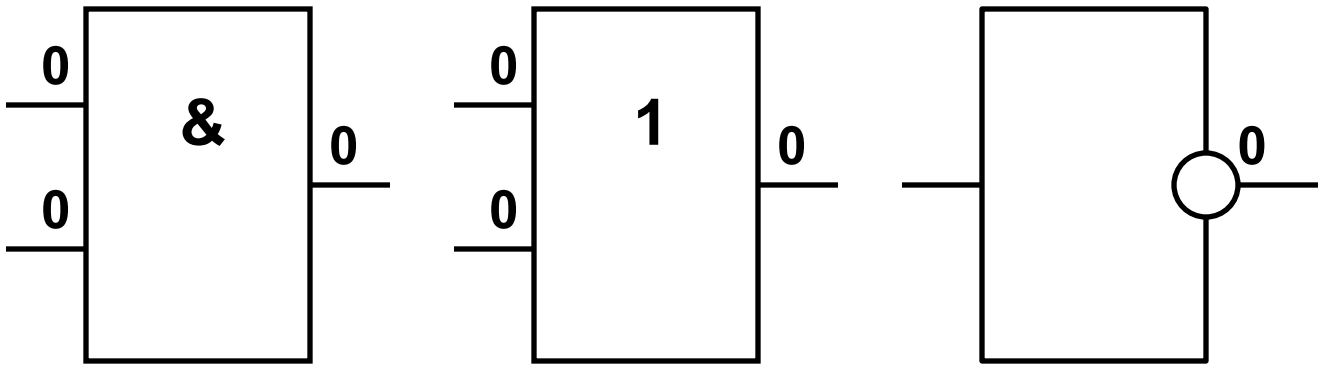


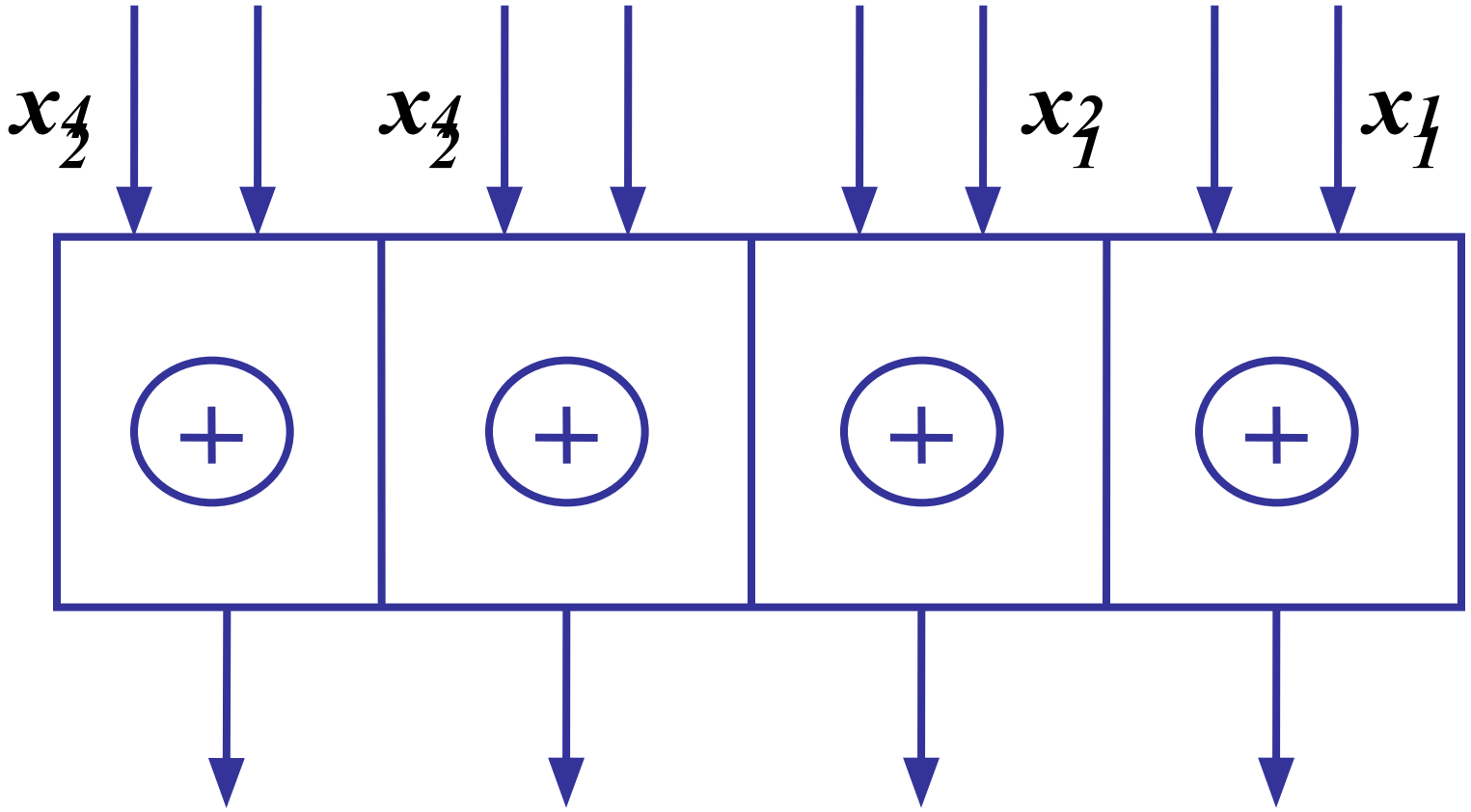
Схема вентиля «или-не» (параллельное соединение)



Логические схемы вентилей



Орисания устройства для функции XOR



XOR

Операционное устройство для функции XOR

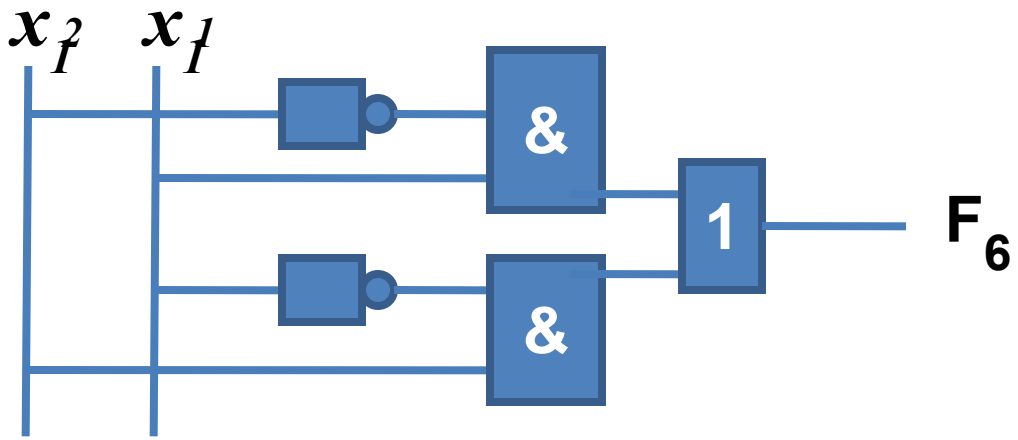
x_2	x_1	+
0	0	0
0	1	1
1	0	1
1	1	0

Таблица истинности

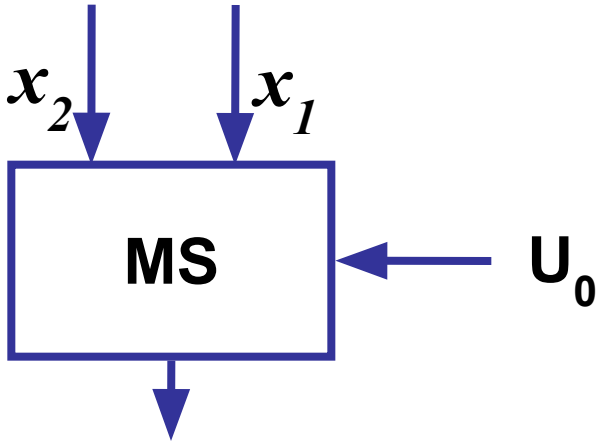
$F_6 = x_1\bar{x}_2 + \bar{x}_1 x_2$

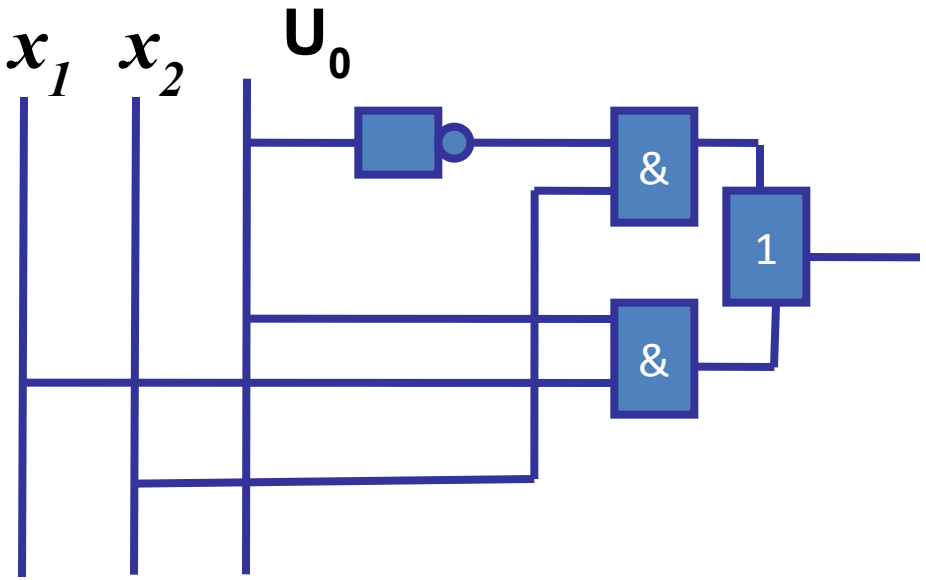
Алгебраическое выражение

Реализация одного узла



Коммутационное устройство типа мультиплексор





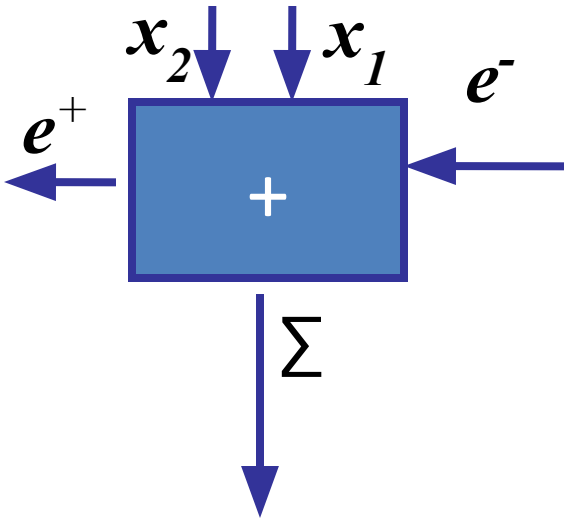
$$F=\begin{cases} x_1, \text{ если } U_0=1 \\ x_2, \text{ если } U_0=0 \end{cases}$$

$$F= x_1 U_0 + x_2 \overline{U}_0$$

48

Одноразрядный сумматор

Перенос в следующий разряд



Перенос из предыдущего разряда

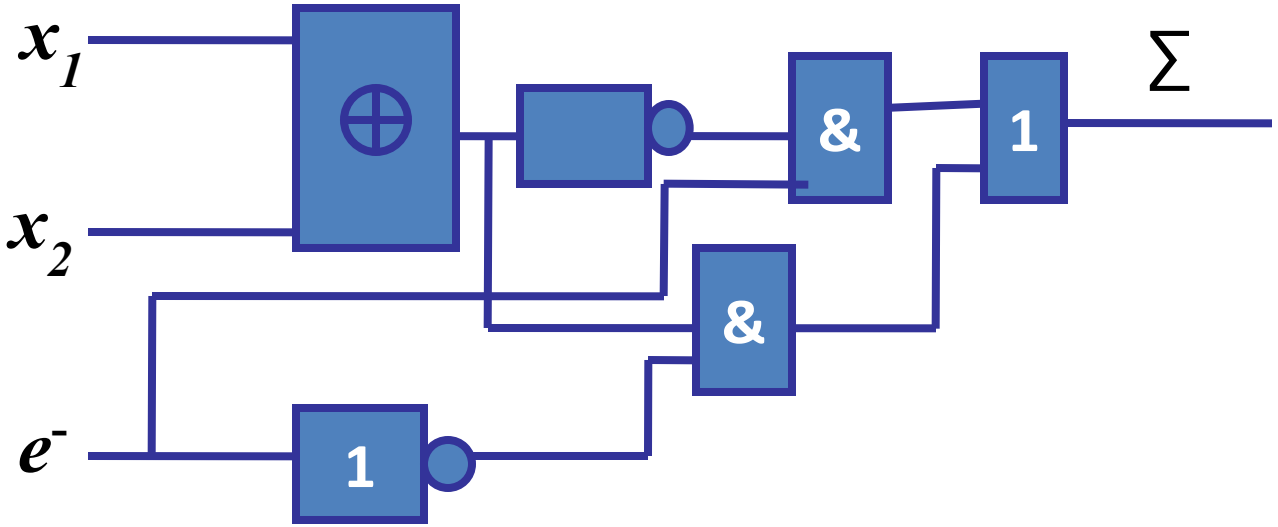
Таблица истинности для функции одноразрядного сумматора

e^-	x_2	x_1	$\sum$	e^+
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\sum = \begin{cases} x_1 \oplus x_2, \text{ если } e^- = 0 \\ \overline{x_1 \oplus x_2}, \text{ если } e^- = 1 \end{cases}$$

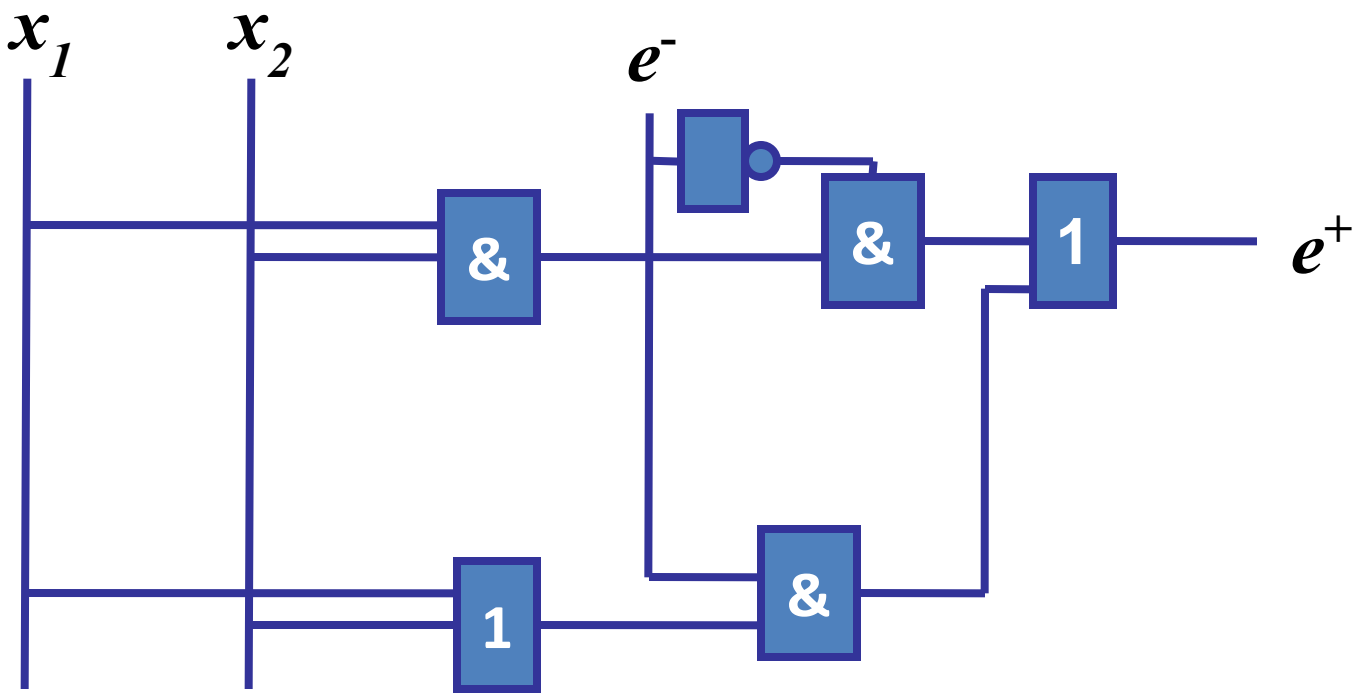
50

Один разряд сумматора

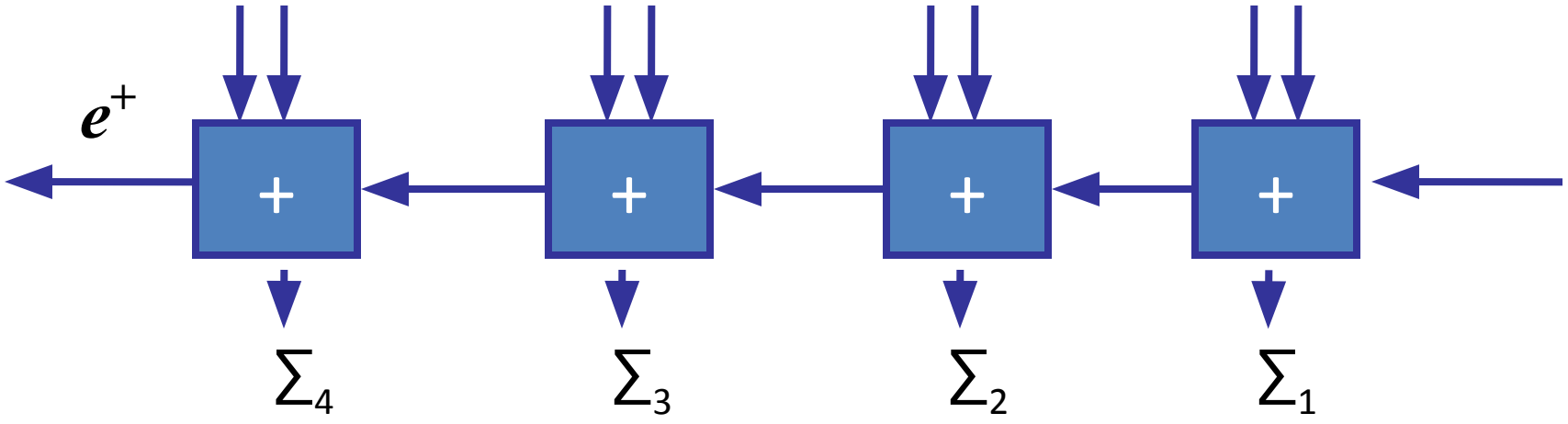


$$e^+ := \begin{cases} x_1 * x_2,\ e^- = 0 \\ x_1 + x_2,\ e^- = 1 \end{cases}$$

Логическая схема



Общая схема



Учебный курс

Принципы построения и функционирования ЭВМ

Лекция 5

Синтез цифровых устройств

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Роль и место булевой алгебры

Последовательность действий синтеза вычислительных устройств:

1. Словесное описание функции
2. Таблица истинности
3. Алгебраическое выражение
4. Логическая схема

Булева алгебра рассматривается как абстрактная модель аппарата, описывающая его работу.

Последовательность действий в аппаратной среде

1. Физико-технический процесс
2. Транзистор
3. Вентиль
4. Узел
5. Блок
6. Устройство

Минимизация алгебраического выражения

Минимизация алгебраического выражения проводится по двум критериям:

1. Минимум аппаратных затрат (вентилей)
2. Минимум времени задержки (в узле, блоке или устройстве)

В современной микроэлектронике доминирует второй критерий, т.е. время задержки в системе стараются снизить в ущерб количеству вентилей

Законы Булевой алгебры

Законы эквивалентности

$X + 1 = 1$

$X + 0 = X$

$X * 1 = X$

$X * 0 = 0$

$X = \overline{\overline{X}}$

$X * X = X$

$X + X = X$

Применение и физический смысл

- **Сочетательный (ассоциативный) закон:**

X3 + (X2 + X1) = (X3 + X2) + X1

X3 * (X2 * X1) = (X3 * X2) * X1

- **Переместительный (коммутативный) закон:**

X3 + X2 + X1 = X3 + X2 + X1

X3 * X2 * X1 = X3 * X2 * X1

- **Распределительный (дистрибутивный) закон:**

1 рода : X3 * (X2 + X1) = X3 * X2 + X3 * X1

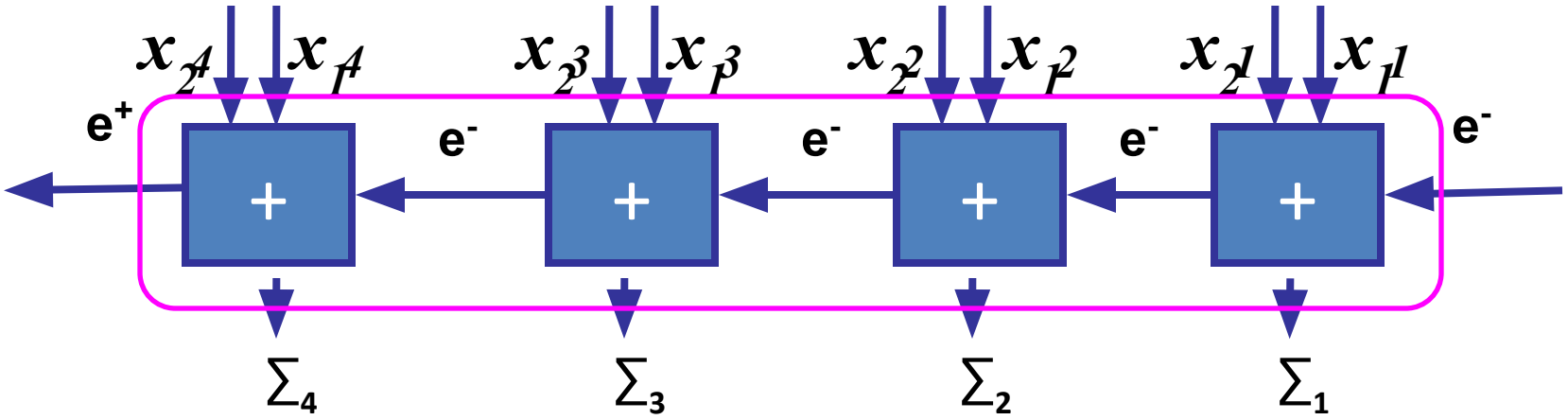
2 рода : X3 + (X2 * X1) = (X3 + X2) * (X3 + X1)

Правила Де-Моргана

$$X2 * X1 = \overline{\overline{X}2 + \overline{X}1}$$

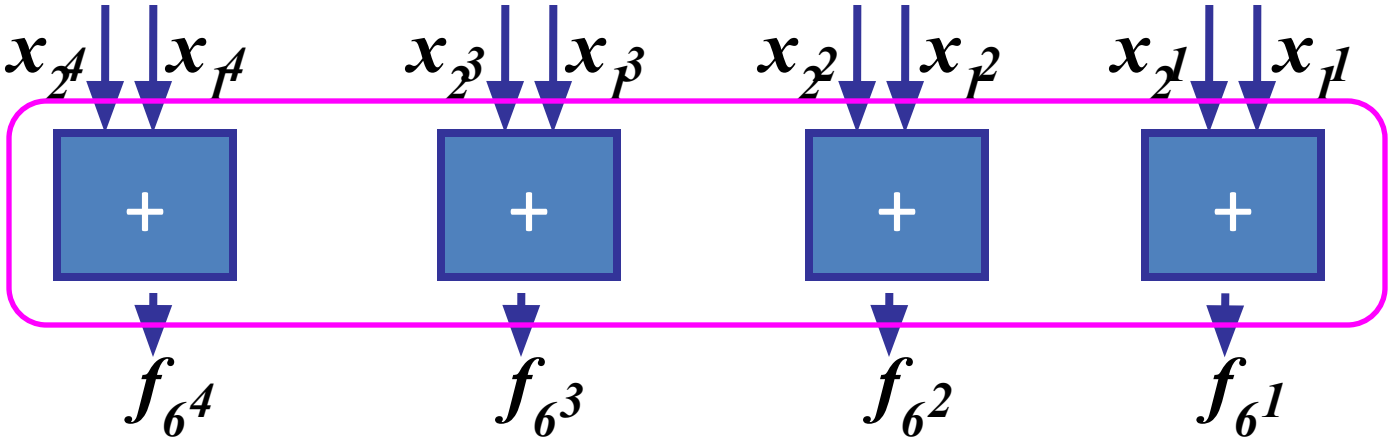
$$X1 + X2 = \overline{\overline{X}1 * \overline{X}2}$$

Многоразрядный сумматор

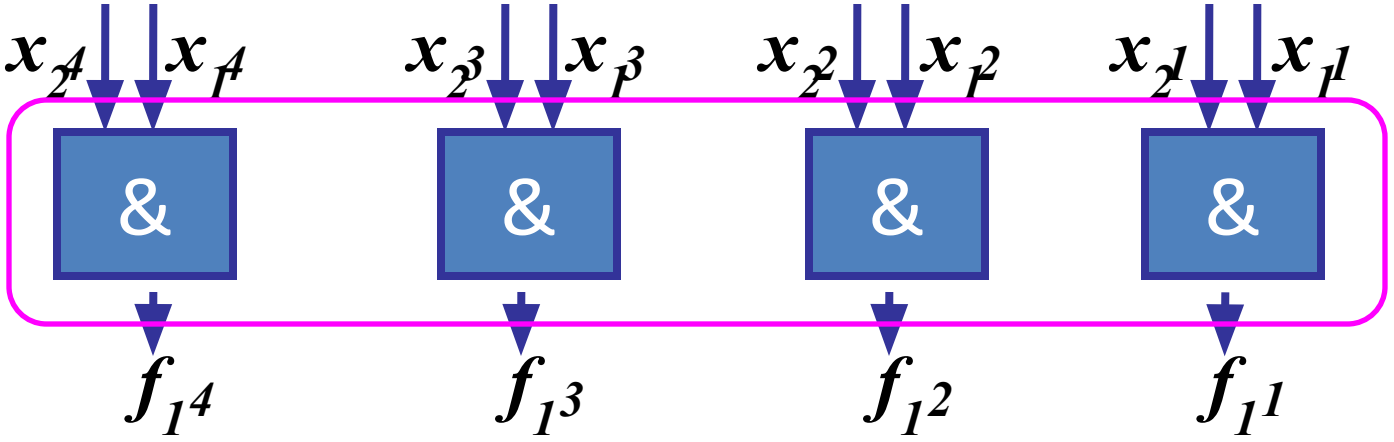


Команда ADD

Сравнение поразрядно



Команда XOR



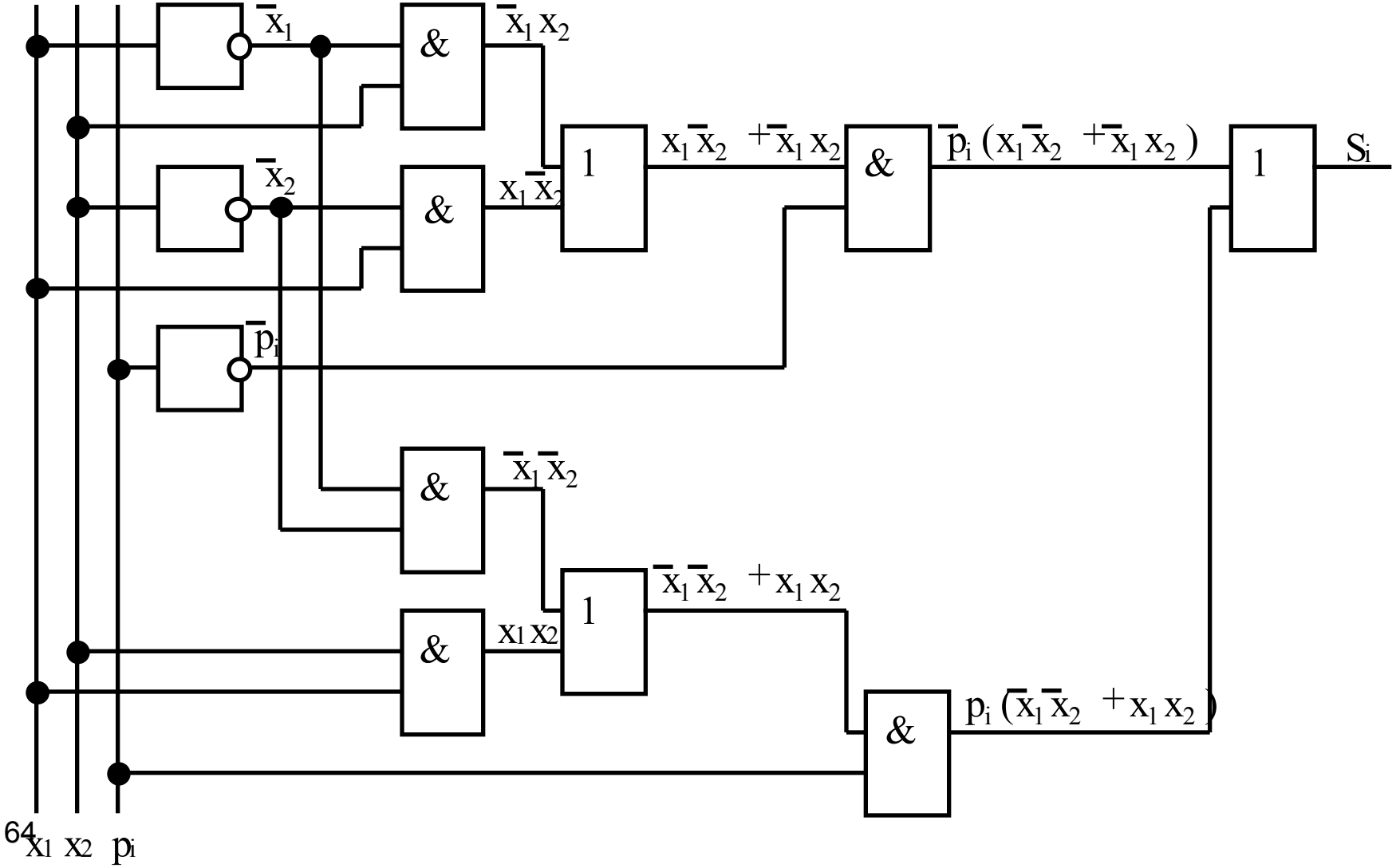
Команда AND

Таблица истинности для функции одноразрядного сумматора

e^-	x_2	x_1	$\sum$	e^+
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\sum = \begin{cases} x_1 \oplus x_2, \text{ если } e^- = 0 \\ \overline{x_1 \oplus x_2}, \text{ если } e^- = 1 \end{cases}$$

Функциональная схема полного одноразрядного сумматора



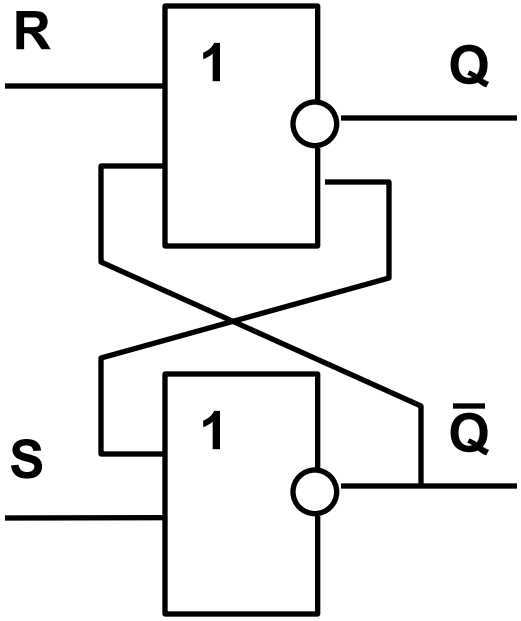
Устройства коммутации

Все рассмотренные узлы являются:

- Комбинационные автоматами, если в них реакция зависит только от содержимого входных переменных.
- Конечными автоматами, если реакция зависит от содержимого входных переменных и внутреннего состояния.

Триггеры

Асинхронный R-S триггер



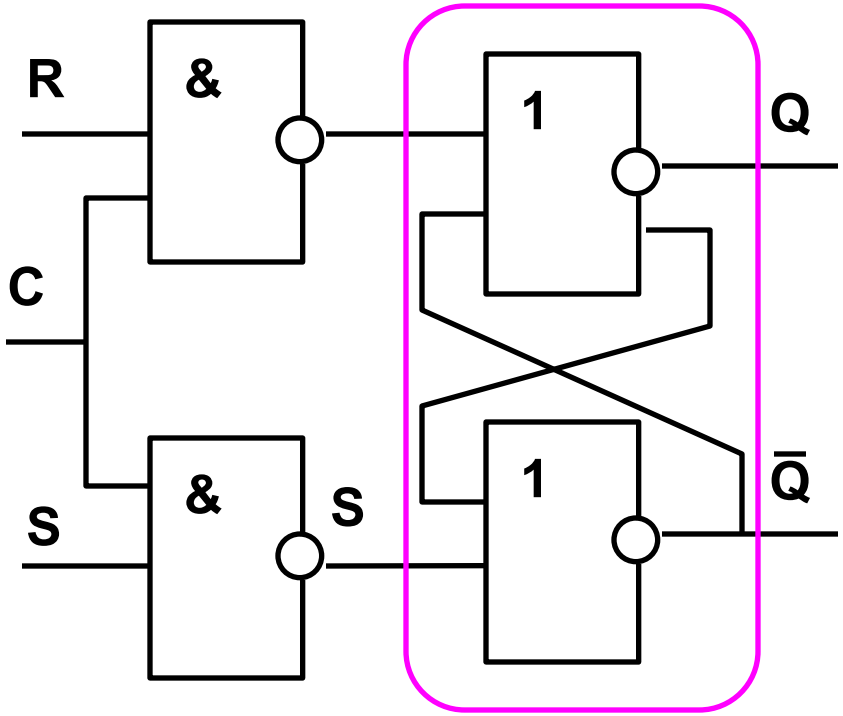
S – set (установить)

R – reset (сбросить)

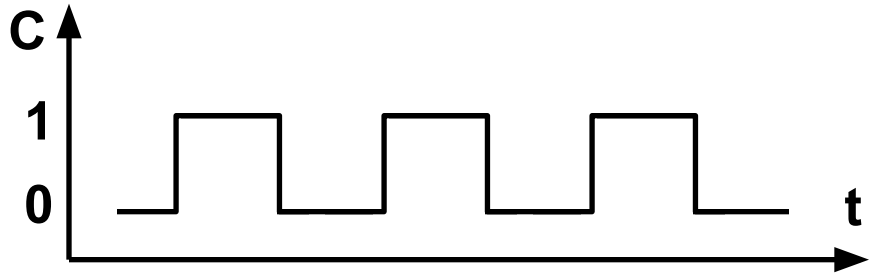
Функция R-S триггера

S	R	Q(t+1)
0	0	0
0	1	0
1	0	1
1	1	*

Синхронный R-S триггер

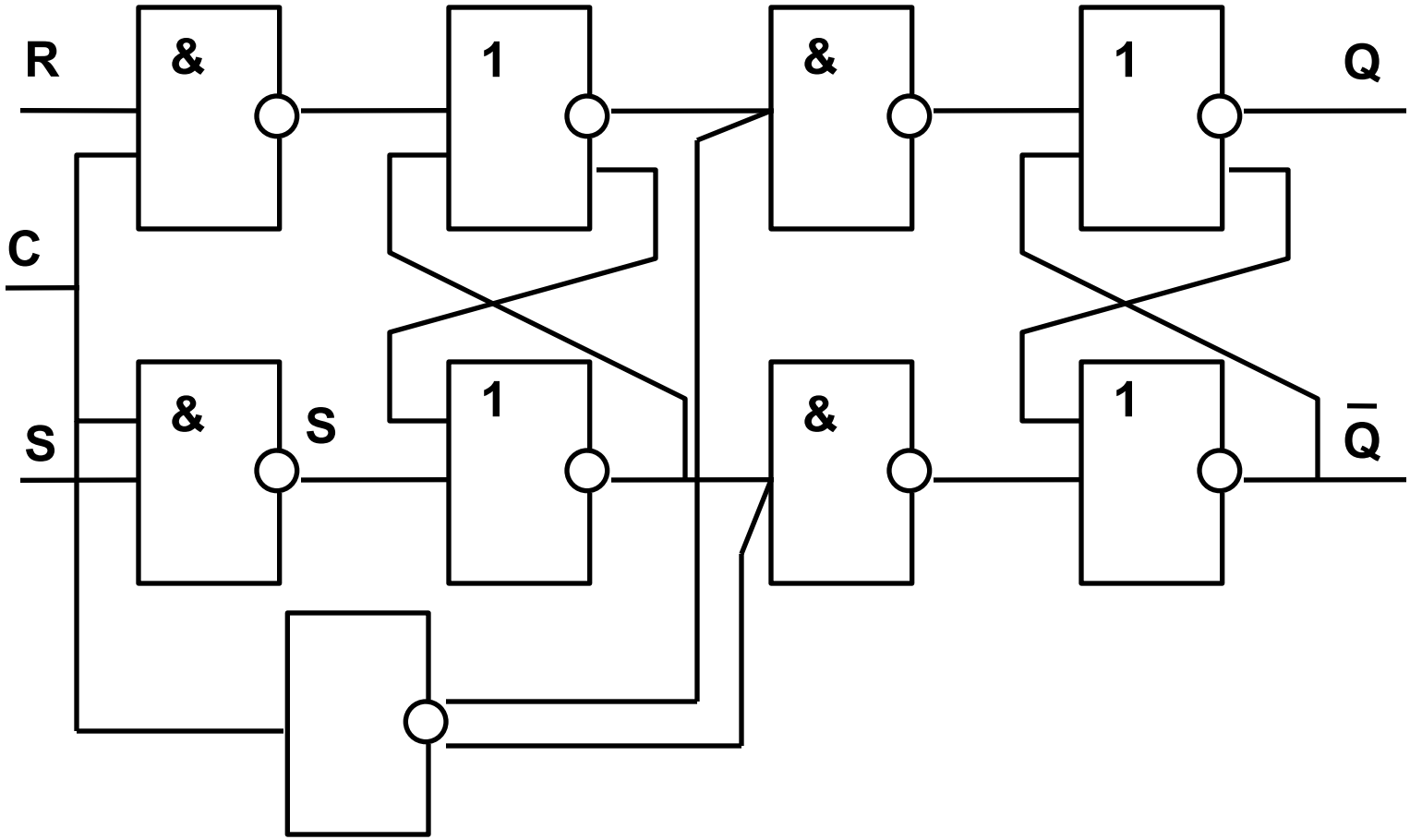


Сигнал «C» определяет, в какой момент времени можно изменить состояние триггера



Временная диаграмма

Двухполупериодный RS-триггер



Учебный курс

Принципы построения и функционирования ЭВМ

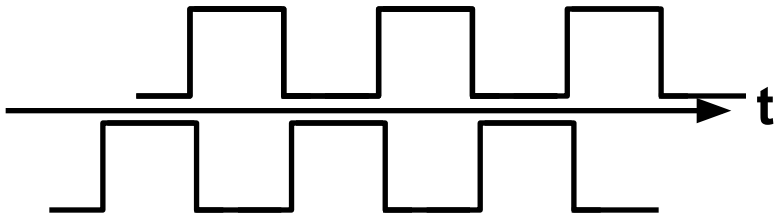
Лекция 6

Триггеры и регистры

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Двухполупериодный триггер

- Информация в двухполупериодном или двухступенчатом триггере продвигается только до внутренних выходов в первый полупериод, и только на выходе появляется следующий момент времени.

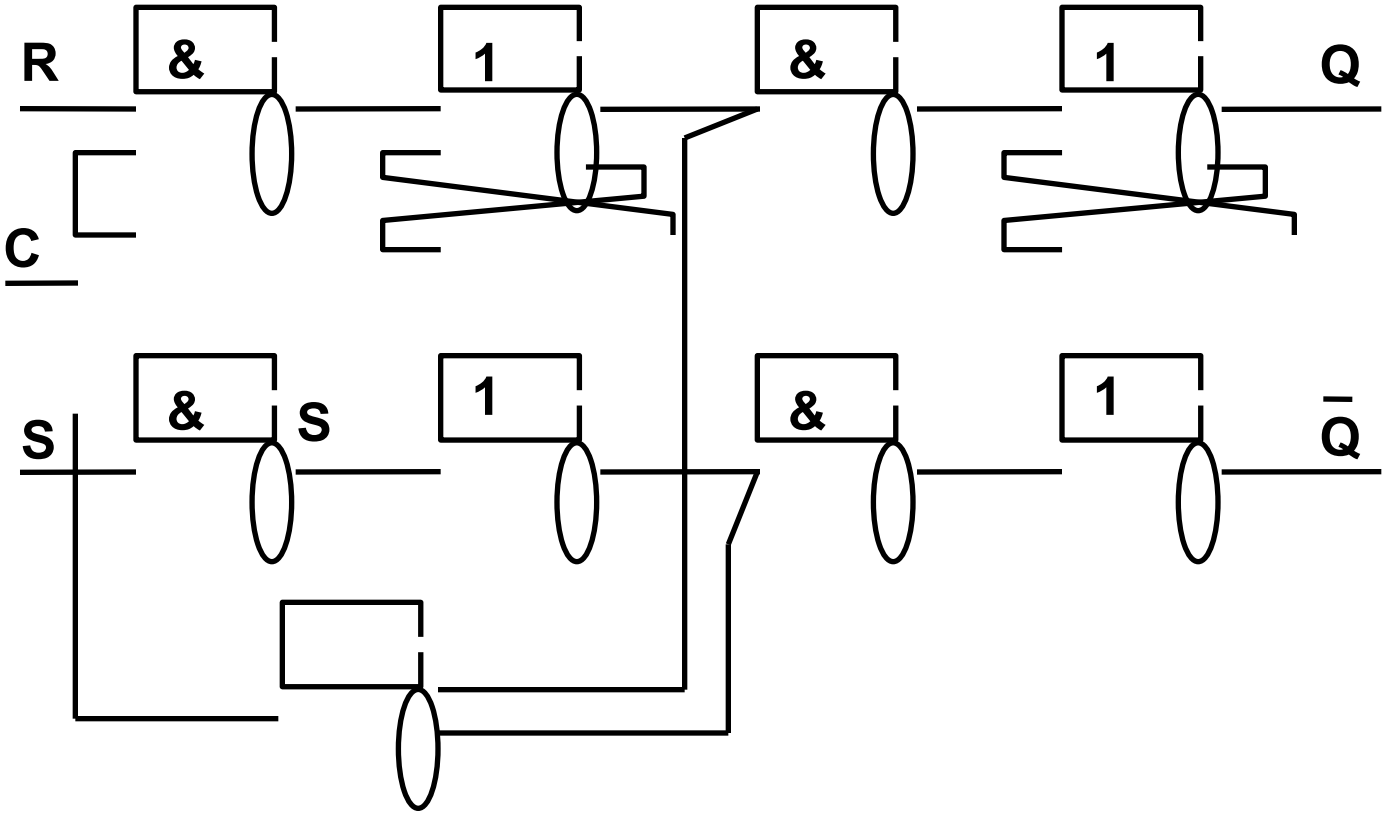


C – Фаза

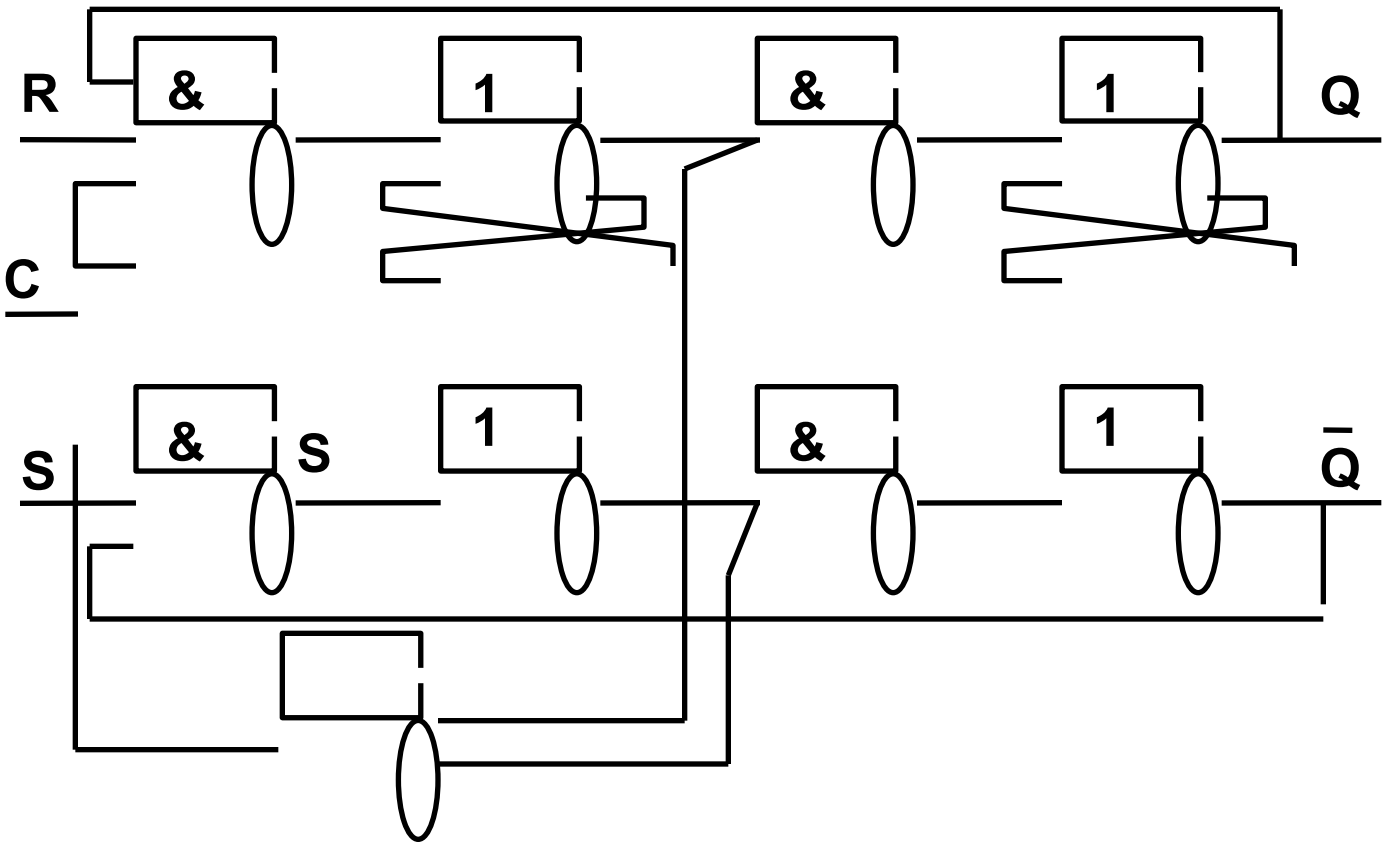
$\overline{C}$ – Противофаза

- Если Є – 0, то ни одно событие R и S не может повлиять на состояние выхода.

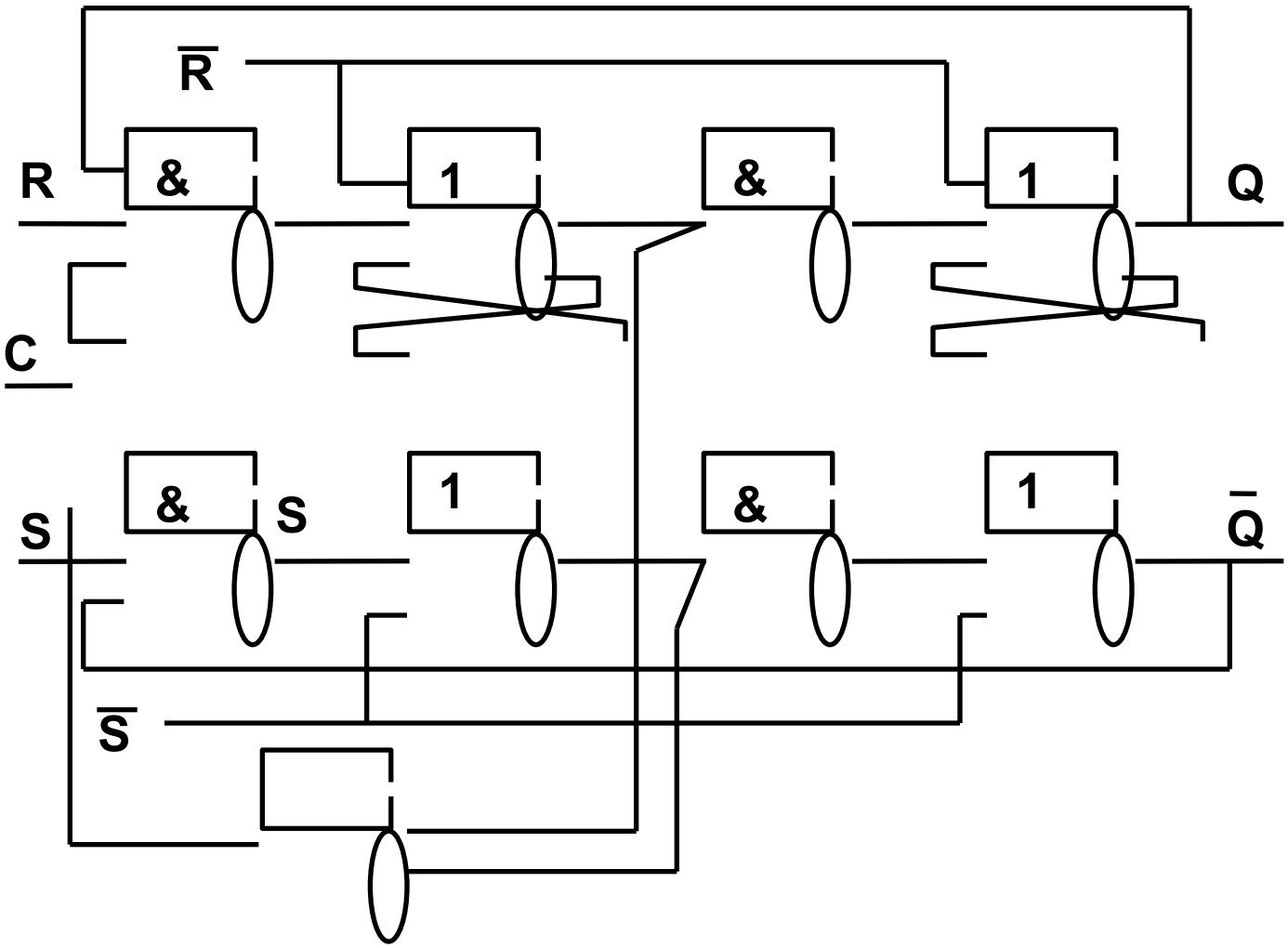
Двухполупериодный RS-триггер



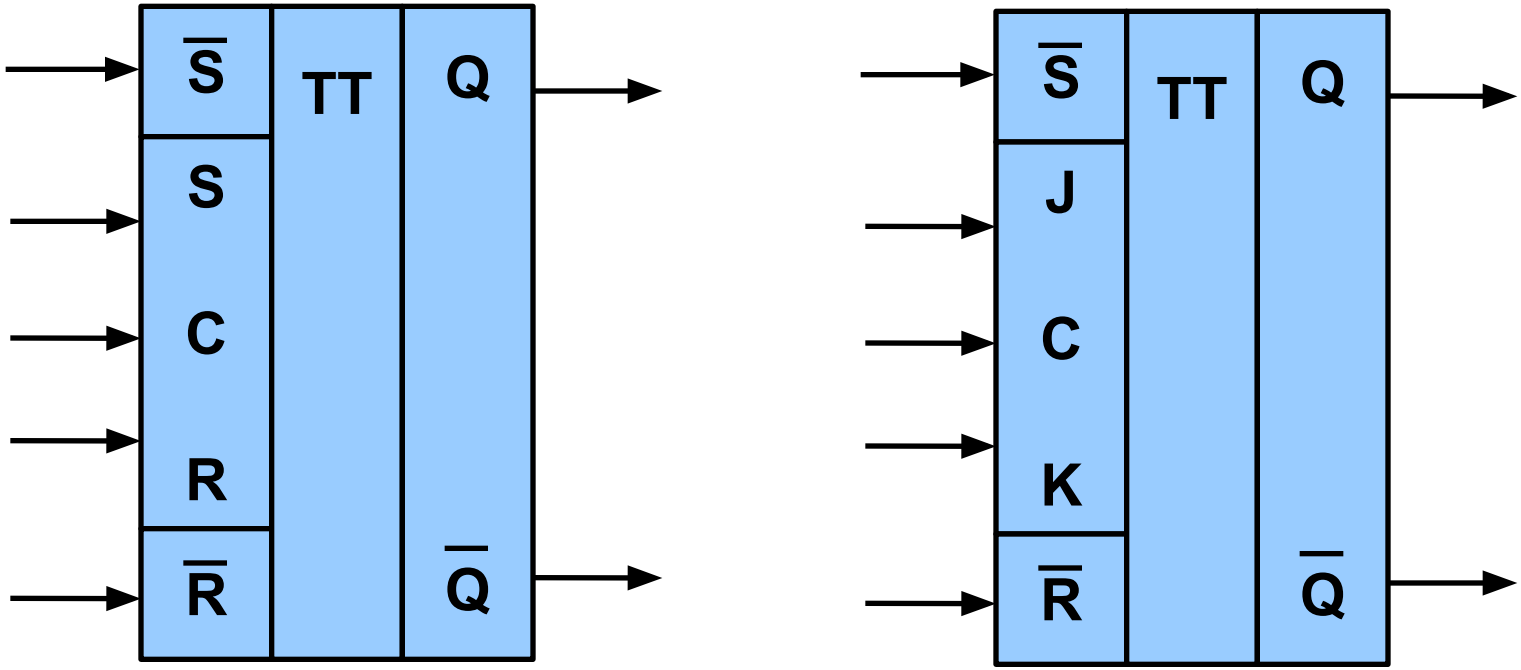
RS-триггер с сигналами обратной связи



Асинхронные сигналы

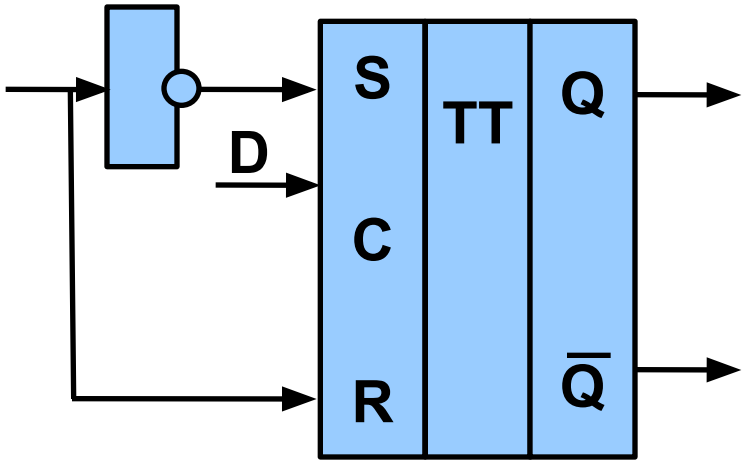


Условные обозначения



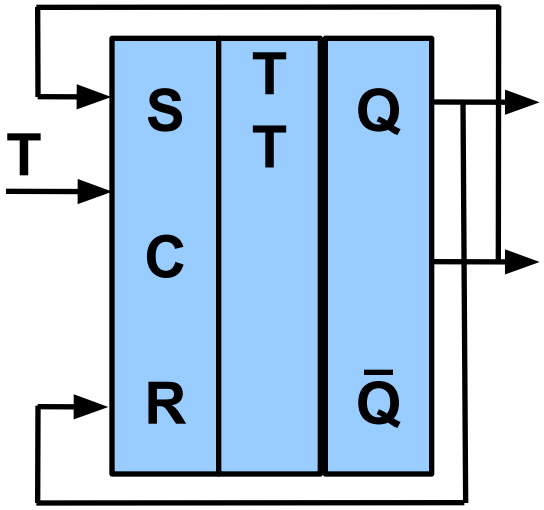
TT – ДПП (Двухполупериодный)
R, S – RS-триггер
C – синхронный
S, R – произвольная установка 0 и 1
J, K – JK триггер

Подача сигнала через инвертор

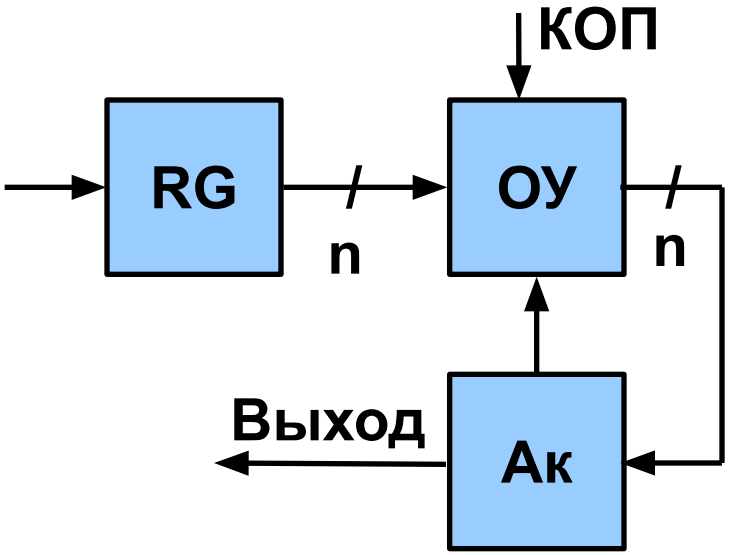


D-триггер

T-триггер со счетным входом



Организация работы операционного устройства



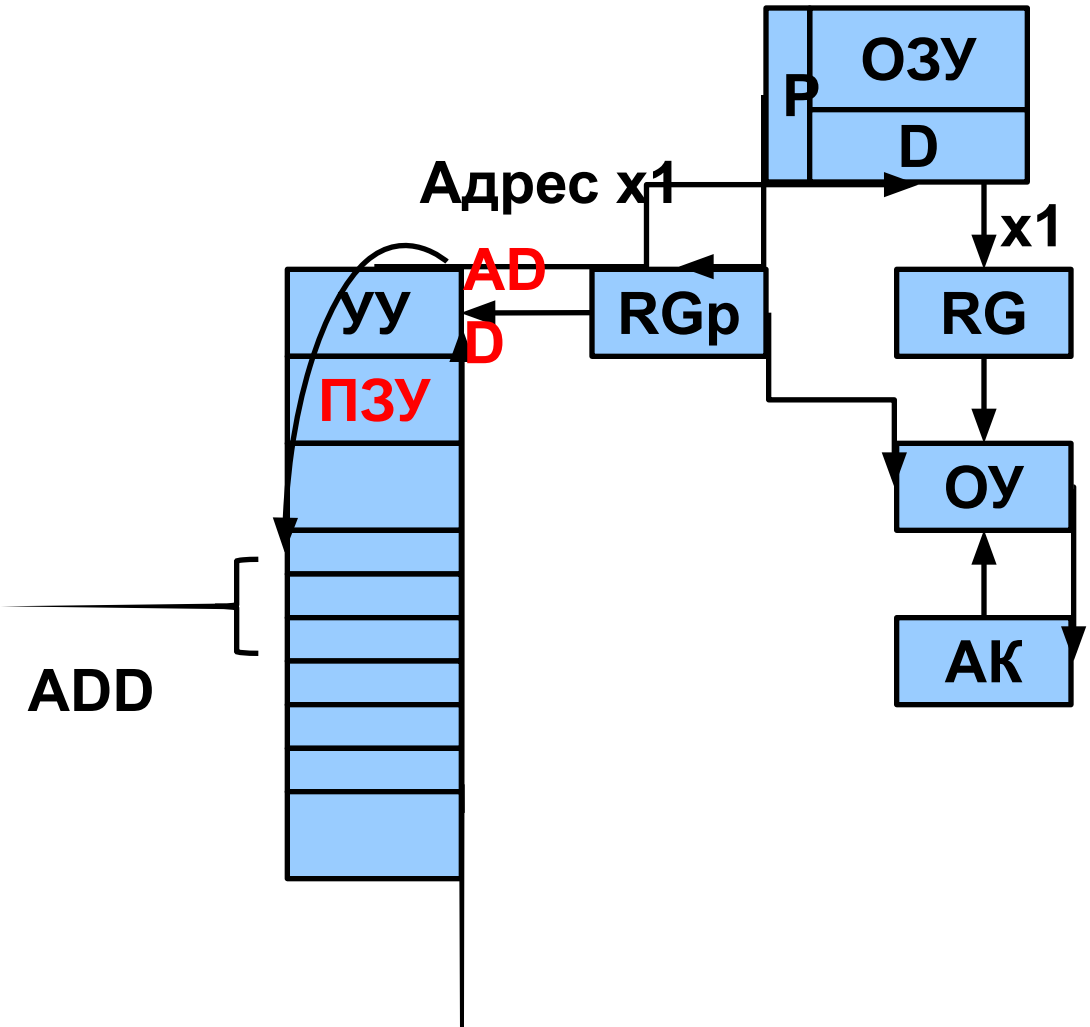
Команда ADD

Последовательность действий

1. Вызов команды
2. Вызов x1 (RG = [Ax1])
3. Ак = Ак + RG
4. Вызов x2 (RG = [Ax2])
5. Ак = Ак + RG
6. [Asum] = Ак

>= 6 тактов

Схема работы

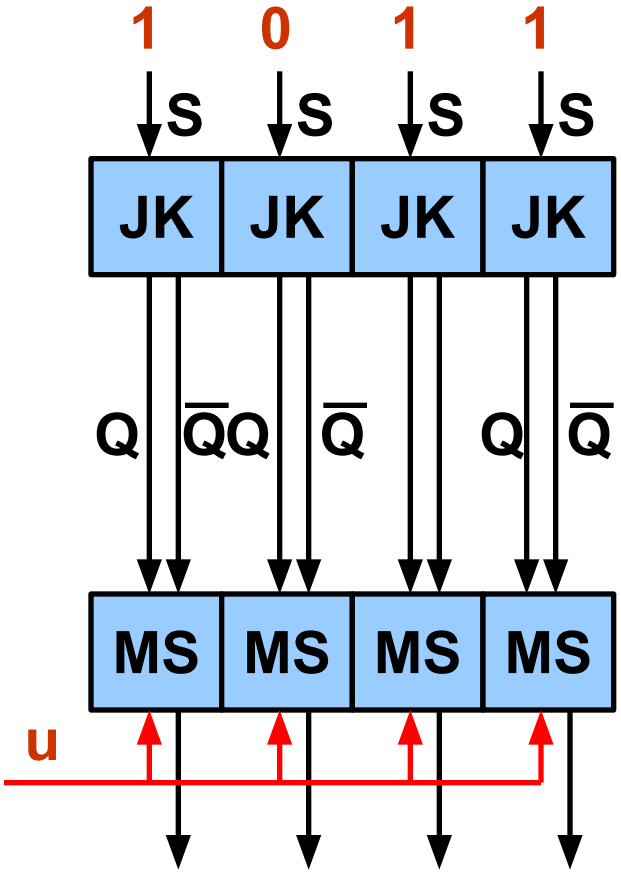


Центральные понятия ВТ

- Имя программы – адрес хранения первой команды программы в ОЗУ.
- Команда (код операции) – начальный адрес микропрограммы исполнения команды
- Несовместимость процессоров заключается в различных кодах операции.
- Из приведенной схемы видно, что общий принцип работы машины таков:

  “Делай то, что находится по этому адресу над тем, что находится по другому адресу”

Временное хранение и преобразование информации



Коммутация

Коммутация на примере 32-разрядного процессора

- Для сумматора требуются Ак, КОП, A1, A2, A3, D1, D2, D3.
- 8*32 = 256, 256 * 1,25 мм = 320
- 320 + расстояние между проводниками => ~620мм в периметре.
- Чем разветвленнее системы внутренних и внешних коммутаци процессора, тем эффективнее загружено ОУ.

Учебный курс

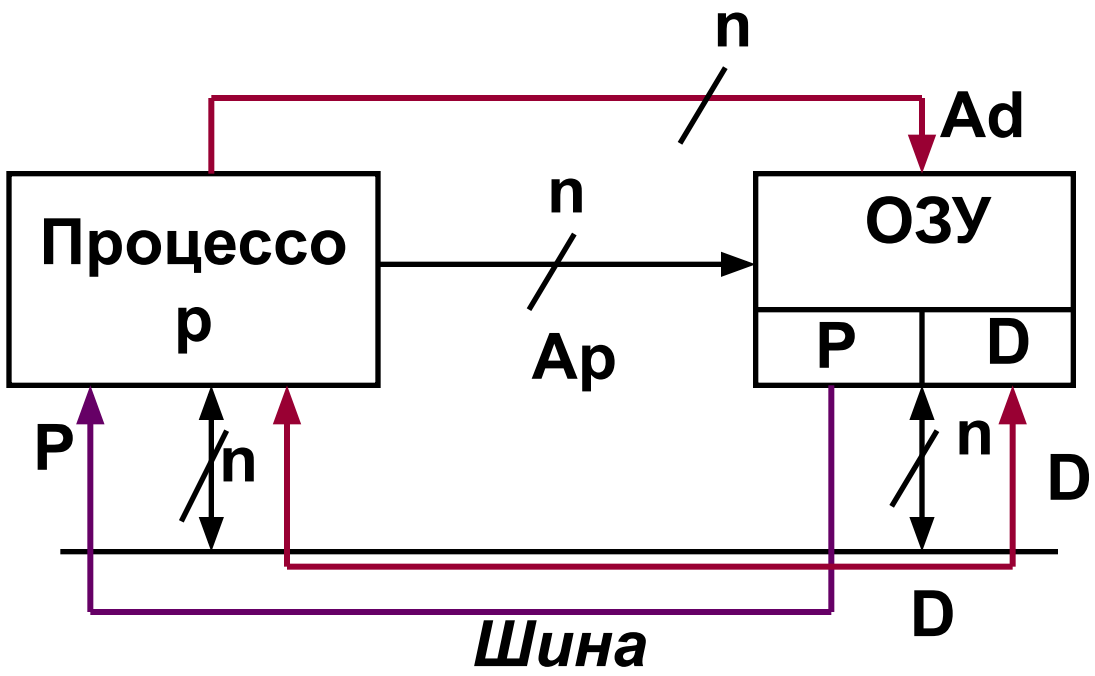
Принципы построения и функционирования ЭВМ

Лекция 7

Элементы и узлы вычислительной техники

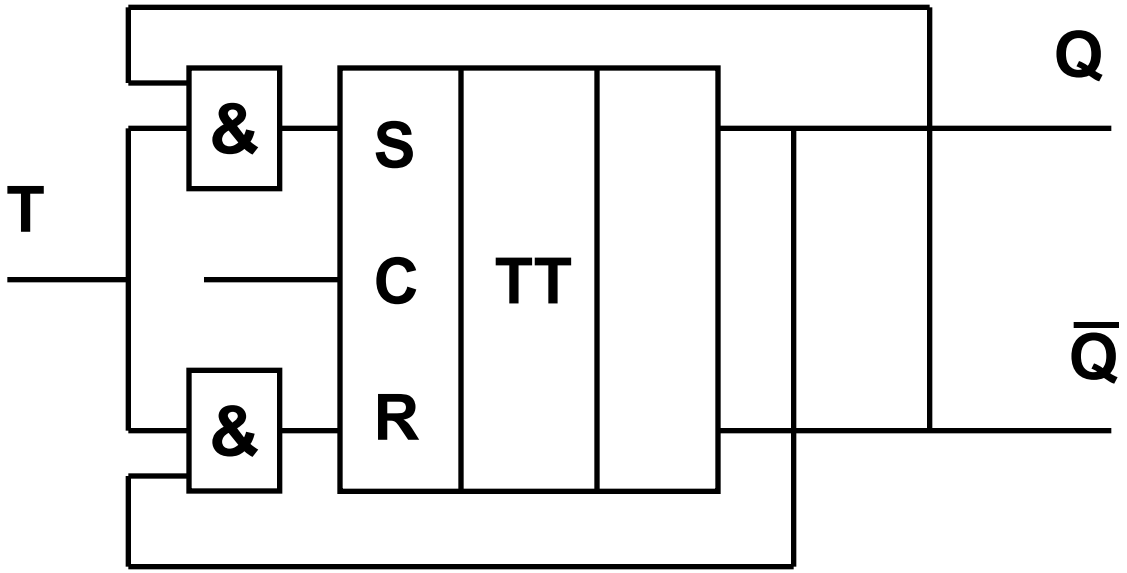
профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Общая схема работы машины



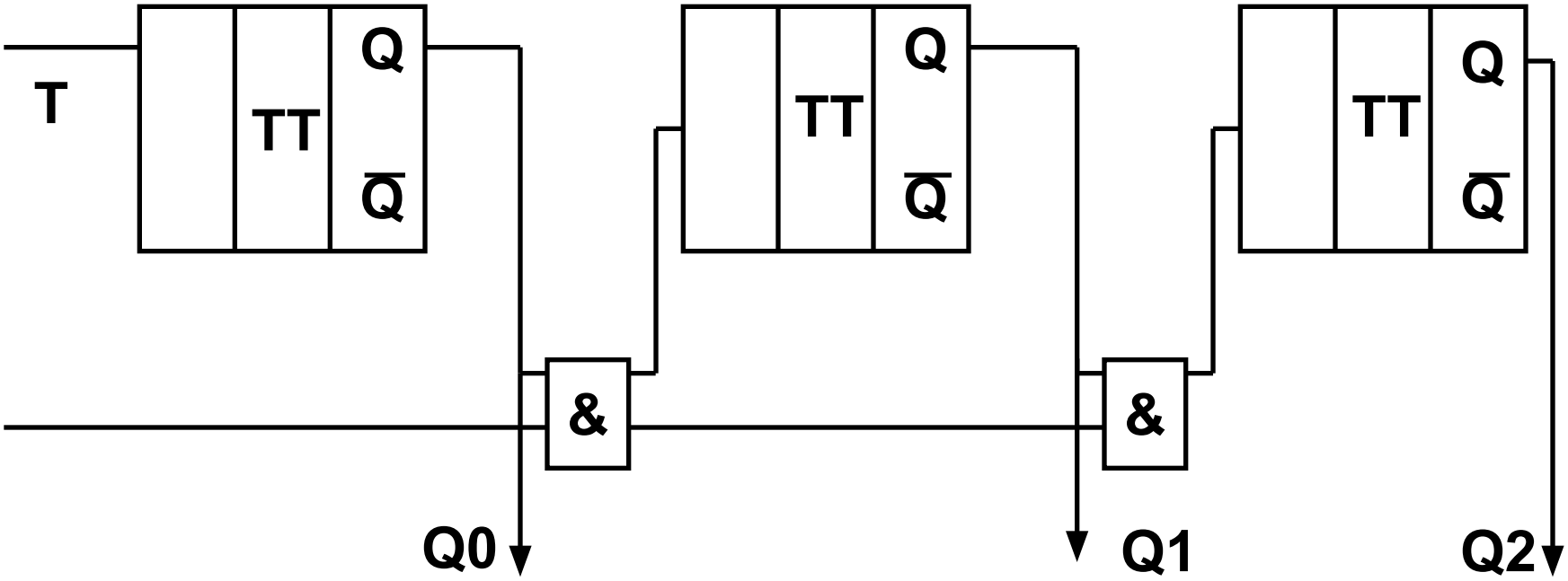
- Основное правило: делай то, что находится по этому адресу над тем, что находится по этому адресу.

Т-триггер



- Счетчики подсчитывают количество единиц, которые поступили на какой-то вход.

Трехразрядный счетчик

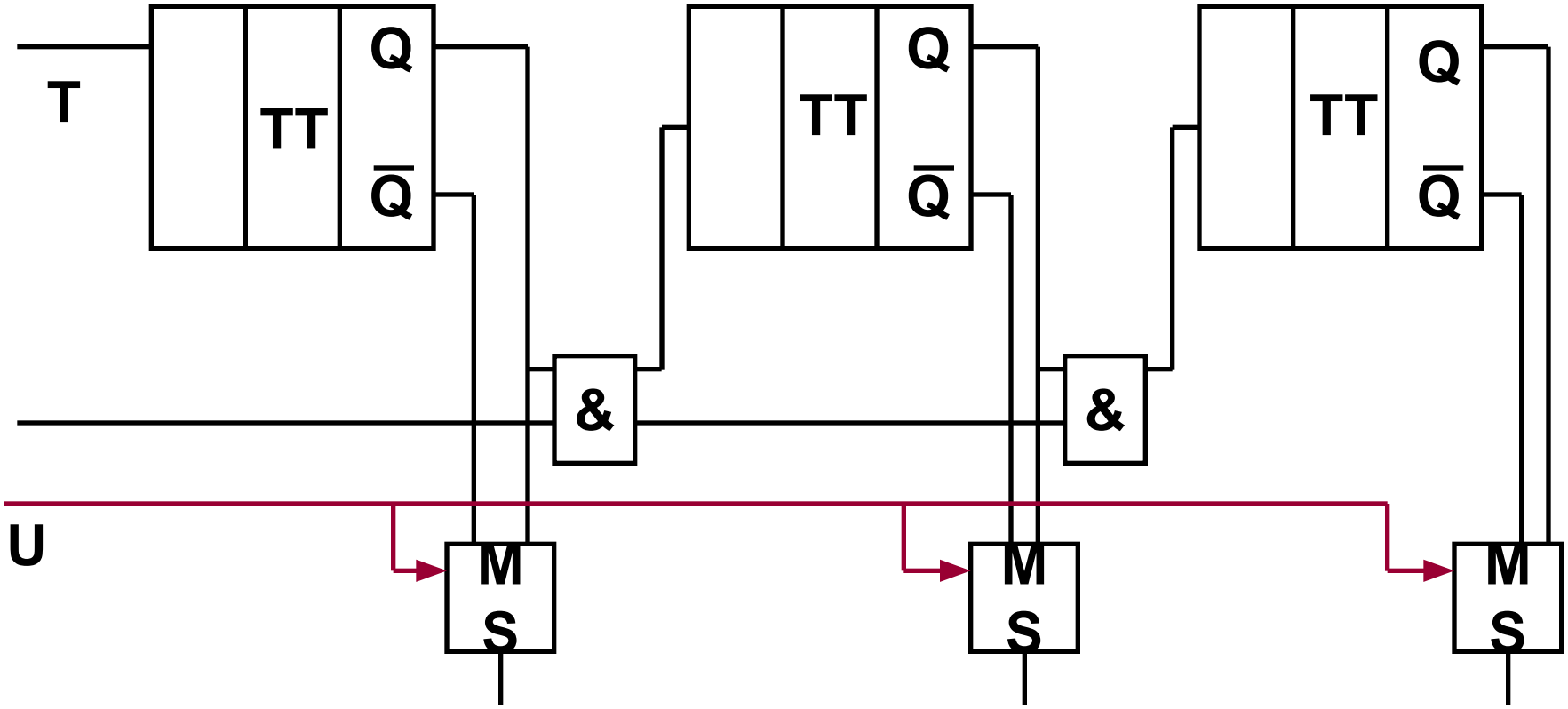


Трехразрядный счетчик

Q2	Q1	Q0	
0	0	0	+1
0	0	1	+1
0	1	0	+1
0	1	1	+1
1	0	0	+1
1	0	1	+1
1	1	0	+1
1	1	1	+1

- Трехразрядный счетчик имеет 8 состояний. Считает от 0 до 7.
- Закон адресации – линейный инкрементный.
- При снятии сигнала с инверсного выхода порядок адресации меняется на декрементный.

Трехразрядный счетчик

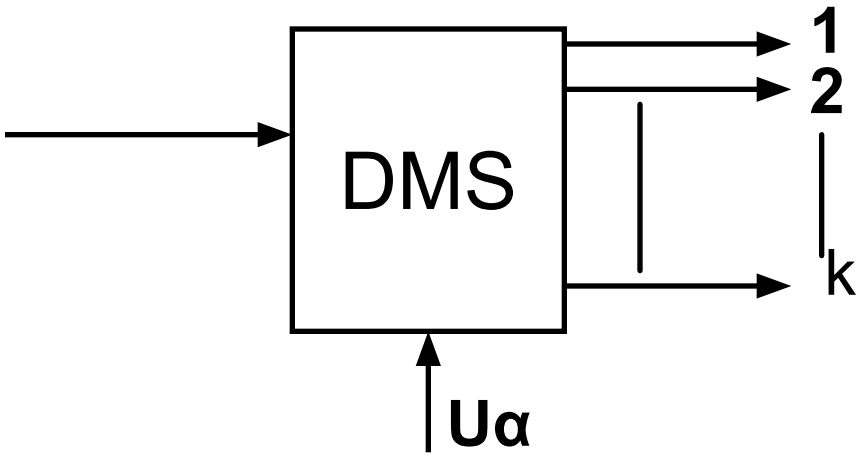


- Инкремент, если U = 1.
- Декремент, если U = 0.

Логическая адресация

- Рассмотренный линейный закон адресации предусматривает перемещение по памяти с шагом 1.
- Для увеличения шага на 2 (или на 4) необходимо подавать входной сигнал на 2 (соответственно, 4) входы счетчика.
- Вывод: **Закон адресации можно модифицировать изменяя вход счетчика.**

Демультиплексор

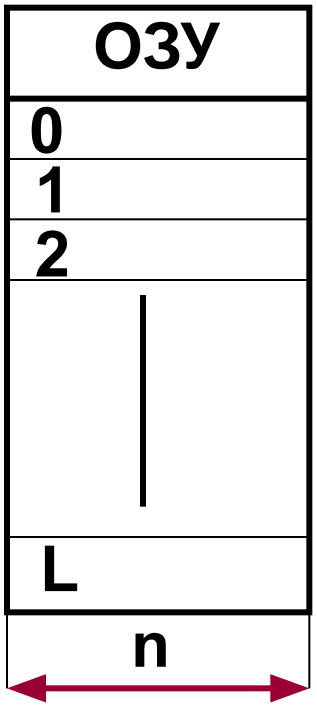


Демультиплексор используется для коммутации входного сигнала на несколько выходов.

$$\alpha =]\log_2 k[$$

Физическая адресация

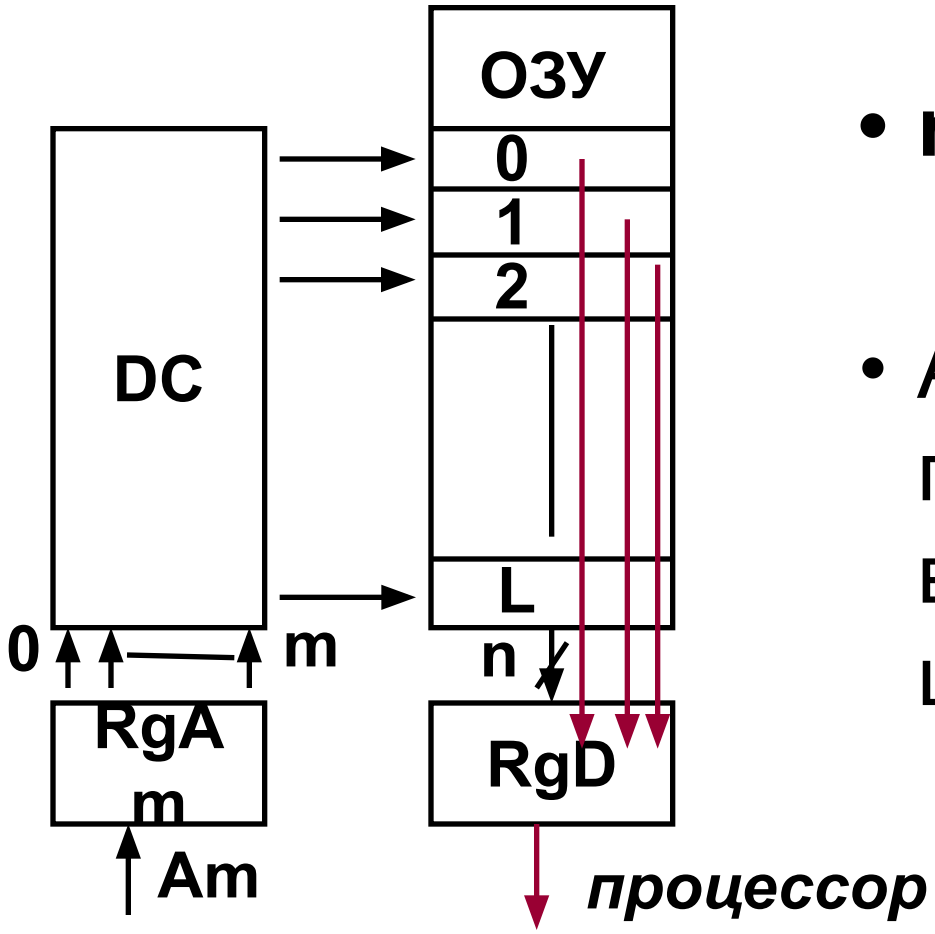
- Любая память представляет собой линейную совокупность ячеек



L – глубина памяти

n - ширина выборки

Порядок работы ОП

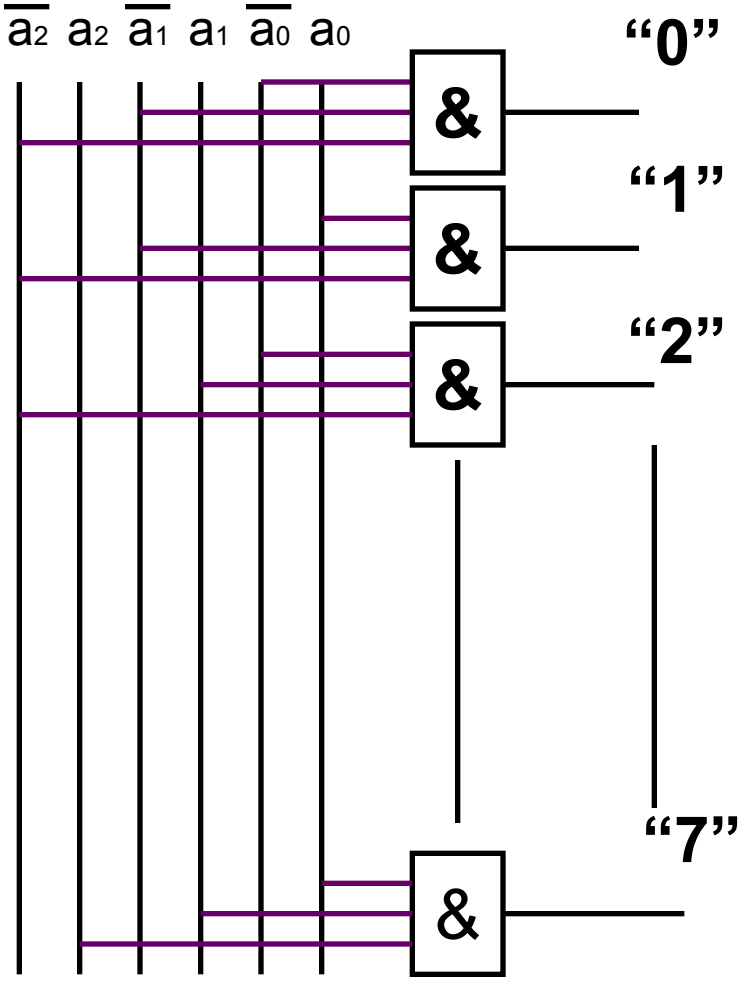


- $n =]\log_2 k[$
- Адрес всегда представляется в машине в виде целого без знака

Общее правило

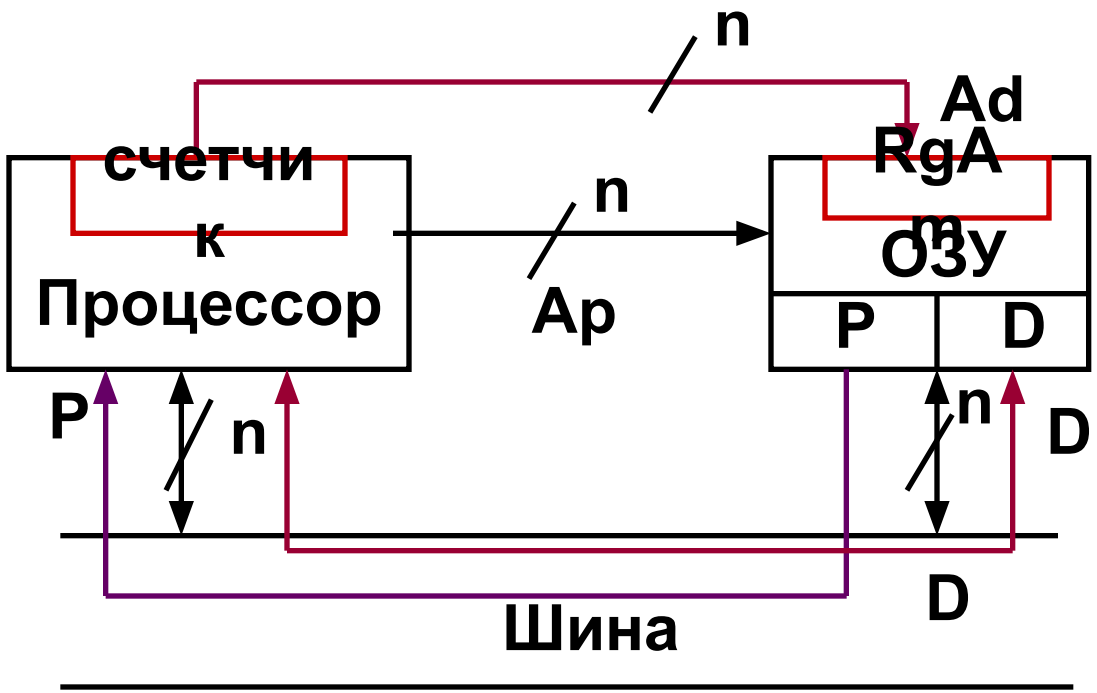
- Любая команда как процессора, так и обращения к памяти, начинается и заканчивается в регистре.
- Следствие: **любое преобразование данных совмещено по времени и пространству с пересылкой данных.**
- Дешифратор (в данном случае DC) преобразует логический адрес Am в физический **L**, которое отличается единичным значением на физическом выходе DC, который соответствует адресу по правилу $\mathbf{2^L}$.

Дешифратор



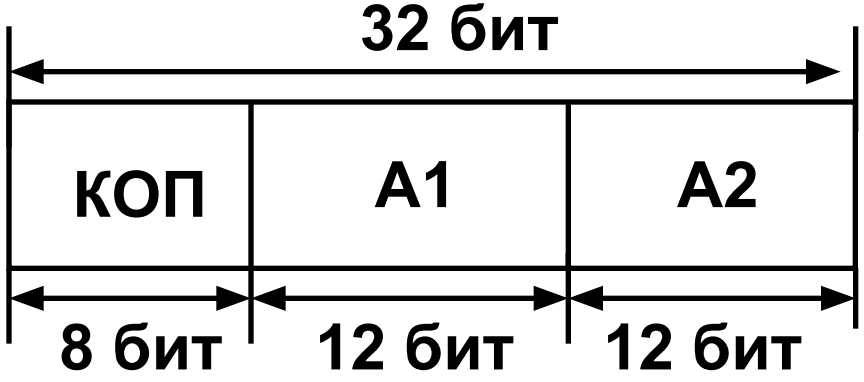
Порядок работы ОП

- Процессор имеет доступ только к регистрам памяти (адресный регистр и регистр данных).



Коды команд и операций

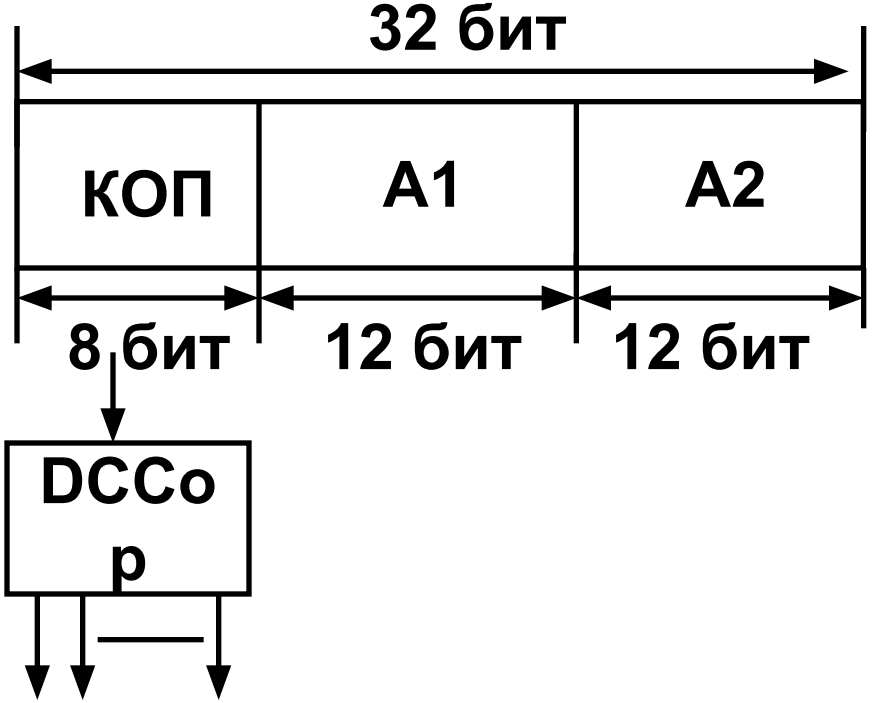
- В простейшем случае формат команды имеет вид:



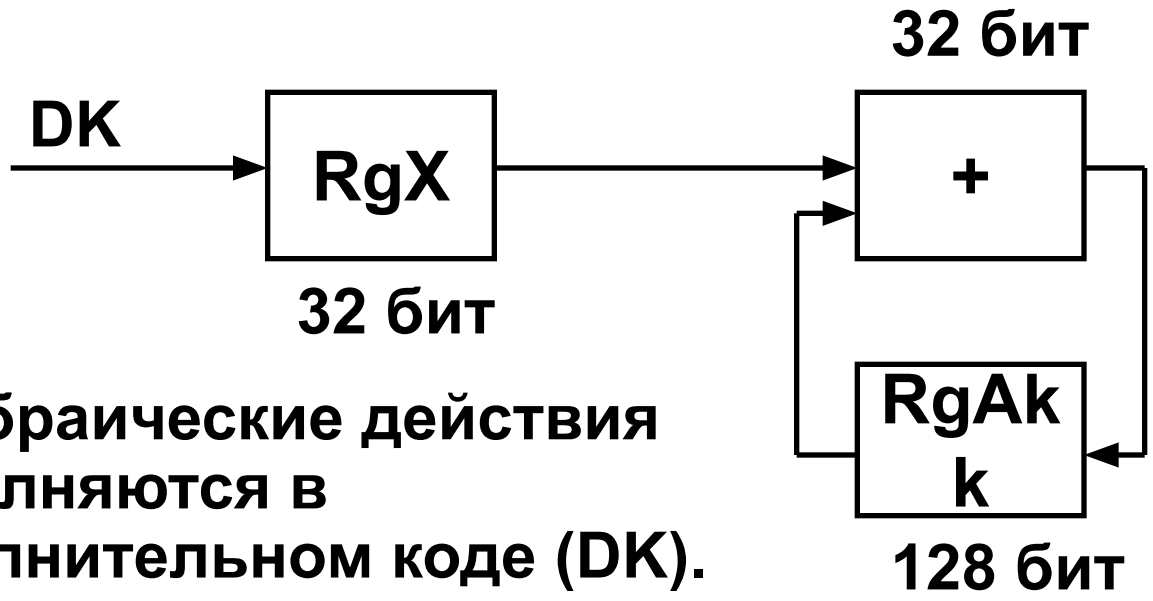
- A1 – адрес источника A2 – адрес источника.
- В таком формате команды первый операнд вызывается по адресу A1, второй – по адресу A2, а третий – результат – засылается по адресу приемника A1.

Коды команд и операций

- DCCop – дешифратор кода операций. Коммутирует входной сигнал на соответствующий выход.



Операционное устройство



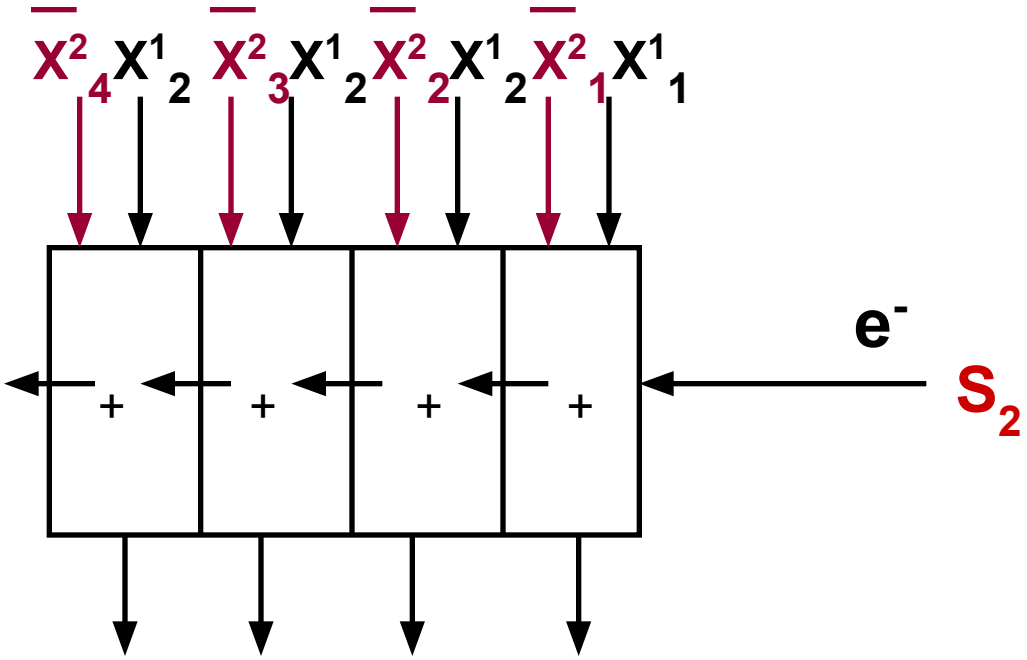
- **Алгебраические действия выполняются в дополнительном коде (DK). Результат также хранится в ОЗУ в дополнительном коде (DK).**

Хдоп = Хпрям, если s = «0»;

Хдоп = $\overline{\text{Хпрям}}$ + 1, если знак «-».

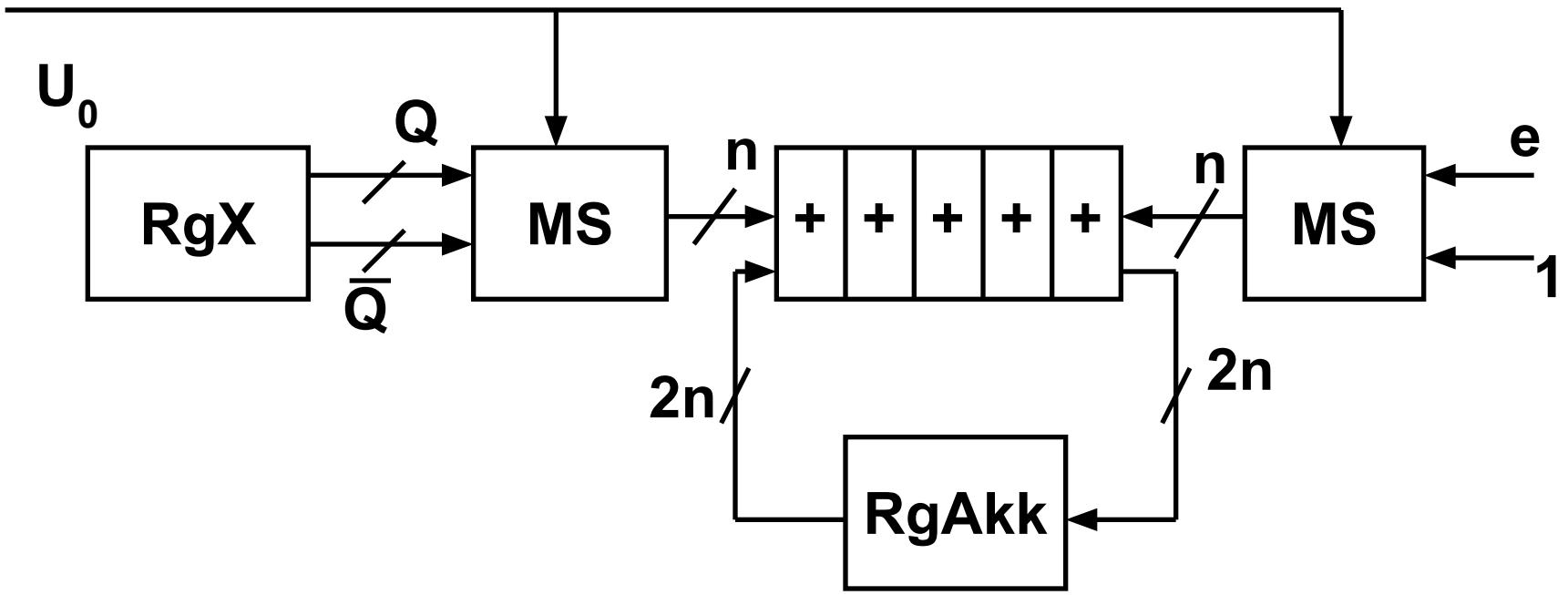
Сумматор

- $(X^1_n) – (X^2_n)$



- Чтобы сумматор превратился в «вычитатель», необходимо взять сигнал с выхода **Q** и в младший разряд добавить $\mathbf{e}^-$, равное **1**.

Общая схема



- U0 = 1, если ADD
- U0 = 0, если SUB

Итог

С помощью переключательной функции и вентилей можно покрыть все функции, которые выполняет машина.

Учебный курс

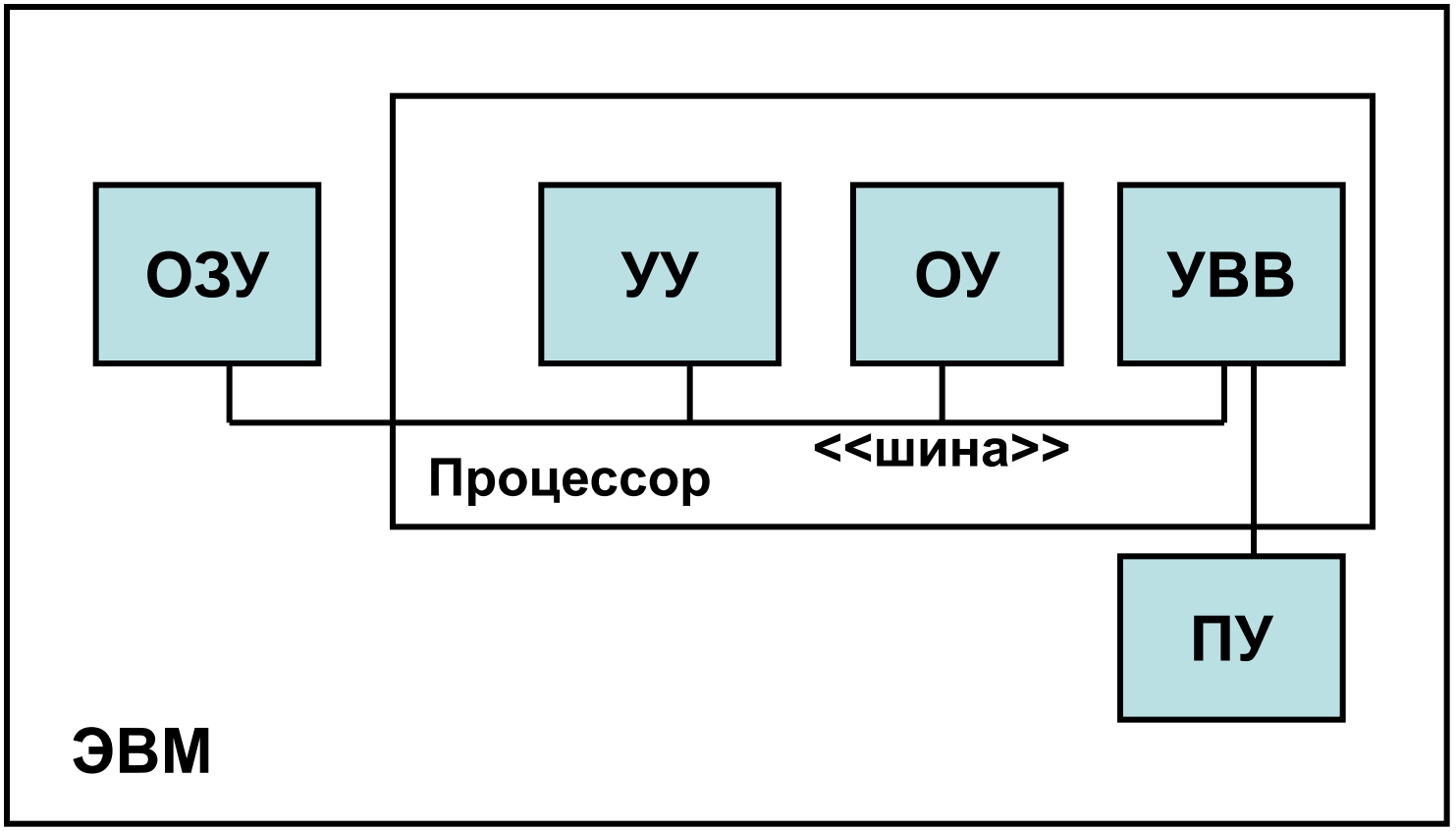
Принципы построения и функционирования ЭВМ

Лекция 8

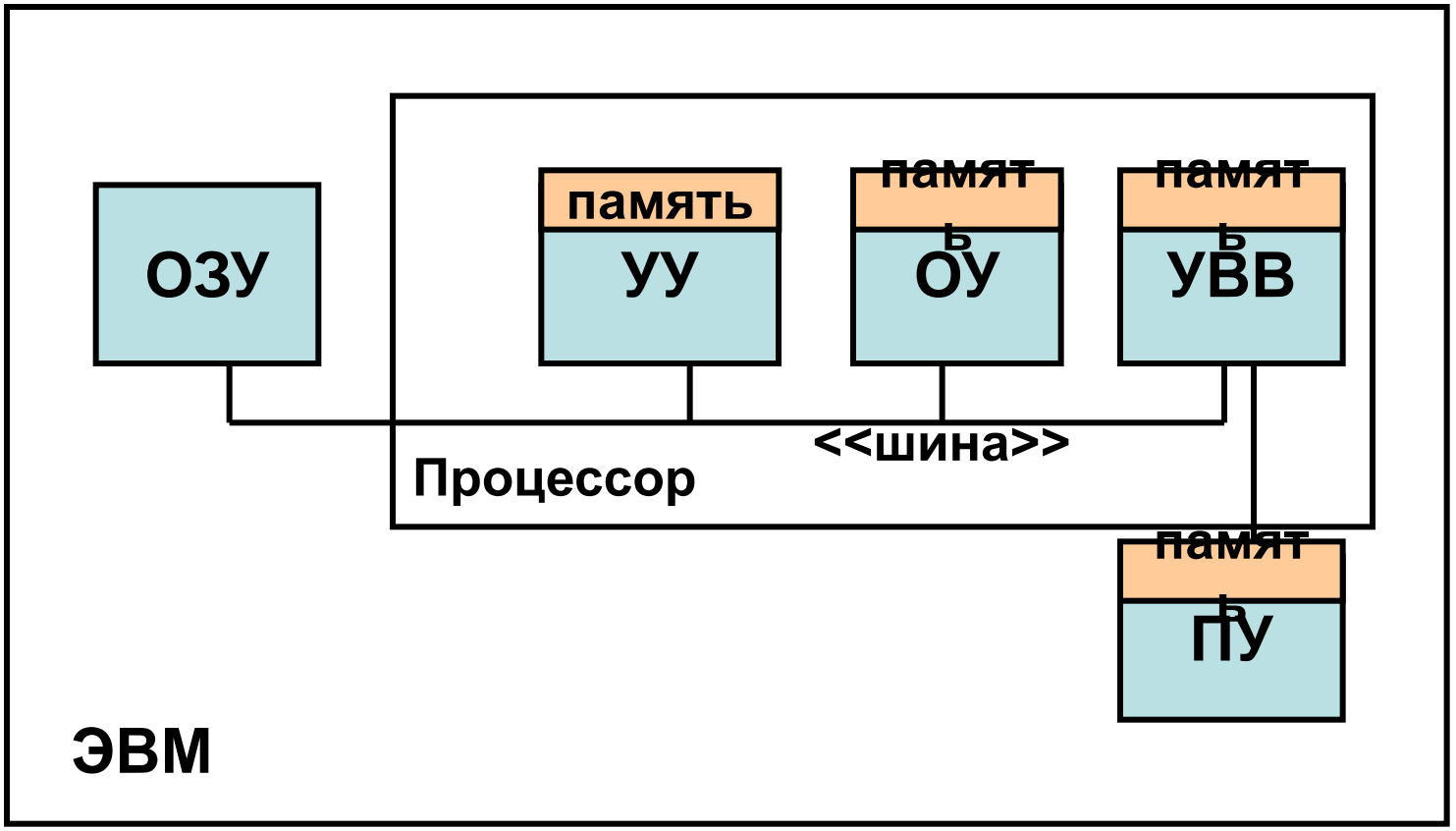
Организация памяти в ЭВМ

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Организация памяти в ЭВМ



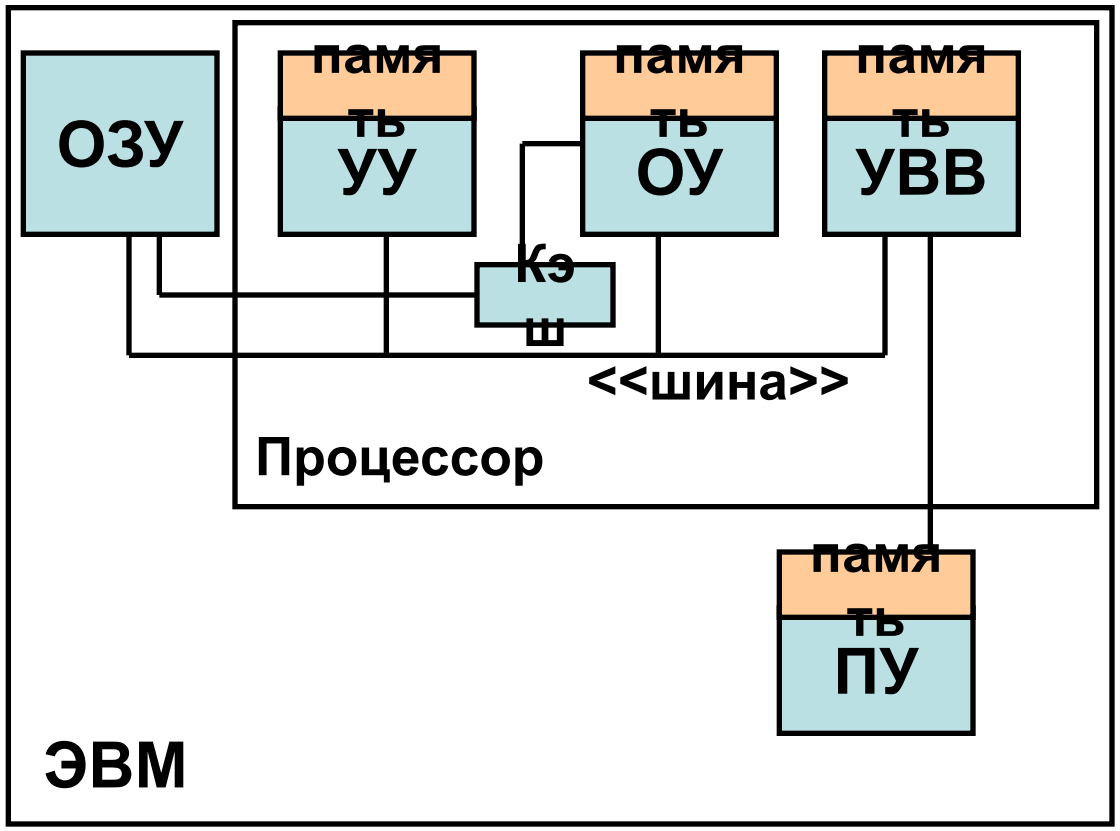
Память в элементах ЭВМ

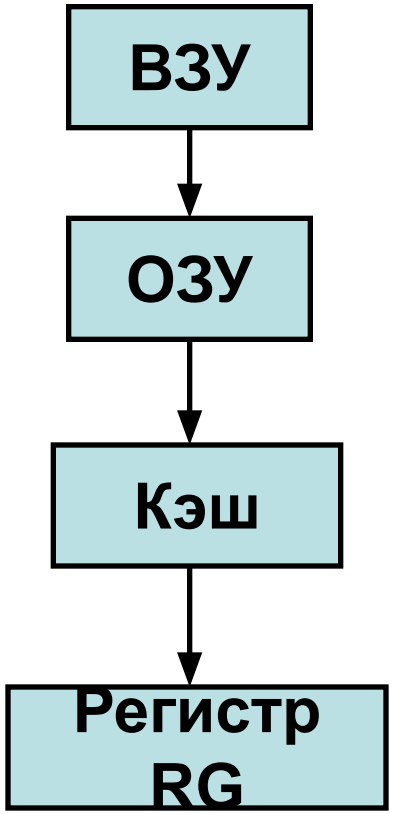


Классификация памяти по признакам

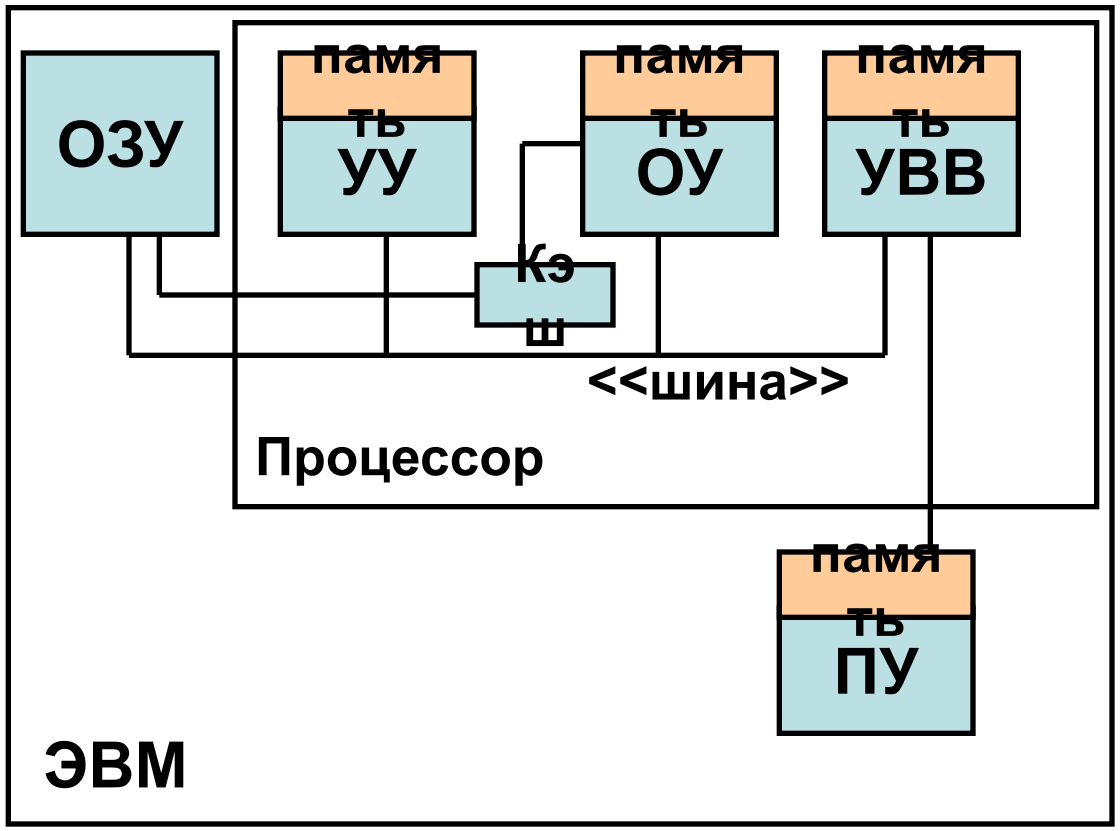
- внешняя (оптическая память, магнитная память) и внутренняя
- оперативная и буферная (кэш)
- регистровая

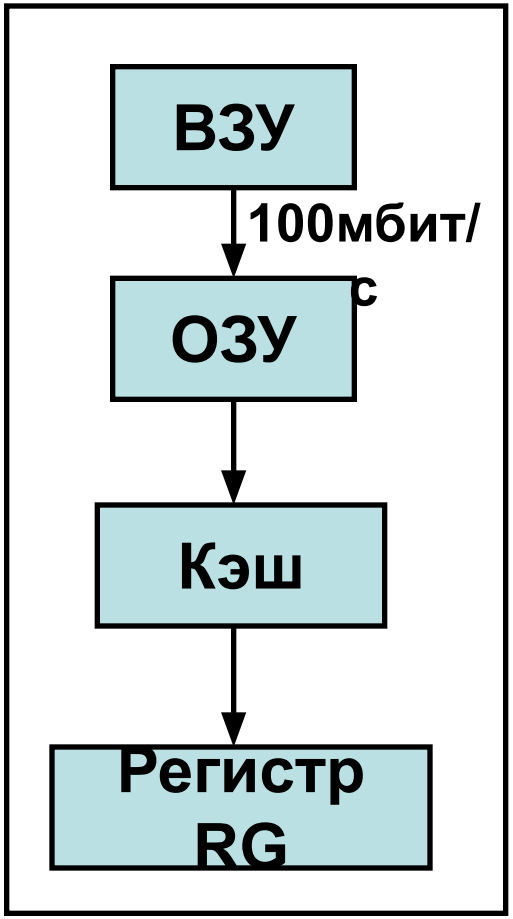
Иерархия в организации памяти





Иерархия в организации памяти





Физико-технические процессы

- электронные переключательные процессы
- магнитные, электромагнитные (фиритные ячейки памяти)
- оптические

Организация доступа к элементам памяти

- параллельный (прямой) доступ

  время доступа = 2*T
- последовательный доступ

  время доступа = (L/2)*T

  L – длина последовательной памяти

Классификация памяти по типу записи

- постоянные запоминающие устройства (ПЗУ, ROM)
- память чтения-записи (RAM)

Классификация памяти по продолжительности хранения

- статическая память
- динамическая память (DRAM)
  (хранение осуществляется на паразитных плоскостях pn-переходов)

Чем характеризуется любая память?

- глубиной и шириной «выборки»
- продолжительностью цикла обращения (быстродействием)
- надежностью

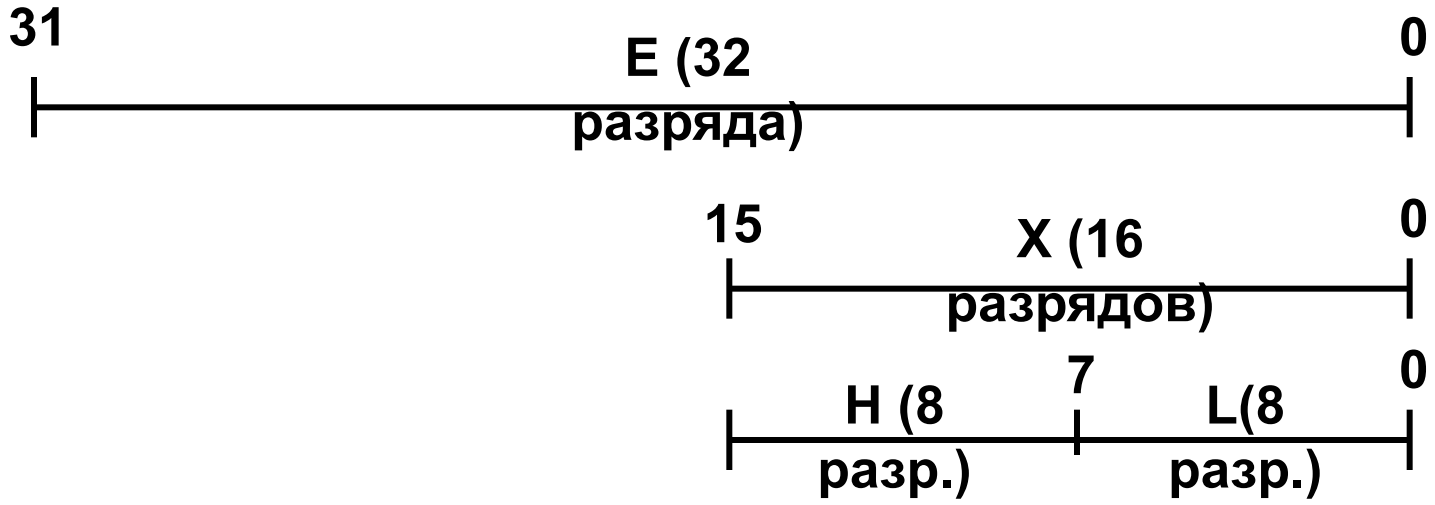
Единица информации

- **Байт (8 бит)**
  - слово 16 бит
  - двойное слово 32 бита
  - квадрослово 64 бита

- **Булева переменная**
  Доступа к отдельным битам нет
  «0» - 00000000
  «1» - 11111111

Организация регистровой памяти

Индексация:



Типы регистров

Регистры:

EAX – аккумулятор (результат)

EDX – регистр данных

EBX – регистр базы

ECX – регистр счета

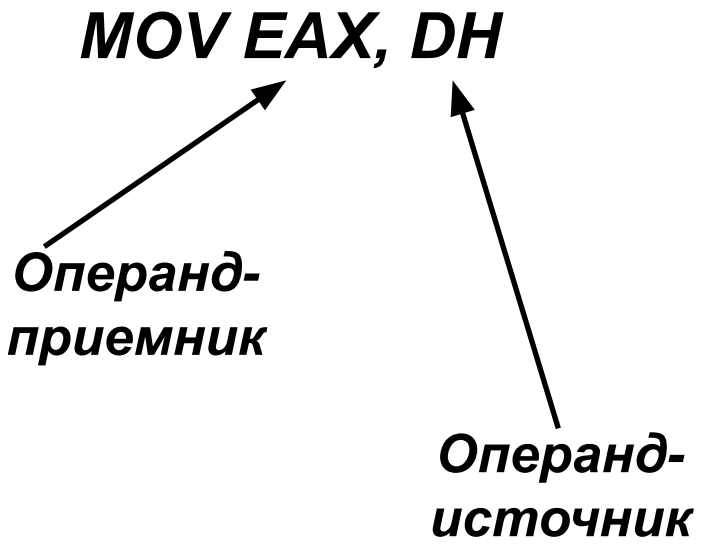
ESP – указатель стека (только 16 разр. формат)

EBP – указатель базы (только 16 разр. формат)

ESI – индекс источника

EDI – индекс приемника

Примеры команд (операций):



Учебный курс

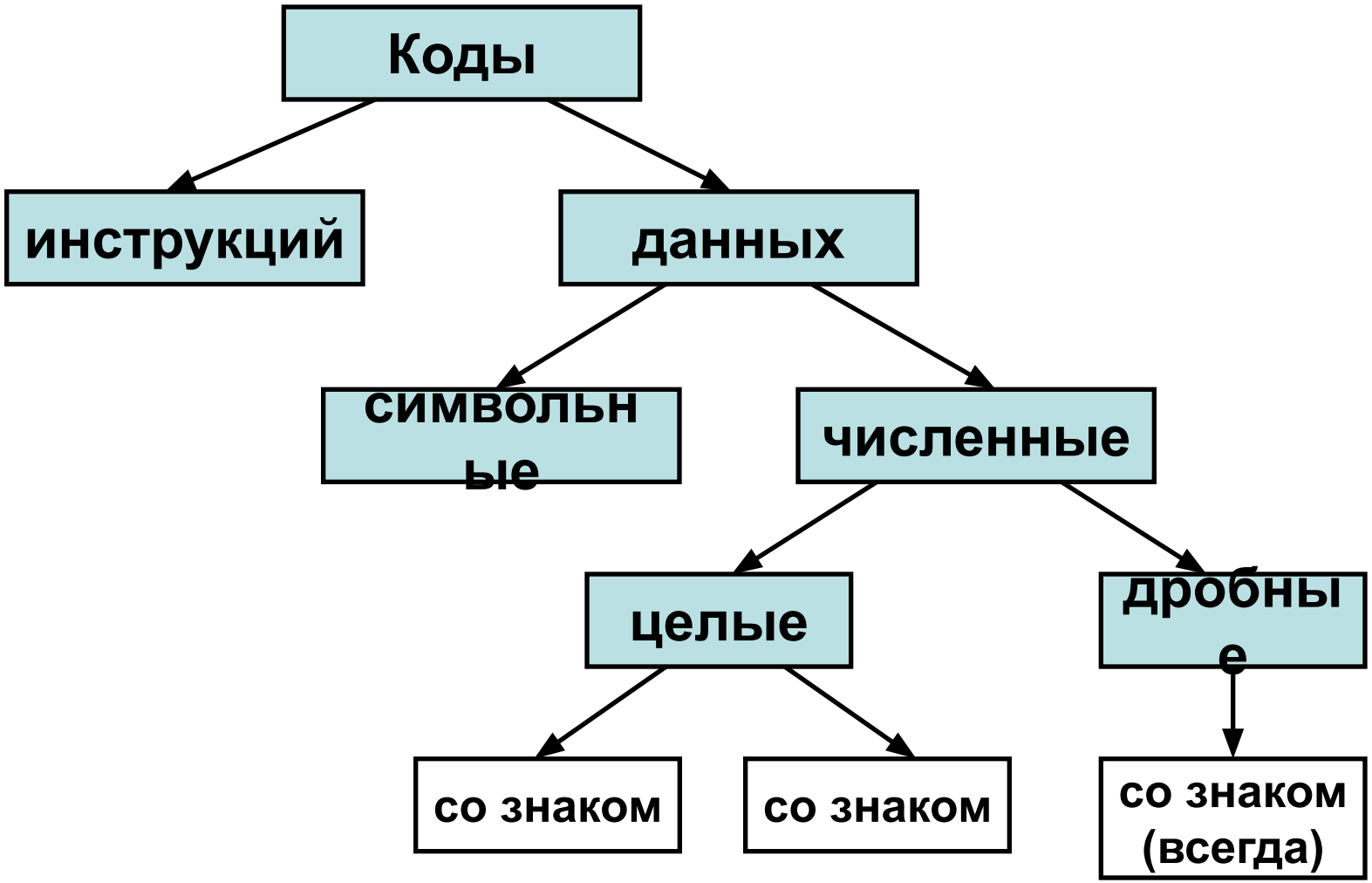
Принципы построения и функционирования ЭВМ

Лекция 9

Методы адресации. Способы представления информации в ЭВМ.

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Представление информации в ЭВМ



116

Представление числа в двоичном коде

$$A = \sum_{i=1}^{n} x_i w_i$$

w_i – вес двоичного разряда

$x_i \in \{0,1\}$

Если число целое: $w_i = 2^{i-1}$

Если число дробное: $w_i = 2^{-i}$

Прямой код

Прямой код – *zm*

z – знаковый разряд

m – мантисса

$A \geq 0$: z = “0”

$A < 0$: z = “1”

118

Дополнительный код

- Все арифметические действия выполняются в дополнительном коде

$$A_{доп} = \begin{cases} A^{пр},\ z=0 \\ \overline{A^{пр}} + 1,\ z=1 \end{cases}$$

119

Цена перехода от десятичного кода к двоичному

При переводе целого числа мы делим до получения результата

При переводе дробного числа мы умножаем *n* раз, где *n* – заранее заданное число

120

Пример перевода

Число – 0,37

$w_1 = 2^{-1} = 0,5$

$w_2 = 2^{-2} = 0,25$

$w_3 = 2^{-3} = 0,125$

$w_4 = 2^{-4} = 0,0625$ ограничение 4 разрядами

$w_5 = 2^{-5} = 0,03125$

Итог: 0,3125

Абсолютная погрешность: 0,0575

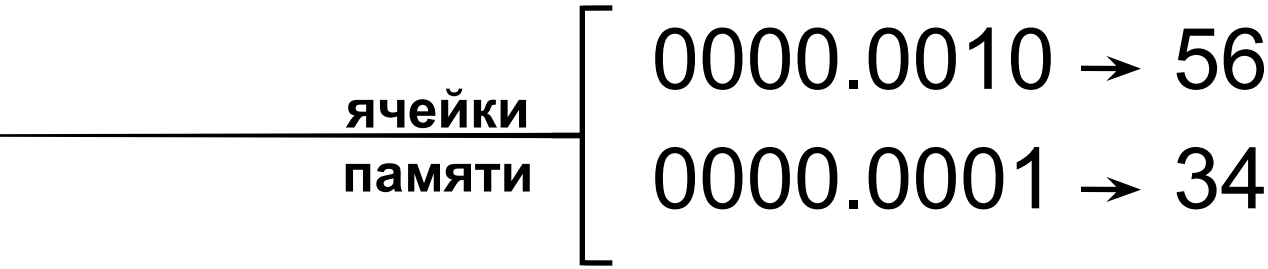
121

Представление данных в памяти

122

Метод записи в память

3456 – целое число



По младшему адресу - старший байт

По старшему адресу – младший байт

Адресация данных в ЭВМ

Основные способы адресации:

- **линейная**
- **сегментная**
- **страничная**
- **смешанная**

124

Сегментация

Сегментация используется для системной организации памяти

Сегментация позволяет:

1) минимизировать паразитные пересылки из ОЗУ во внешнюю память
2) улучшить защиту памяти в многозадачном режиме
3) повысить отказоустойчивость

Основные сегменты

Выделяют пять основных сегментов:

- сегмент программ
- стек
- три сегмента пользователя данных

Техника адресации

Существует девять типов адресаций:

1) непосредственная
2) регистровая
3) прямая адресация к памяти
4) косвенная регистровая
5) относительная
6) прямая индексная
7) относительная индексная
8) масштабирование
9) поразрядная

127

Учебный курс

Принципы построения и функционирования ЭВМ

Лекция 10

Типы адресации. Стеки. Процессоры. ОЗУ.

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

1.Организация работы с памятью

- пропускная способность ЭВМ определяется скоростью обмена с памятью
- скорость обмена с внешними устройствами

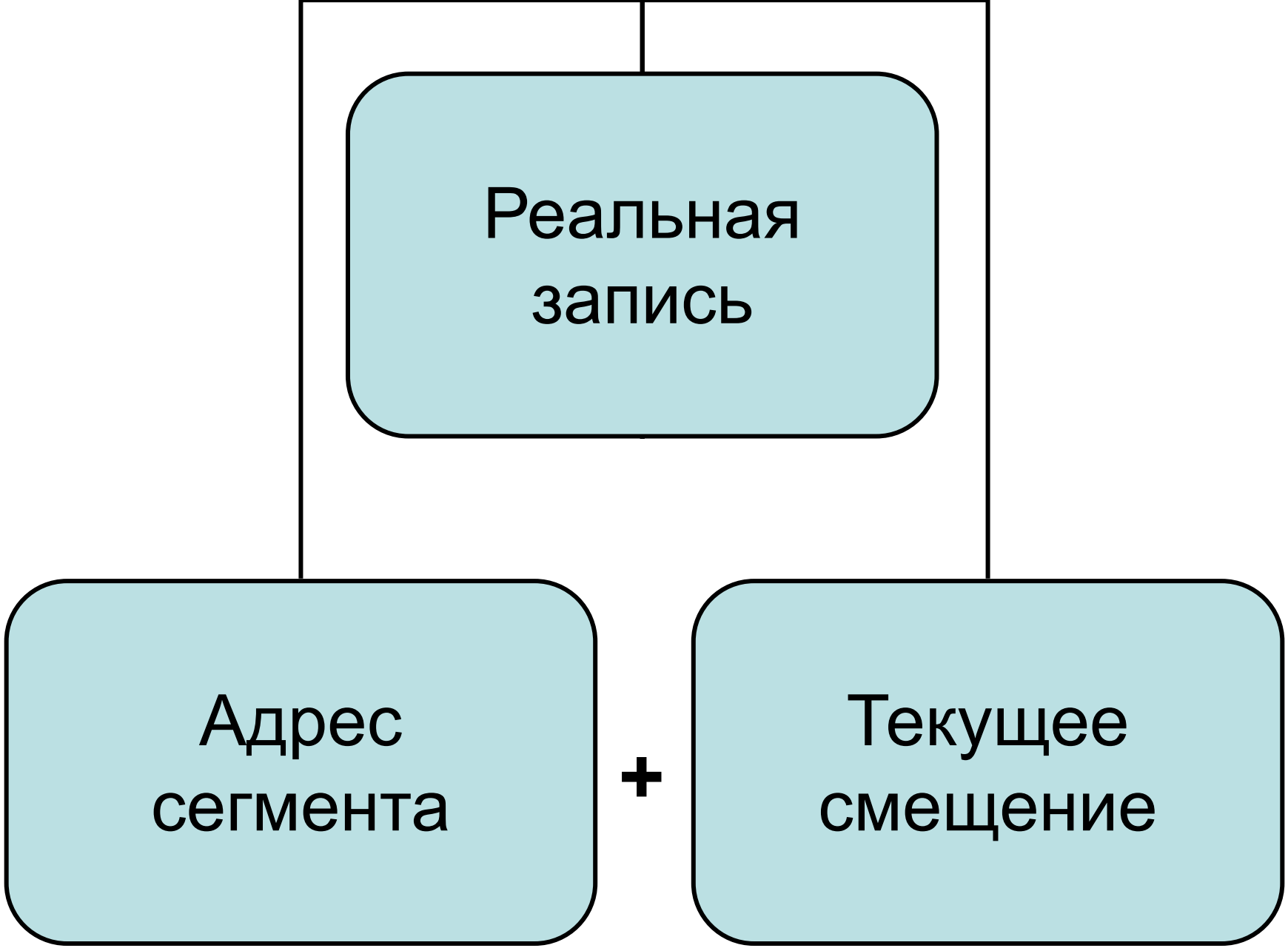
Организация работы с памятью определяет эффективность использования процессора

1.1 Логическая и физическая организация памяти

- на логическом уровне организация памяти структурируется сегментами
- на физическом – страницами

1.1.1 Сегментация памяти

- **повышение надежности**
- **динамическое управление памятью**


Реальная запись
Адрес сегмента
+
Текущее смещение

2. Непосредственная адресация

Пересылка команды и пересылка данных осуществляется одновременно, т.к. операнд находится в теле программы

+ снижение временных издержек на пересылки данных

- нужно перетранслировать программу

Непосредственная адресация удобна для хранения констант

3. Регистровая адресация

При регистровой адресации значение операнда-источника предварительно запоминается в одном из внутренних регистров процессора

Пример:

- *MOV_EBX, EDX* – из регистра данных в регистр базы (процессоры фирмы Intel)
- *MOV_EDX, BX* – в данном случае 2 варианта:

1) 16 нулей + заданное число
2) 16 единиц + заданное число , т.е. со знаковым расширением

1000 1110 1010 0011 →

1000 0000 0000 0000 0111 0001 0101 1101 – прямой код

При сохранении численных значений выполняется пересылка со знаковым расширением

Пример:

(-12) – прямой код

↓

1000 1100 – восьмиразрядный код

↓

1111 0100 – дополнительный код

↓

1111 1111 1111 0100 – шестнадцатиразрядный код

↓

1000 0000 0000 1100 – (-12)

! Все преобразования кодов выполняются во время пересылки данных.

4. Прямая адресация

Операндом является переменное имя или метка.

Пример:

***MOV_AX, mydata* – по имени *mydata* обращаемся к памяти →**

получаем адрес →

осуществляем пересылку данных

5. Косвенная регистровая адресация

В этом случае вместо метки используется значение операнда по адресу смещения, который хранится в одном из регистров.

***SI* – регистр индексов;**

***DI* – индекс приемника;**

***BX* – регистр базы;**

***BP* – указатель базы.**

<u>Пример:</u>

***MOV_BX, [DI]* (этой команде обычно предшествует команда *OFF SET*)**

- Данный способ адресации наиболее удобно использовать, когда данные хранятся в форме таблицы

- Доступ к отдельным значениям данных ускоряется за счет увеличения содержимого регистра базы

6. Относительная адресация

Действующий адрес получается суммированием смещения с содержимым регистра базы

Пример:

MOV_EDX, [EBX+4]

Данный способ используется при работе со списками

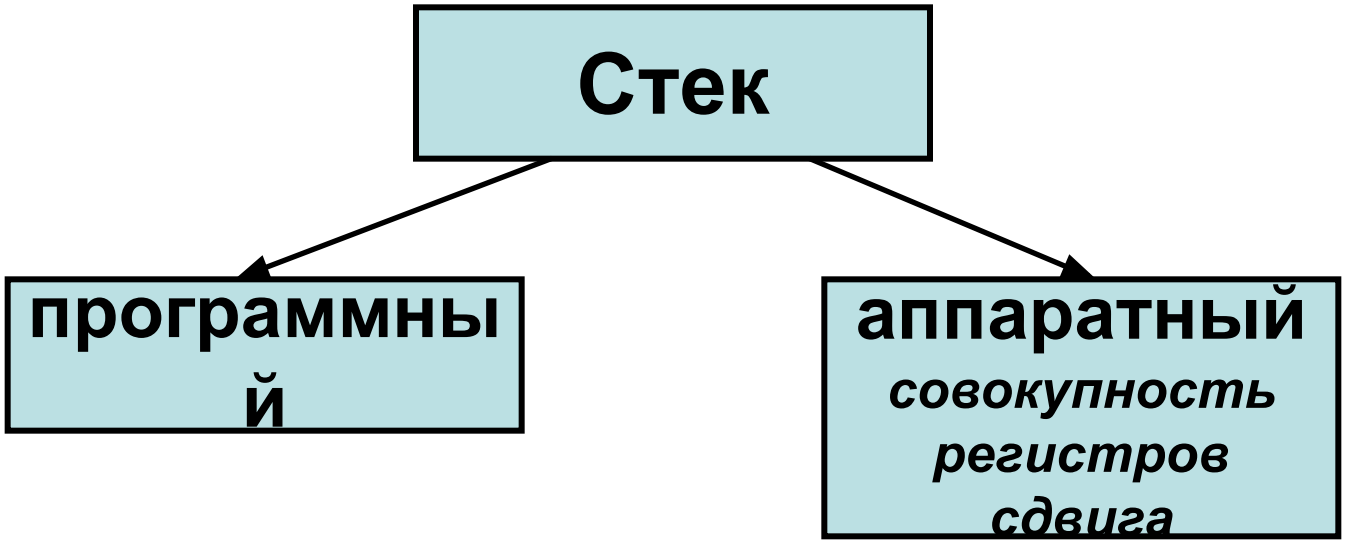
7. Прямая индексная адресация

- смещение операнда определяется суммой смещения и значения одного из индексных регистров

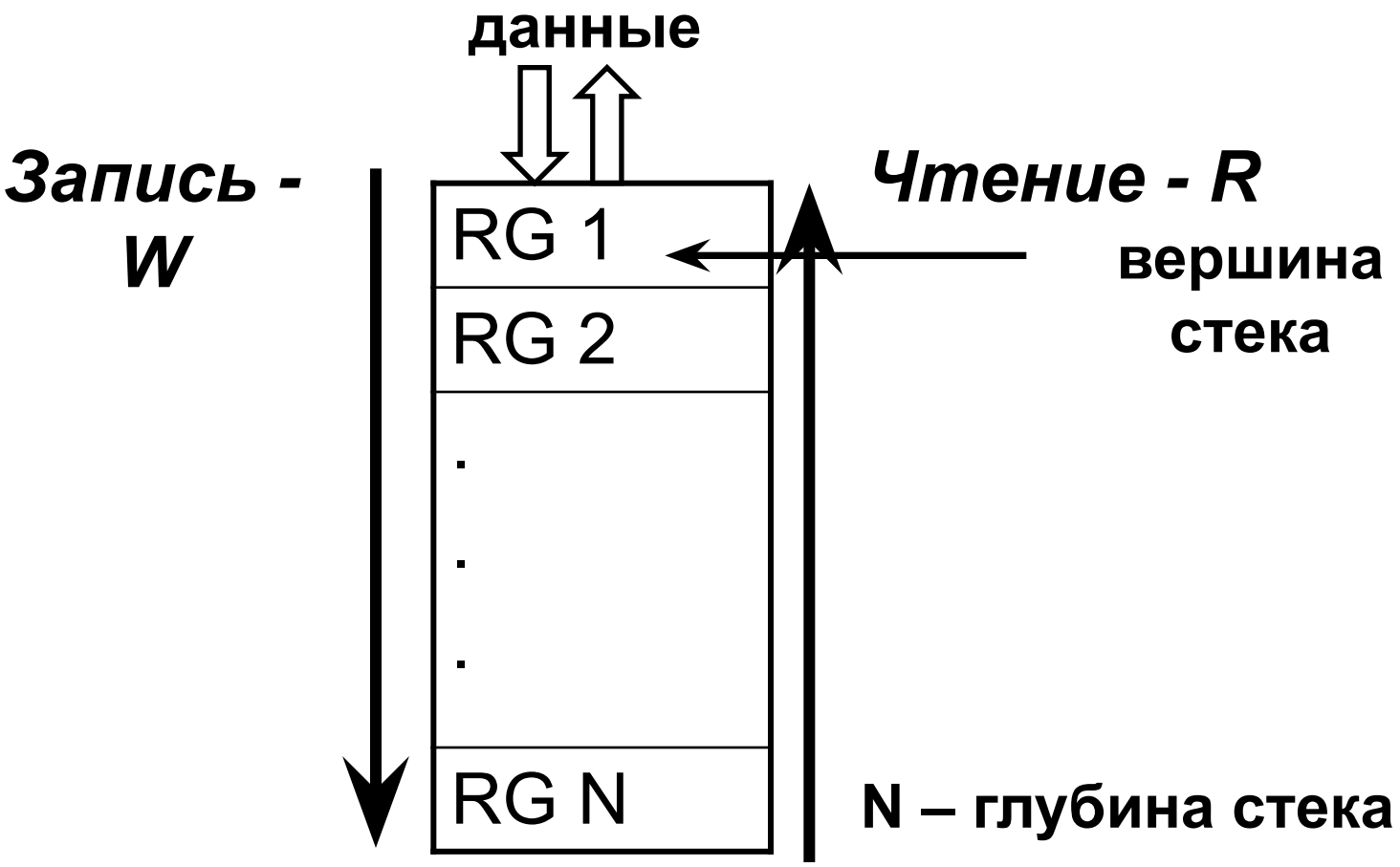
  E(SI), E(DI)
- индексная адресация удобна при векторной организации памяти (данные представляют собой статические вектора)

8. Стеки

Механизм стеков снижает издержки на адресацию



8.1 Аппаратный стек



! При стековой организации все данные идентифицируются вершиной стека

Учебный курс

Принципы построения и функционирования ЭВМ

Лекция 11

Микрокоманды и микрооперации

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Процессоры

Непосредственно преобразуют данные и управляют эти процессом

Процессор:

- дешифрует
- выполняет команды программы
- организует обращение к АЗУ
- инициирует работу периферийных устройств
- воспринимает и обрабатывает запросы, поступившие от устройств ЭВМ и внешней среды (запросы прерывания)

- **Выполнение команд ЭВМ строго регламентировано во времени: на каждом этапе выполняется одна или несколько микроопераций**
- **Конкретный состав микроопераций и последовательность их выполнения определяются системой команд ассемблера, логической структурой и особенностями работы каждого процессора**
- **Последовательность микроопераций, реализующих данную команду, называют микропрограммой**

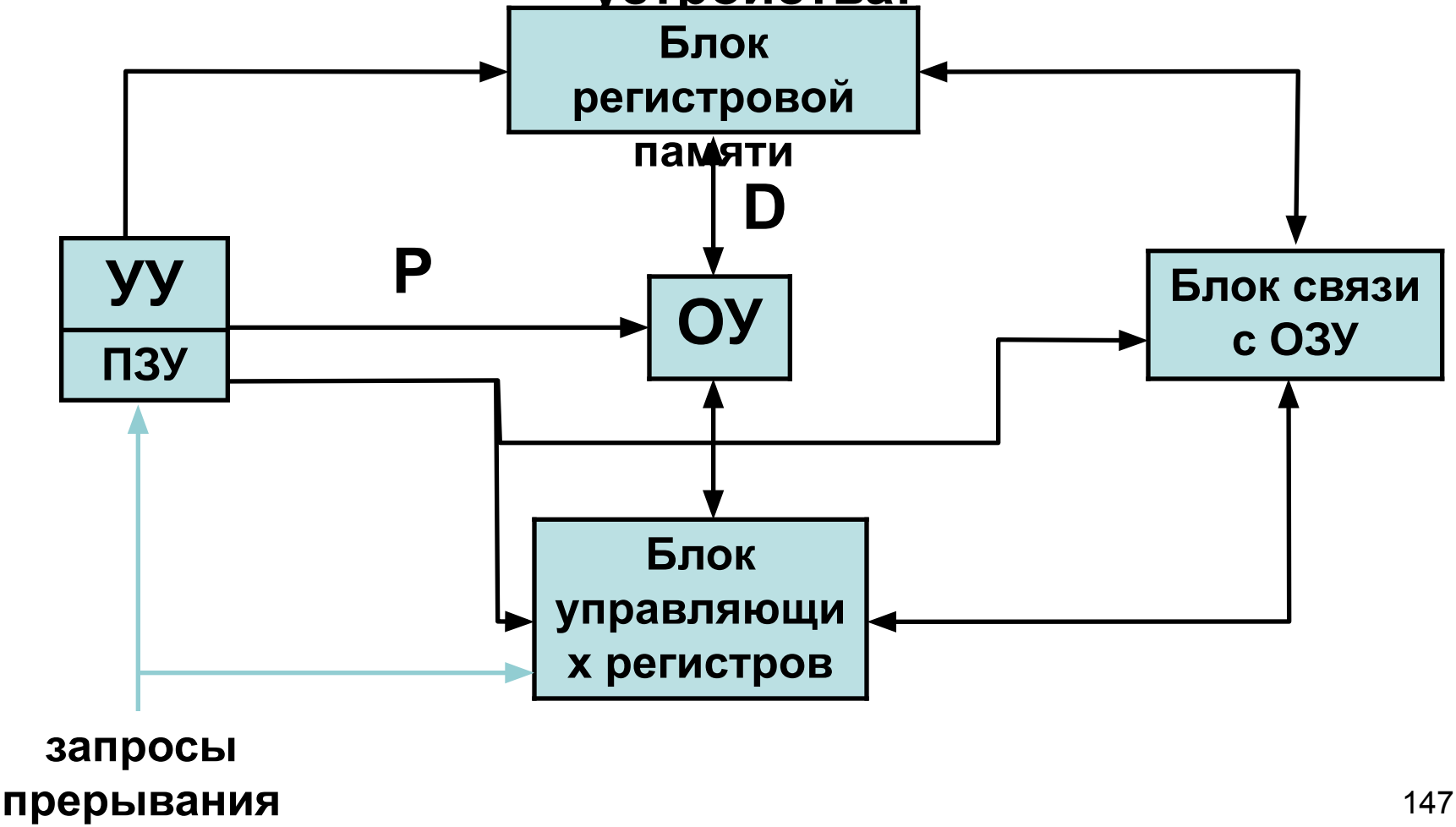
- **Машинный такт процессора регламентирует интервал времени, в течение которого выполняется одна или несколько микроопераций**
- **Границы тактов задаются схемой синхронизации**

<u>Иерархия выполнения программы:</u>

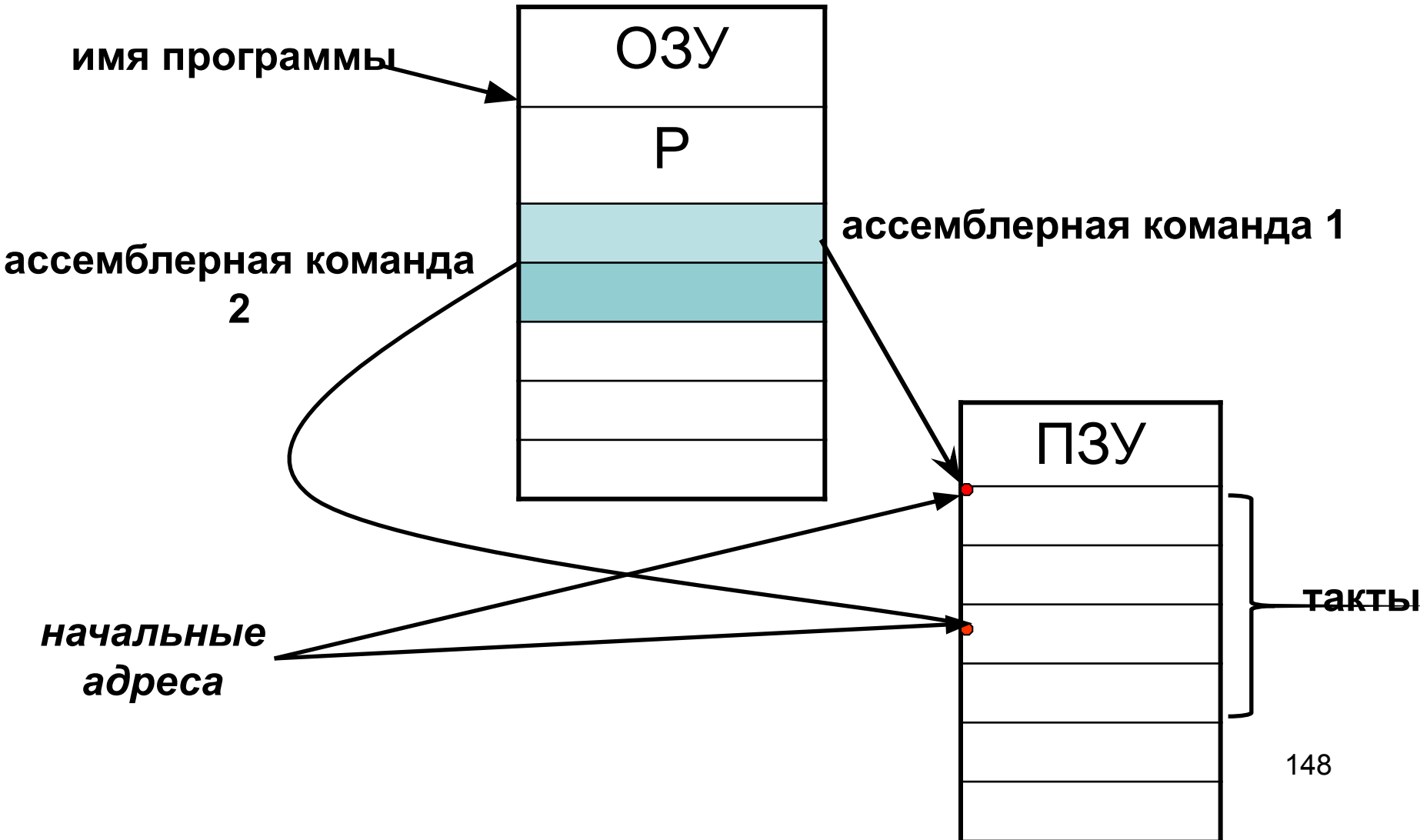
1. **Микрооперация – 1 такт**
2. **Команда ассемблера – несколько тактов**
3. **Программа – множество команд ассемблера**

Строение процессора

В простейшем случае процессор содержит следующие устройства:



ОЗУ


имя программы
ОЗУ
Р
ассемблерная команда 1
ассемблерная команда 2
ПЗУ
такты
начальные адреса

Функции операционного устройства

- непосредственное преобразование информации над данными постоянной или переменной длины (в формате фиксированной и плавающей запятой)
- модификация кодов команд

В современных процессорах операционное устройство двухуровневое:

- **в центральном процессоре выполняются арифметико-логические действия формата фиксированной запятой**
- **операции формата плавающей запятой выполняет арифметический сопроцессор**

Устройство управления

- формирует необходимые управляющие сигналы для выборки очередной команды из ОЗУ,
- дешифрации кода операции,
- формирование адресов операндов,
- выборки операндов из ОЗУ,
- передача операндов в операционное устройство,
- выполнение операций операционным устройством,
- передача результата из операционного устройства в ОЗУ,
- инициирование операции ввода/вывода,
- организация реакции процессора на запросы прерывания.

Блок управляющих регистров

- предназначен для временного хранения управляющей информации
- содержит как регистры, так и счетчики

Интерфейс процессора

Обеспечивает:

- обмен информацией
- защиту сегментов ОЗУ от недозволенных обращений
- связь процессора с периферийными устройствами

Блок контроля и диагностики

Служит для:

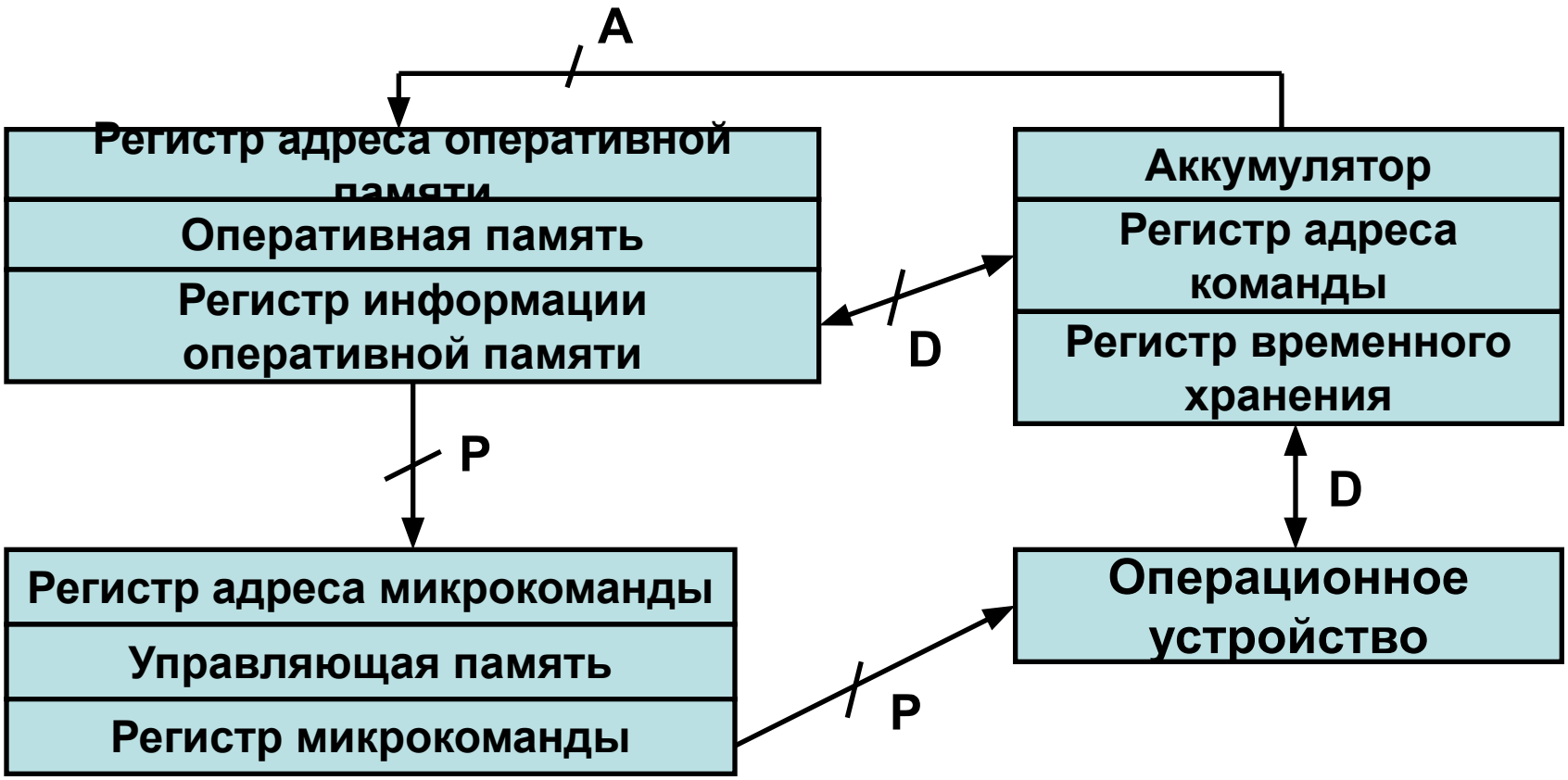
- обнаружения сбоев и отказов в аппаратуре
- восстановления работы после сбоя
- поиска, локализации и идентификации отказов

Сбои и отказы

- Сбой - это кратковременный отказ
- Для предотвращения ошибок вводятся специальные контрольные регистры, определяющие чётность количества единиц в переданном коде
- При сбое аппаратура автоматически генерирует запрос на получение кода, пока тот не станет верным, либо количество попыток не превысит заранее определенное значение
- При превышении этого значения сбой считается отказом, то есть постоянной неисправностью
- Контроль происходит непрерывно и параллельно с основным вычислительным процессом и должен быть максимально полным

Микропрограммная реализация команд ЭВМ

Простейшее устройство ЭВМ



Группа команд межрегистровой пересылки данных

В группу входят команды типа *ADD*
(Акк := Акк + (А) или if (условный переход)

В группе работают следующие правила:

- регистр информации оперативной памяти и регистр адреса оперативной памяти способны получать данные из любых других регистров
- любому регистру можно присвоить данные из регистра информации оперативной памяти
- для передачи данных используется коммутатор

Микрооперации управления выборкой регистра

В этой группе используются следующие правила:

- любому регистру можно присвоить значение аккумулятора, регистра адреса команды и регистра временного хранения
- используется полный коммутатор

Считывание и запись информации оперативной памяти

В эту группу входит возможность обмена

данными между регистром информации

оперативной памяти и регистром

оперативной памяти в точке, указанной

регистром адреса оперативной памяти

Адресная арифметика

В эту группу входит возможность обмена данными между аккумулятором и любым другим регистром, арифметическое сложение и вычитание, а также перебор значений

Регистр адреса микрокоманды

В эту группу входит реализация счетчика и получение регистром адреса микрокоманды значения регистра информации оперативной памяти

Учебный курс

Принципы построения и функционирования ЭВМ

Лекция 12

Архитектура ЭВМ. Прерывания.

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Распараллеливание вычислений

- производится для повышения производительности вычислительной системы
- необходим аппаратный параллельный коммутационный ресурс

1. Способы распараллеливания вычислений

- Векторизация. Преобразования осуществляются параллельно над несколькими потоками.
- Конвейеризация. Один и тот же поток команд или данных проходит несколько фаз преобразований, каждая из которых поддерживается своим аппаратным ресурсом.

2. Конвейер команд

Команды проходят 3 фазы:

выборка, дешифрация, исполнение.

- Ресурсы:
- Выборка – процессор, ОЗУ
- Дешифрация – устройство управления процессора (УУ)
- Исполнение – операционное устройство (ОУ)

2.1. Организация циклов

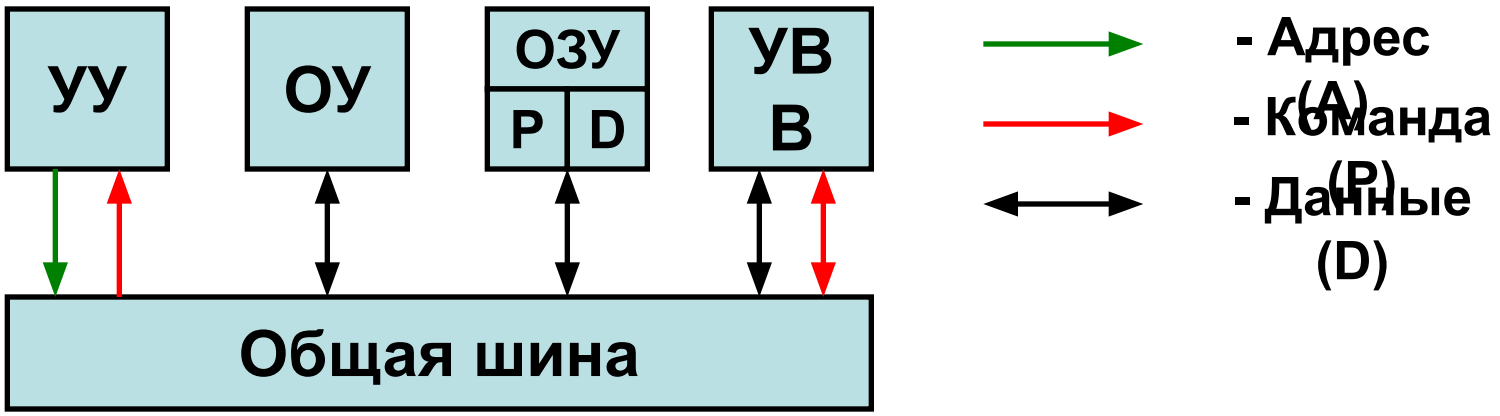
- Циклы применяются для решения проблемы конечности памяти
- Использование цикла в конвейере существенно уменьшает производительность
- $T_{общее} = T_{цикла} + \Delta T_{конвейера}$
- Чем длиннее конвейер, тем больше издержки
- Глубина конвейера ограничена набором команд условного перехода

3. Векторизация

- Несколько процессов должны выполняться одновременно и параллельно

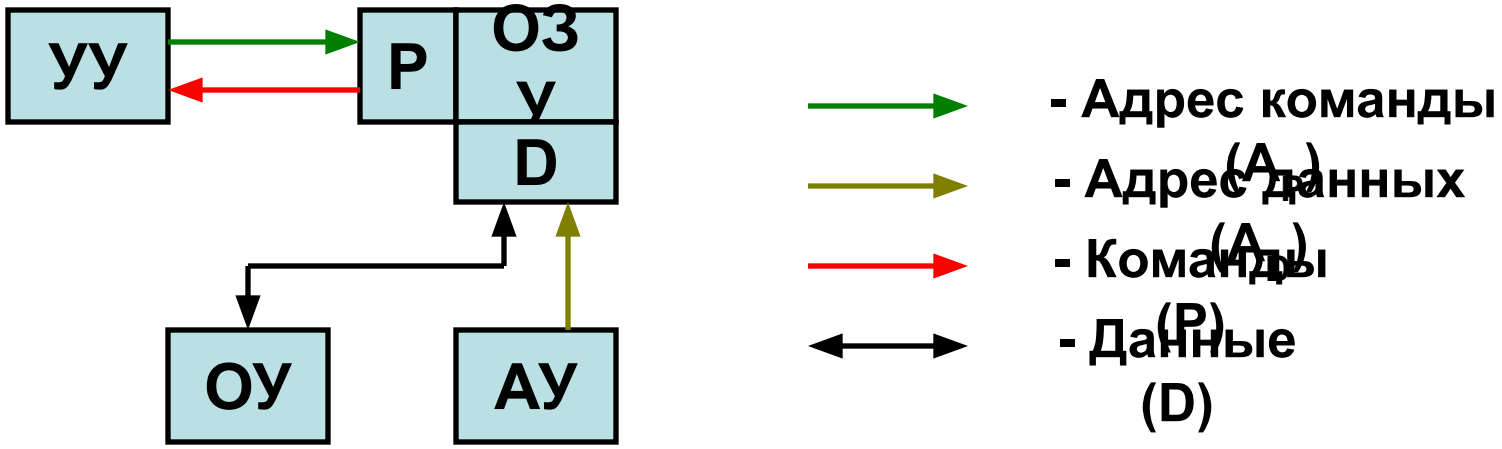
- Команды становятся протяжённей, но выполняются за меньшее количество тактов

3.1. Архитектура Фон-Неймана



- Недостаток архитектуры – общая шина
- Решение проблемы – организация раздельных шин для команд и данных

3.2. Гарвардская архитектура



- Используются раздельные шины для адресов и данных
- Обращение в память от УУ и АУ происходит параллельно

3.3. RISC-архитектура

- Архитектура с редуцированной системой команд
- Все команды выполняются за равное количество тактов, что позволяет произвести их внутреннюю конвейеризацию
- На ассемблерном уровне реализуются только те команды, которые отвечают единственной стандартной процедуре исполнения

4. Система прерывания

- Изначально СП появилась для повышения надежности вычислительного процесса
- Инженерное решение состояло в том, чтобы периодически снимать состояние регистров процессора и сохранять в ОЗУ
- В дальнейшем СП стали использовать для мультипрограммного режима работы

- Совокупность информации, необходимой для восстановления вычислительного процесса называется вектором состояния или словом состояния (PSW)
- Вектор состояния в каждый момент времени содержит информацию, достаточную для продолжения выполнения программы или для повторного пуска в контрольной точке

4.1. PSW фирмы Intel

Кроме регистров общего назначения (РОН) существует информация, которая хранится в регистре FLAG. FLAG отмечает события.

Основные флаги:

- CF – флаг переноса. Используется для команд сдвига или циклического сдвига.
- PF – флаг паритета. Используется для установления четности или нечетности паритета.

- AF – вспомогательный флаг переноса. Используется для двоично-десятичной арифметики.
- ZF – признак нуля.
- SF – флаг знака (1 = «-»; 0 = «+»).
- OF – флаг переполнения.
- TF, IF, DF, VM – предназначены для прямых действий процессора.

- TF – «1» - пошаговый режим. Запрет перехода к следующей команде.
- IF – флаг разрешения прерывания.
- DF – контроль направления цепочки операций.
- VM – флаг виртуального режима (виртуальная машина). Переводит процессор в режим эмуляции команд процессора более ранней версии.

4.2. Управление системой

- Все периферийные устройства также характеризуются своим состоянием
- Управление вычислительным процессом осуществляется с помощью анализа слов состояния устройств ЭВМ
- Периферийные устройства должны просигналить центральному процессору о готовности обменяться информацией

Учебный курс

Принципы построения и функционирования ЭВМ

Лекция 13

Система прерывания. Команды ввода/вывода.

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Система прерывания

- Характеристики системы прерывания
- Показатели качества работы системы

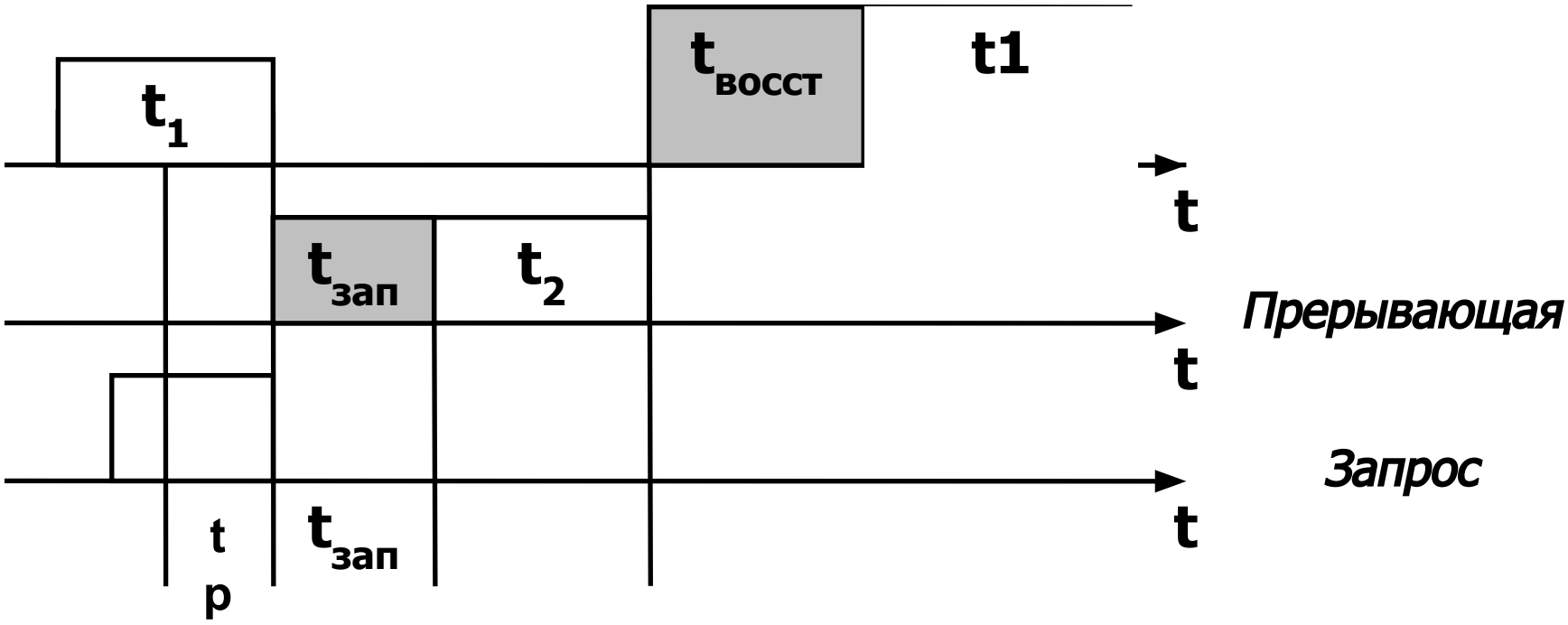
Основные функции системы прерывания

- обнаружение и идентификация типа прерывания
- запоминание состояния прерываемой программы и управление переходом к прерывающей программе
- восстановление состояния прерванной программы и продолжение выполнения

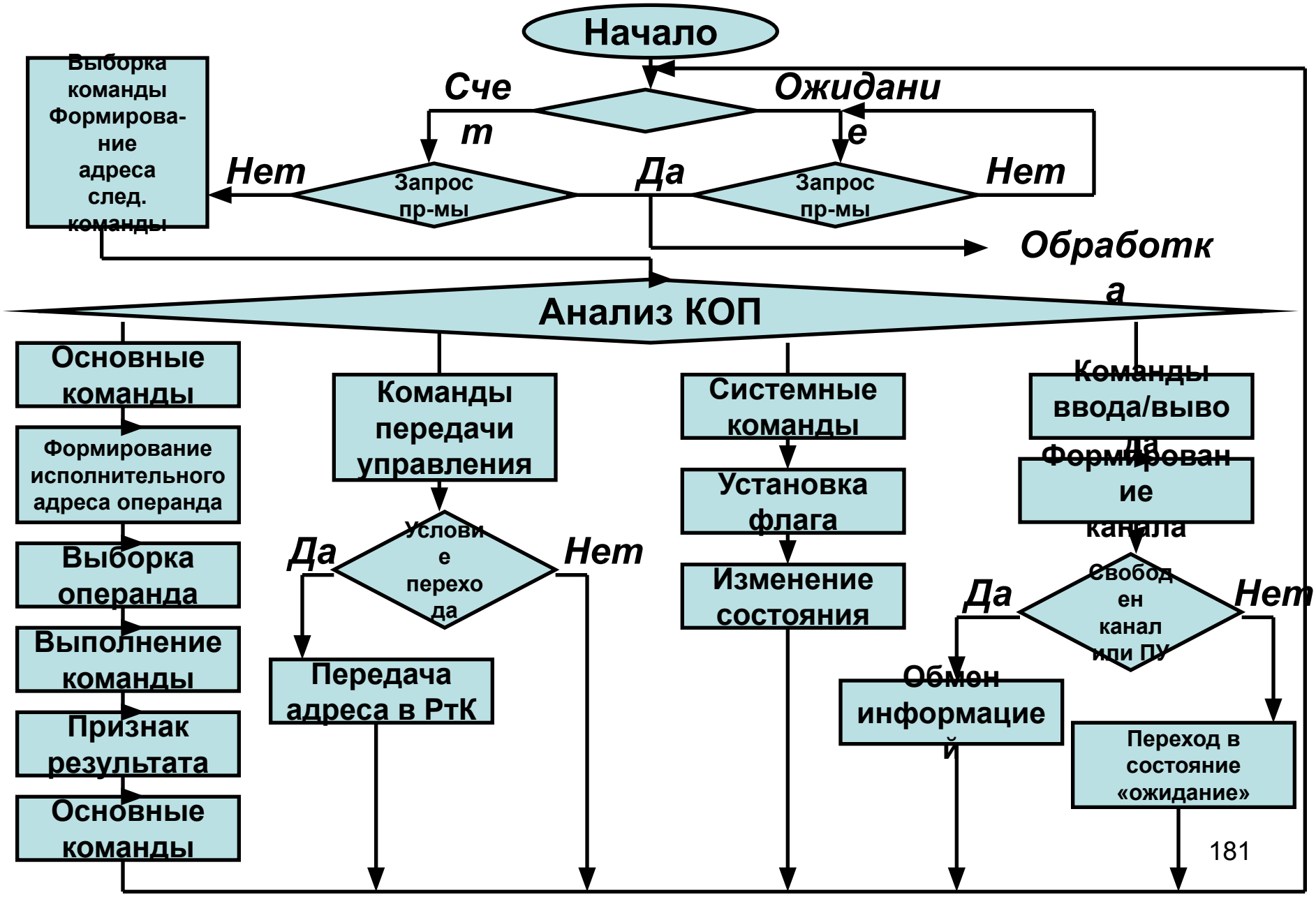
Характеристики системы прерывания

- общее количество запросов прерывания
- время реакции системы
- глубина прерывания
- насыщение системы прерывания
- допустимые моменты прерывания программ
- число уровней прерывания

Время реакции системы



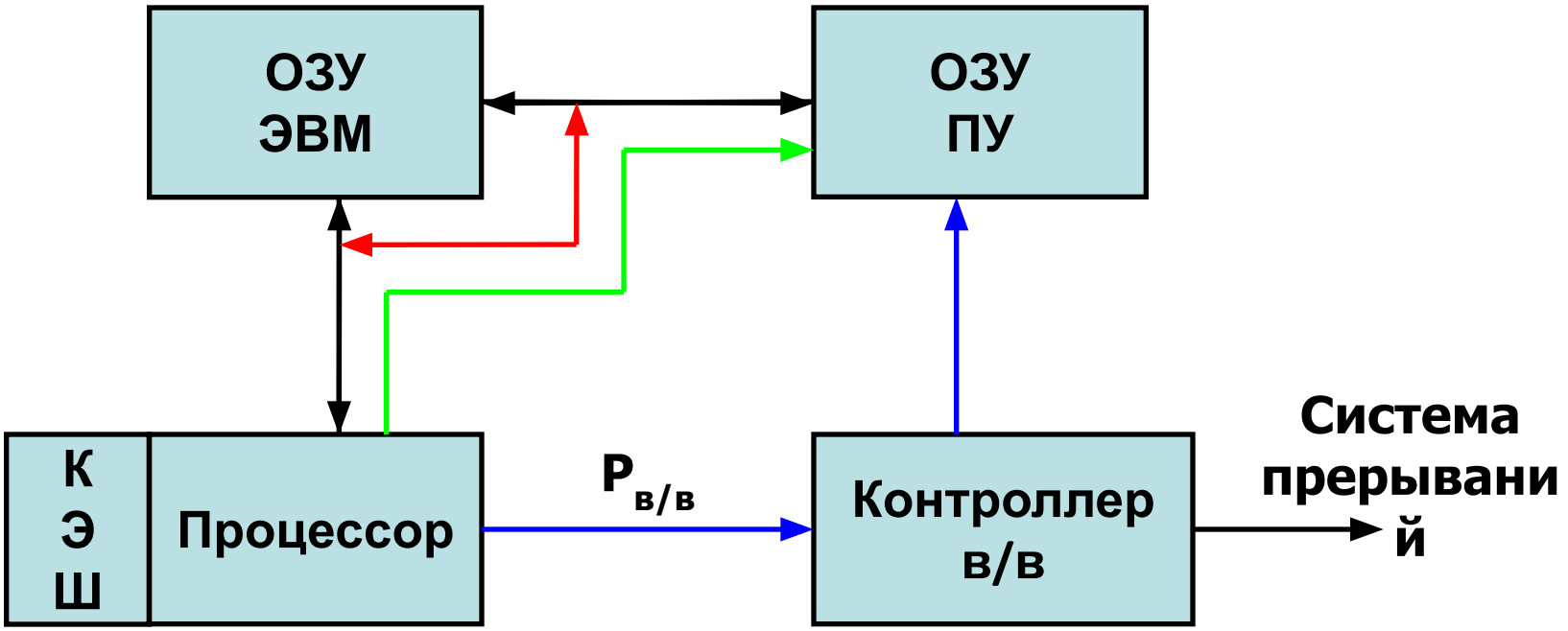
Полный цикл выполнения программы



Специфика выполнения команд ввода/вывода

Основная специфика порождается относительно медленной работой устройств

Выполнение команд ввода/вывода



Учебный курс

Принципы построения и функционирования ЭВМ

Лекция 14

Организация вычислительных сетей

профессор ГУ-ВШЭ, доктор технических наук
Геннадий Михайлович Алакоз

Два способа повышения производительности

1) ↑ F_T (повышение тактовой частоты)работы, поддерживаемая технологией производства

$V_{физ}$(физическая производительность) ≤ F_T

возможность выполнения 1 микрокоманды

быстрее, чем за 1 такт

$V_{физ} = F_T$ –достигается в RISC-архитектура

$V_{пользователя} \geq L_p/T_з$- в пользовательской архитектуре, где:

L_p-кол-во команд критического пути

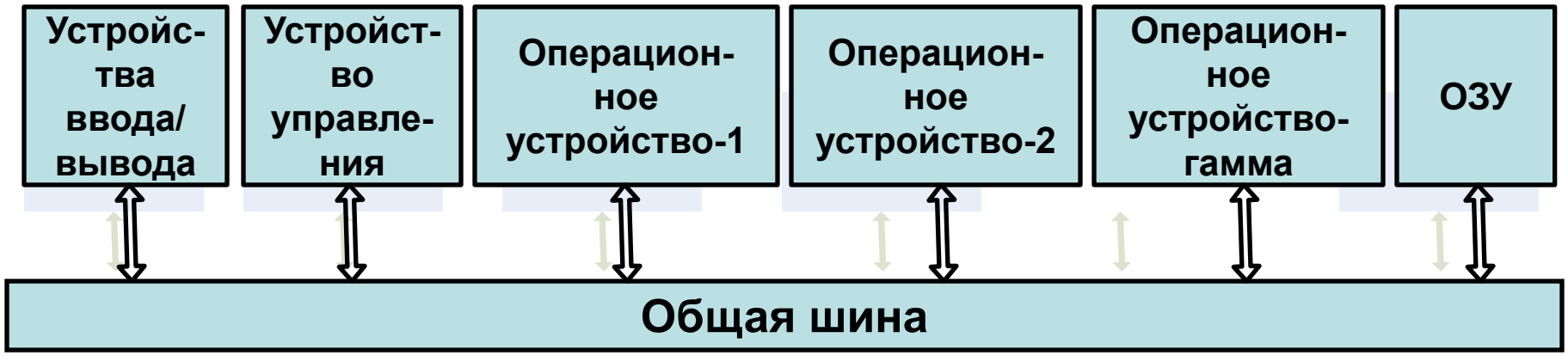
Распараллеливание вычислений

2) Распараллеливание вычислений

$V_{физ} \le F_T * \gamma$, где:

γ-коэффициент распараллеливания вычислений ($\gamma \ge 1$)

Увеличение количества исполненных команд за 1 такт работы
Повышение производительности в фон-неймановской архитектуре возможно, когда *тш* ≤ *Lp*тц*γ*, где:
***тц* = 1/*FT*, *тш* - цикл срабатывания шины**
Увеличение производительности шины



Идеал системы коммутации

- Отсутствие конфликтов (блокировок) запроса на передачу команд и данных между всеми без исключения устройствами ЭВМ: операционными, блоками памяти, устройствами вода/вывода и т.д.

- Основная задача- ликвидирование конфликтов в системе коммутаций

Коммутационный аспект

- В данном аспекте любая вычислительная система представляет собой сеть, узлы которой связаны трактами передачи данных (каналами)
- Узлами могут быть: процессоры, модули памяти, устройства ввода/вывода, коммутаторы, концентраторы
- Некоторые узлы могут быть объединены в определенные коммутационные группы

Организация внутренней коммутации и топология

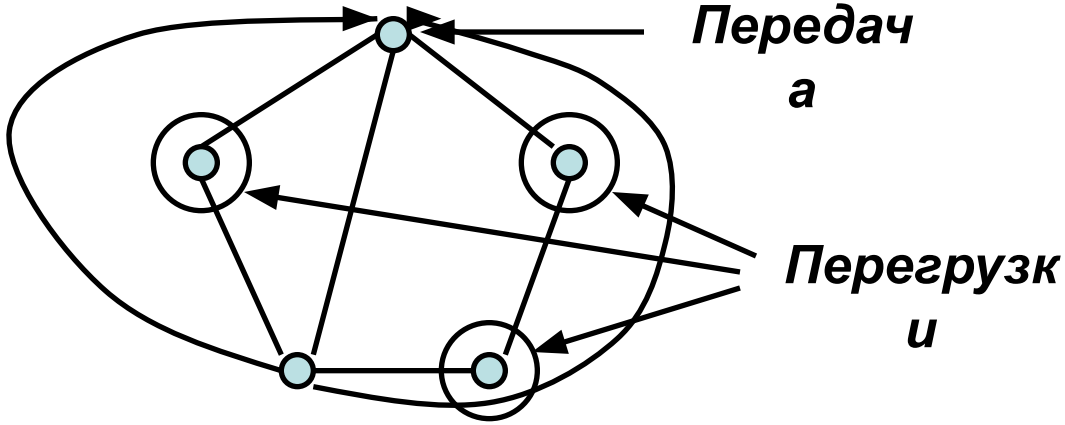
Организация внутренней коммутации вычислительной системы называется ее топологией, определяемая множеством узлов и множеством связей

$Ui \in U$ **- узел**

$Kj \in K$ **- связь**

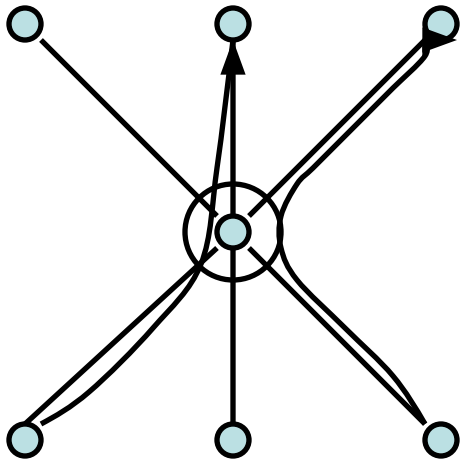
Примеры топологии

- Кольцевая топология сети



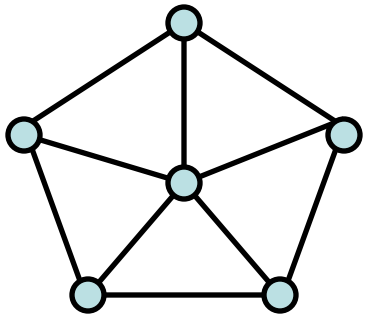
Достоинства:
Передать сообщение можно в любую точку сети,
но
только двум ближайшим соседям напрямую,
остальным - через транзитные пункты

- Радиальная топология

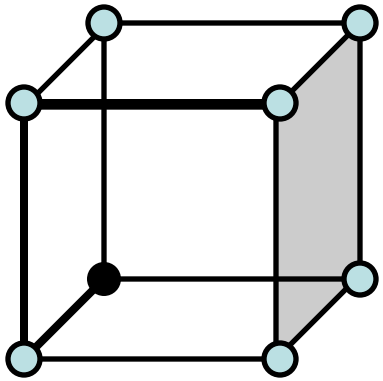


Достоинства:
Не больше двух плеч для передачи любому абоненту
Но происходит перегрузка центра коммутации, в результате данная топология превращается в линейную

- Радиально кольцевая топология



- **Топология типа гиперкуб**



Каждый узел данной сети связан с *n* соседями
(в данном случае *n*=3)
Транзитных пунктов у данной сети не более
***n*-1 (в данном случае транзитных путей 2)**

Достоинства:
Потеря 1 узла только сокращает число вариантов
транзитной передачи данных, но не уничтожает
возможность обмена между узлами сети по

Смежные узлы и каналы

Смежные узлы - узлы, между которыми есть прямой канал обмена

Канал характеризуется:

- шириной (количеством сигнальных линий)
- частотой (скоростью передачи одной выборки данных)
- начальной задержкой на передачу

Начальная задержка

- от источника к приемнику
- физическая

  $d/V_к = t^o к$, где:

  d - длина канала

  $t^o к$ - начальная задержка
- логическая

“Грамматика” передаваемого сообщения

Сущность в регламенте и уровне сигналов

в каждом канале

- Параметры электромагнитных сигналов и правила кодирования логических переменных на физическом уровне
- Структура передаваемого сообщения на логическом уровне

Адрес исполнителя-адрес приемника-байт

перед сообщением

Топология сети и три главных атрибута сети

- статическая
- динамическая (программируемая)

Главная задача - связать алгоритм с коммутационными возможностями сети

Атрибуты:

- **стратегия синхронизации**
- **стратегия коммутации**
- **стратегия управления**

Стратегия синхронизации и коммутации

Сети

синхронные **(все фиксировано)**

асинхронные **(обмен информацией происходит между двумя активными узлами по принципу рукопожатия)**

Стратегия коммутации

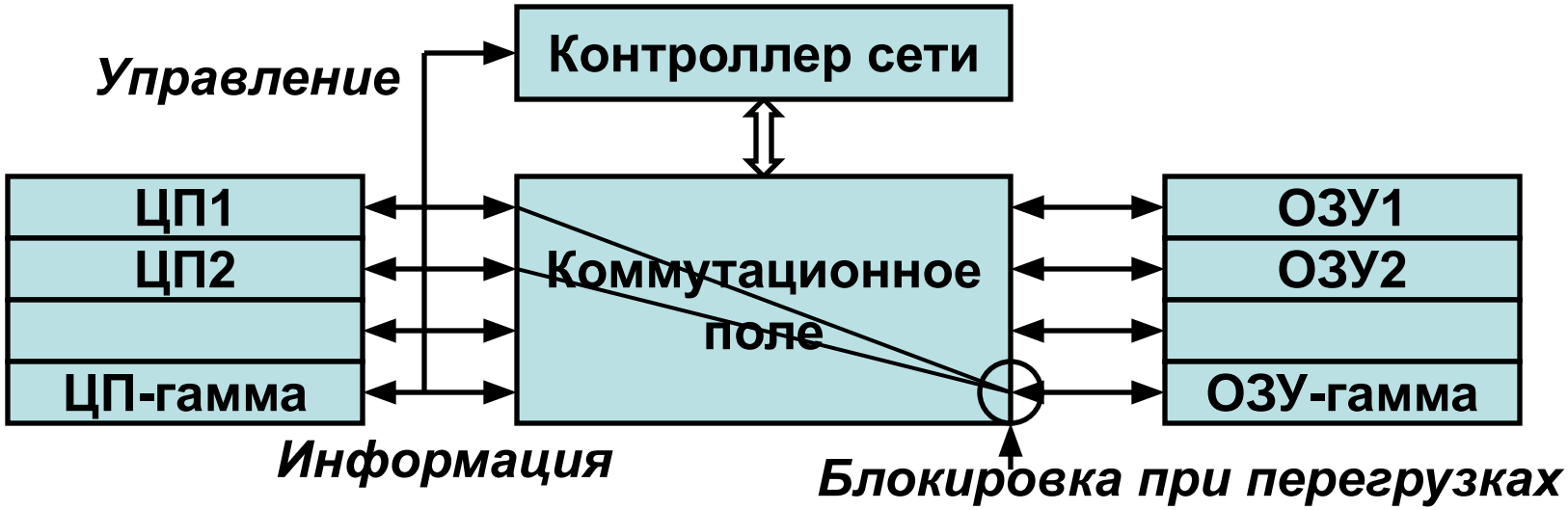
коммутация каналов **(прямая физическая связь между источником и приемником)**

коммутация сообщений **(связь между источником и приемником устанавливается через пакеты, доставляемые в ближайшие направления)**

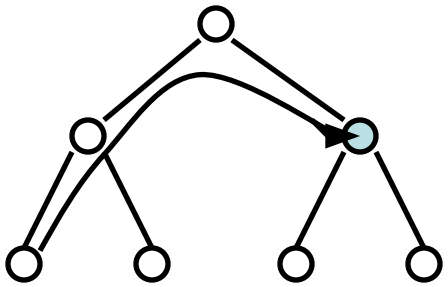
Недостаток:задержка на формирование пакета, издержки на сортировку

Стратегии управления

- Централизованная (единый центр управления)

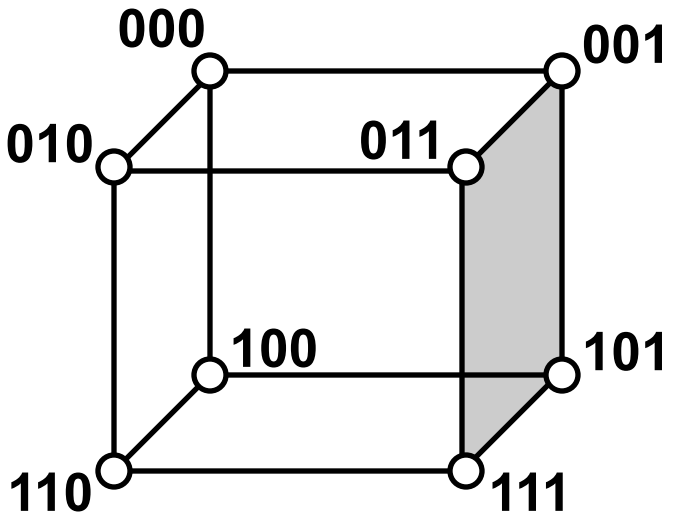


- **Децентрализованная**

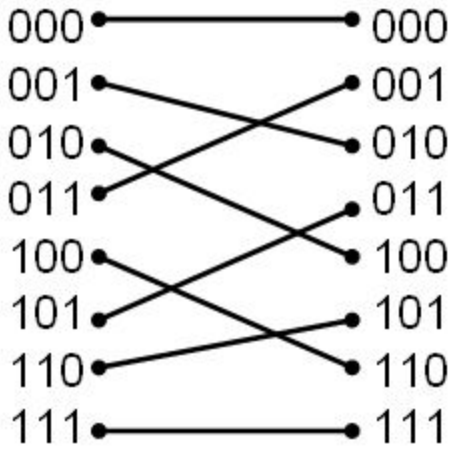


Поочередный проход через все узлы к нужной точке

Маршрутизация потоков в сети и сеть идеальной тасовки



Сеть идеальной



Правило кодировки: изменение содержимого 1 бита

В топологическом гиперкубе изменение маршрутизации осуществляется инверсией 1 бита:

вправо – младший, влево – средний, по вертикали – старший

Данная сеть реализует преобразование адресной части по правилу циклического сдвига влево